Проект „Повишаване квалификацията на служителите от администрацията на централно ниво чрез усъвършенстване на знанията и практическите им умения за управление на софтуерни ИТ проекти в съответствие със съвременните методологии“, осъществяван с финансовата подкрепа на Оперативна програма „Административен капацитет” (ОПАК), съфинансирана от Европейския съюз, чрез Европейския социален фонд”, съгласно Договор № К13-22-1/05.03.2014 г.

НАРЪЧНИК

Дейност 7 “Провеждане на обучение за управление и администриране на уеб портал за 31 служители на централната администрация и издаване на сертификати за проведеното обучение”

Изготвен в изпълнение на Договор № Д-37/11.12.2014 г.
между
МИНИСТЕРСТВО НА ТРАНСПОРТА, ИНФОРМАЦИОННИТЕ ТЕХНОЛОГИИ И СЪОБЩЕНИЯТА
и
„КОНСОРЦИУМ ИТ ОБУЧЕНИЯ 2015“ДЗЗД

„КОНСОРЦИУМ ИТ ОБУЧЕНИЯ 2015“ ДЗЗД

София 1040, ж.к. Изток, бул. Драган Цанков 36, СТЦ Интерпред, блок А, ет.6; тел: 024210040; имейл: ittraining2015@newhorizons.bg;

Авторски колектив:

Никола Баненкин;

Божил Божилов.

Одобрил:

Николай Пенев – Ръководител проект

София, 2015 г.

Съдържание:

Модул 1: Следващото поколение портали за електронно правителство 13
От електронно правителство към електронно управление 13
Електронно правителство 13
Електронно управление (е-управление) 13
Принципи на електронното управление 13
Законова база 14
Стандарти и термини 14
Програма за електронно правителство в т.ч. уеб портали, с основа и цел гражданите, бизнеса и обществото 15
Стратегия за развитие на електронното управление в Република България 2014 – 2020 г. 15
Визия 15
Модел на е-управление 16
Пътна карта 17
Целеви области и ползи, посочени в пътната карта 17
Специфични цели: 18
Принципи на електронното управление: 18
Електронна административна услуга (ЕАУ) 18
Примери от развитието на електронното правителство и уеб портали в България 20
Ползи от интегрирано електронно правителство чрез уеб портали 22
Модул 2: Мобилни и облачни технологии за уеб портали 22
Електронно и мобилно управление 22
Мобилни приложения в правителствения сектор 22
Характеристики на „големите данни“: 23
Предизвикателства пред Big Data: 23
Поява и еволюция на облачните технологии 23
Типове облаци 24
Облак от публични услуги 25
Характеристики и условия за пилотни проекти 25
Ползи от правителствения облак 25
Модул 3: Въведение в HTML 26
Основна концепция на уеб 26
HTTP заявки 26

Структура на HTML документа 26
Тагове 28
Атрибути 29
Модул 4: Форматиране с HTML 29
Форматиране на уеб страници 30
Списъци 30
Таблици 31
Хипервръзки 33
Изображения 34
Управление на текстовия поток 35
Модул 5: Форми и структура на HTML 36
HTML форми 36
Събиране на потребителски вход 36
Валидация 39
Структурни елементи 39
Навигация и менюта 40
Модул 6: Контрол на цвета и типографията 40
Въведение в CSS 41
CSS селектори 41
Начини за прилагане на стилове 42
Позициониране на блокови елементи 44
Block layout модел 44
Цветове и градиенти 45
Промяна на стила на шрифта 47
Шрифтове: 48
Създаване на свързан style sheet 49
Модул 7: CSS селектори 49
CSS селектори 50
HTML наследяване 51
Групиране на селектори 52
Псевдо елементи и псевдо класове 52
Модул 8: Проектиране на раздели на съдържанието 53
CSS Box Model 53

Inline и Block елементи 54
Размери на елемент и позициониране 55
Позициониране на елементи 55
Overflow и resize 56
Модул 9: Анимации с CSS 57
CSS преходи 57
Трансформация на елементи 59
Анимации 60
Модул 10: Изработка на адаптивен потребителски интерфейс 62
Поддръжка на различни устройства 62
Media Types 62
Media Queries 63
Дефиниране на Style Sheets за печат 64
Модул 11: Въведение в JavaScript 64
Структура и употреба 65
Функции и оператори 65
Библиотеки 66
Връзка между JavaScript и HTML 66
JSON 67
Document Object Model 67
jQuery 68
Node JS 69
Knockout JS 69
Модул 12: AJAX 70
HTTP 70
Fiddler 71
AJAX 71
AJAX и jQuery 72
Модул 13: Структура на JavaScript 73
Област на действие на променливите (scope) 73
Създаване на прости обекти 74
Object Literal Notation 75
Конструктори 75

Прототипи 76
Капсулация 77
Наследяване чрез Chaining Prototypes 77
Модул 14: Реализиране на графика чрез CSS и JavaScript 77
Създаване на интерактивни графики чрез SVG 78
Рисуване на графика с Canvas API 82
Модул 15: Разработване на интерактивни страници чрез HTML5 APIs 84
Работа с файлове 84
Drag-and-drop (провлачване) 85
HTML видео 87
HTML аудио 87
HTML5 Геолокация 88
Модул 16: Какво представляват CMS системите 89
Въведение в CMS 89
Основно разделение на видовете CMS 89
Основни характеристики 90
Уеб система за управление на съдържанието 91
Компонентна система за управление на съдържанието 91
Joomla 92
PrestaShop - CMS за електронна търговия 93
CMS Hosting – физическото местоположение на сайтовете изградени чрез CMS 94
Какво е SharePoint 95
Изграждане на CMS 96
CMS шаблони и допълнителни приложения 97
Редактиране и управление на съдържание 98
SharePoint продукти 98
Технологии в SharePoint 99
Възможности на SharePoint 100
Модул 17: Защо имаме нужда от CMS 100
Статични уеб сайтове 101
Предимства на статичния уеб сайт 102
Недостатъци на статичния уеб сайт: 102
Основи на софтуерната архитектура 102

Уеб архитектура 103
Сигурност, архитектура и мобилност на уеб портала 104
Практически примери от правителствени уеб-портали: 105
Изисквания за инсталация SharePoint 2013 106
Минимални хардуерни изисквания 108
Инсталиране на SharePoint 109
Логическа архитектура на SharePoint 116
Модул 18: Нуждата от уеб портали и системи за управление на съдържанието 118
Нуждите на бизнеса 118
Кой уеб сайт е успешен 119
Възможности на CMS 120
Дизайн на уеб портали 121
Основни възможности на CMS системите 122
Категоризиране на CMS 123
Информационна архитектура в SharePoint 2013 125
SharePoint Site columns 126
SharePoint Content Types - преизползваема колекция от метаданни 126
Видове страници в SharePoint 2013 127
Страница с уеб части 127
HTML страница 128
Модул 19: Облачна архитектура за услуги с масово приложение 128
Специализирани бизнес решения 128
Облачните решения 129
Определящи характеристики 130
Видове облачна архитектура 130
Проблеми при възприемането 131
Доставка на облачни технологии 132
Infrastructure as a Service – Инфраструктура-като-услуга 133
Platform as a Service – платформа-като-услуга 134
Software as a Service – Софтуер-като-услуга 135
SharePoint Online 135
Microsoft Azure 136
Модул 20: Употреба на уеб аналитика (web analytics) и контрол в портали 137

Какво е уеб аналитика ... 137
Видове web analytics ... 138
Следене на събитията ... 139
Разлики между порталите и обикновените сайтове ... 140
Контрол на версиите в SharePoint ... 141
Особености на контрола на версии ... 141
Recycle Bin ... 143
Оторизация ... 144
Планиране на оторизацията ... 145
Управление на нивата за достъп ... 146
Управление на потребителските групи ... 147
Аутентикация – процес на идентификазия ... 148
Claims-based аутентикация ... 149
Security Token Service Application ... 150
Windows Claims-mode аутентикация ... 150
Планиране на NTLM и Kerberos аутентикация ... 151
Модул 21: Интегриране на търсене в уеб портал ... 151
Защо да интегрираме търсене ... 152
Търсене и URLs ... 153
Crawling ... 154
Главни аспекти на търсенето ... 155
Search процеси ... 155
Search Index ... 157
Crawled Properties - данни за елемент от източника на информация или част от този елемент . 157
Managed Properties - отделни crawl properties или съвкупност от такива, които са конфигурирани за специфични за търсенето цели ... 158
Search Schema ... 158
Full crawl (пълно обхождане) ... 159
Incremental crawl ... 159
Езици за заявки в SharePoint ... 160
Search APIs ... 160
Изпълняване на Web Service заявки ... 161
Изпълнение на Representational State Transfer (REST) заявки ... 161

Модул 22: Предизвикателства пред съвместимостта на портали с множество мобилни устройства 162
Въведение 162
N x M матрица 163
За какво да се внимава 164
Удобство при достъп 164
Използване на ресурсите 165
Web Parts 165
SharePoint Apps – приложения, които работят в SharePoint среда 166
Инсталиране на SharePoint Apps 166
App Permissions 167
Device Channels (Канали за Устройства) 167
Master pages 168
Page Layouts 169
Модул 23: Подобряване на подхода към проекти, свързани с Интернет и SOA 170
Възходът на интернет 170
Какво е SOA? 170
Преимущества на SOA 172
Уроци и заключения от имплементации на SOA (Service Oriented Architecture) 172
SharePoint Service Applications 173
Функции на Service Application 174
Инстанции на services и Service Application Dependencies 174
Създаване на Service Applications 175
Service Application проксита и прокси групи 175
Стартиране и спиране на инстанции 176
Модул 24: Контейнери и конектори като елементи от дизайна на портала 176
Web Applications 177
Логическа инфраструктура на web applications 177
Web Application зони 178
Управление на Content Databases 178
Проектиране и конфигуриране на Managed Paths 179
Resource Throttling 179
Site Collection и Webs 180

Настройване на Site Collections 181
SharePoint Навигация 182
Site Templates (Шаблони на сайтове) 183
Модул 25: Подобряване на потребителското преживяване от портала 184
Потребителски профили в SharePoint 184
Планиране на импортиране на потребителски профили 185
Създаване на User Profile Service Application за импорт от Active Directory 186
Конфигуриране на настройките на потребителските профили в SharePoint 186
Audiences (Аудитории) 187
Таксономия в SharePoint 187
Term Sets(Множества от термове) 188
SharePoint Managed Navigation (Управляема навигация на SharePoint) 189

Речник:

Web Part (Уеб част)	Обособена фунционалност в контекста на SharePoint Server, която може да има и визуална част и може да се преизползва в различни страници
Content Management System (Система за управление на съдържанието)	Компютърна програма, която позволява публикуването и редактирането на съдържания, както и поддръжка на главен интерфейс.
Component CMS (Компонентна CMS, CCMS)	CMS, Която р азделя информацията на гранулярно ниво, вместо на документи. Отделни заглавия, изображения и съдържание се разделят на единици информация)
Search Engine Optimisation	Практики за изграждане на електронно съдържание, така че то да бъде по-лесно обработвано от търсачките
PowerShell	Система за автоматизация на управлението на задачите и настройване на продуктите на Microsoft
.NET Framework	Платформа за разработка на приложения на Microsoft
Internet Information Services (преди Internet Information Server, IIS)	Сървърно приложение на Microsoft за предоставяне на информация по мрежа, което работи с повечето популярни протоколи
ASP.NET	Технология на Microsoft за разработване на уеб приложения
Farm (Ферма)	Структура от SharePoint сървъри, която организира архитектурата на най-високо ниво
Service application	Логически компонент на SharePoint, който осигурява ресурси които могат да бъдат споделени между сайтове във ферма или между отделни ферми
Application pools	Група от един или повече сайтове които се обслужват от определен процес на операционната система
Web application	Логически компонент на SharePoint, който реализира групиране на функционалност и общи настройки между колекции от сайтове
Zone	Логически компонент на SharePoint, който се използва за да се осигури достъп до един и същ web application през различни адреси
Site collection	Това е логически обособено множество от сайтове, които имат общ главен сайт и споделят определени административни настройки
Lists и Libraries (Списъци и библиотеки)	Контейнери за данни, които съхраняват елементи с еднаква структура като контакти, задачи или документи
Application Programming Interface	И нтерфейсът на изходния код, който операционната система или нейните библиотеки от ниско ниво предлагат за

	поддръжката на заявките от приложния софтуер или компютърните програми.
REST	Разпределителна системна рамка, базирана на уеб протоколи и технологии. Архитектурният модел Rest включва взаимодействията между сървър и клиент, осъществени по време на трансфера на данни.
HTML - HyperText Markup Language (език за маркиране на хипертекст)	Основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в Интернет, а правилата се определят от международния консорциум W3C. Текущата версия на стандарта е HTML 5.0
HTTP – hypertext transfer protocol (протокол за трансфер на хипертекст)	Мрежов протокол, от приложния слой на OSI модела, за пренос на информация в компютърни мрежи. Създаден като средство за публикуване на HTML страници, протоколът довежда до формирането на Световната уеб мрежа.
Таг	Градивната частица на HTML и XML документите.
XML - eXtensible Markup Language (разширяем маркиращ език)	Стандарт (метаезик), дефиниращ правила за създаване на специализирани маркиращи езици, както и синтаксисът, на който тези езици трябва да се подчиняват.
CSS – cascading style sheets (език за стилови листове)	Използва се основно за описване на представянето на документ, написан на език за маркиране.
Метаданни	Данни които ни съдържат информация за самата страница.
Browser (браузър)	Програма, която се използва за преглеждане на документи с хипермедия и уеб навигация.

Модул 1: Следващото поколение портали за електронно правителство

От електронно правителство към електронно управление

Електронно правителство

Електронното правителство е сложна и комплексна структура, която се гради на няколко основни стълба - услуги от администрацията за гражданите, от администрацията за бизнеса и от администрацията за администрацията, от една страна, междуведомствена свързаност и системна и оперативна съвместимост, осигуряваща служебен обмен на данни, който повишава рязко ефективността на институциите и електронни системи, които подобряват вътрешноведомствената ефективност и ефикасност. Най-видимата част от всичко това са услугите, предлагани от общинските и централните администрации на гражданите и бизнеса, както и възможността на институциите да обменят лесно и бързо информация по служебен път.
Е-правителството е малка част от електронното управление.
Е-управлението е инструмент за изграждането на ефективно работеща администрация.

Електронно управление (е-управление)

Обхваща съвременните информационни и комуникационни технологии, съчетано с усъвършенстване на организационната структура и прилагане на нови умения в държавната администрация, за предоставяне на административни електронни услуги (е-услуги) по-ефективно, по-евтино и по-бързо. Електронното управление обхваща четири основни направления за комуникация и услуги:

- -“Администрация – Граждани” – включва съвременни Интернет и интранет WEB базирани решения, съчетани с традиционните средства за осигуряване на широк достъп, които да водят до качествени промени в условията за комуникиране и предоставяне на услуги за гражданите.
- “Администрация – Бизнес” – включва съвременни решения, които оптимизират процесите и деловите отношения между администрацията и различните икономически субекти.
- “Администрация – Администрация” – включва развитие на информационните технологии в национален и междудържавен аспект с оглед на ефективно взаимодействие между различните административни структури.
- “Вътрешноведомствена ефективност и ефикасност” – включва организиране и оптимизиране на бизнес процесите, на отношенията “Администрация – Служители” и на комуникацията в отделните административни структури.

Принципи на електронното управление

- Еднократно събиране и многократно използване на информация за граждани и бизнес
- Административно разпространение на информация
- Архитектура ориентирана около услуги (SOA)
- Сигурност и надеждност на обмена на информация
- Платформа, базирана на стандарти

Законова база

- ЗЕУ – Закон за електронното управление
- ЗЕДЕП - Закон за електронния документ и електронния подпис
- ЗЕС – Закон за електронните съобщения
- ЗДПУ - Закон за достъп до пространствени данни
- НРИОЕУ - Наредба за регистрите на информационните обекти и електронните услуги
- НУЕПА – Наредба за употребата на електронен подпис
- НЕАУ – Наредба за електронните административни услуги
- НЕСОЕД – Наредба за единната среда за обмен на електронни документи
- НОИОСИС – Наредба за общите изисквания за оперативна съвместимост и информационна сигурност
- НВОЕДДХНА – Наредба за вътрешния оборот на електронни документи и документи на хартиен носител в администрацията

Стандарти и термини

ЕПДЕАУ	Единен портал за достъп до електронни административни услуги
ЕСОЕД	Единна среда за обмен на електронни документи
ЗДПУ	Закон за дейностите по предоставяне на услуги
ЗЕДЕП	Закон за електронния документ и електронния подпис
ЗЕУ	Закон за електронното управление
КТЦЕП	Контролно-технически център на електронното правителство
АИС	Административна информационна система

РЕУ	Регистър на електронните услуги
РИО	Регистър на електронни документи
РОС	Регистър за оперативна съвместимост
СУНАУ	Списък на унифицираните наименования на административните услуги
ЗЕУ	Закон за електронното управление
ТЦЕП	Териториален център на електронното правителство
УРИ	Уникален регистров идентификатор

Програма за електронно правителство в т.ч. уеб портали, с основа и цел гражданите, бизнеса и обществото

Общата цел е изложена в стратегически документи като Стратегия за развитие на електронното управление в Република България 2014 – 2020 г. и пътна карта за изпълнение на Стратегия за развитие на електронното управление в Република България за периода 2014 – 2020 г., като се цели подобряване на ефективността на административното обслужване за гражданите и бизнеса чрез използване на възможностите на електронното управление.

Стратегия за развитие на електронното управление в Република България 2014 – 2020 г.

Визия

- Въведени ефективни бизнес модели в работата на администрацията – от рутинни дейности, към услуги за гражданите и бизнеса

- Изградена цифрова администрация – администрация, структурирана в съответствие със съвременните управленски технологии и постиженията на информационните и комуникационните технологии
- Постигната оперативна съвместимост на национално ниво – от фрагментирани и затворени, към интегрирани и технологично независими решения
- Изграден механизъм за координирано планиране и реализиране всички инициативи за развитие на електронното управление
- Осигурено предоставяне на административни услуги през единния портал на електронното управление по всяко време, от всяко място и чрез различни устройства

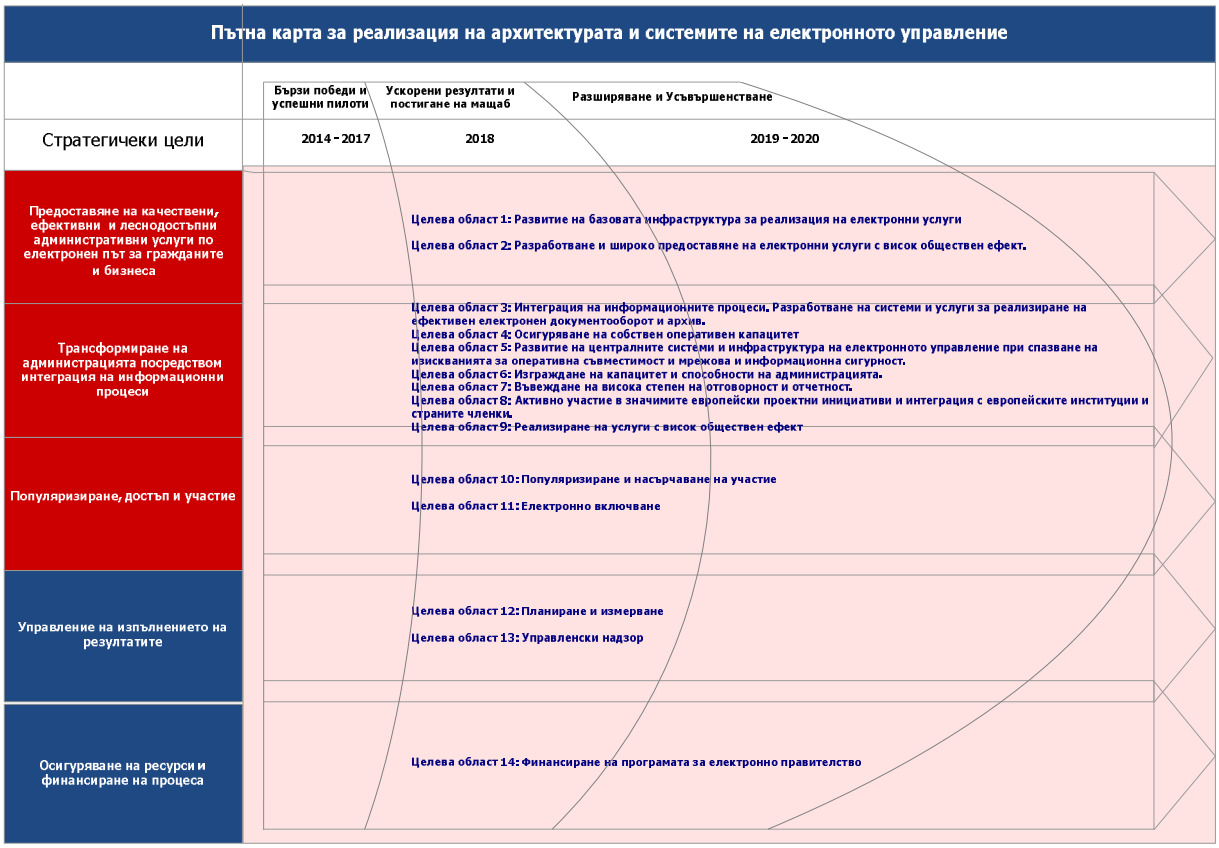
Модел на е-управление

За да се постигнат целите в Стратегията за електронно управление в Република България за периода 2014 – 2020 г. е направен преглед и анализ в следните ключови области:

- съществуващата на ниво ЕС и Република България нормативна уредба, регулираща развитието на електронното управление, респективно правителство
- съществуващите на ниво ЕС и Република България стратегически документи, които представляват стратегическата рамка за развитието на електронното управление

- текущо реализирана архитектура на електронното управление в Република България;
- проекти

Пътна карта



Пътна карта за електронното управление

Целеви области и ползи, посочени в пътната карта

- Предоставяне на качествени, ефективни и леснодостъпни електронни услуги за гражданите и бизнеса.;
- Разработване и широко предоставяне на електронни услуги с висок обществен ефект при гарантирано високо ниво на киберсигурност;
- Трансформиране на администрацията в цифрова администрация посредством интеграция на информационните процеси;
- Популяризиране, достъп и участие;
- Управление на изпълнението на резултатите;
- Осигуряване на ресурси и финансиране на процеса.

Специфични цели:

- Усъвършенстване на нормативната база с оглед предоставянето на административни услуги по електронен път
- Осигуряване на служебно начало при административното обслужване
- Усъвършенстване на процесите за предоставяне на електронни административни услуги
- Създаване на условия за интегрирано административно обслужване по електронен път
- Изграждане на капацитет за прилагането на политиката за електронното управление

Принципи на електронното управление:

- Приемственост – след критичен анализ всичко полезно се използва и надгражда
- Координация – преход от "работа на парче" към координация и изпълнение на комплекс от взаимосвързани проекти
- Измерими резултати – в края на всеки проект ще бъдат постигани заложените резултати с ефект върху големи групи от обществото – физически лица и бизнес
- Контрол

Електронна административна услуга (ЕАУ)

Електронните административни услуги се делят на следните видове:

Спрямо сложността на изпълнение се делят на:

- Първични - включват опростена административна услуга, засягаща най-често едно административно звено
- Комплексни - със сложна маршрутизация, включваща най-често множество административни единици.

Спрямо степента на реализация онлайн, електронните услуги се делят на :

- Еднопосочни
- Двупосочни
- Интерактивни
- Трансакционни
- Трансформационни

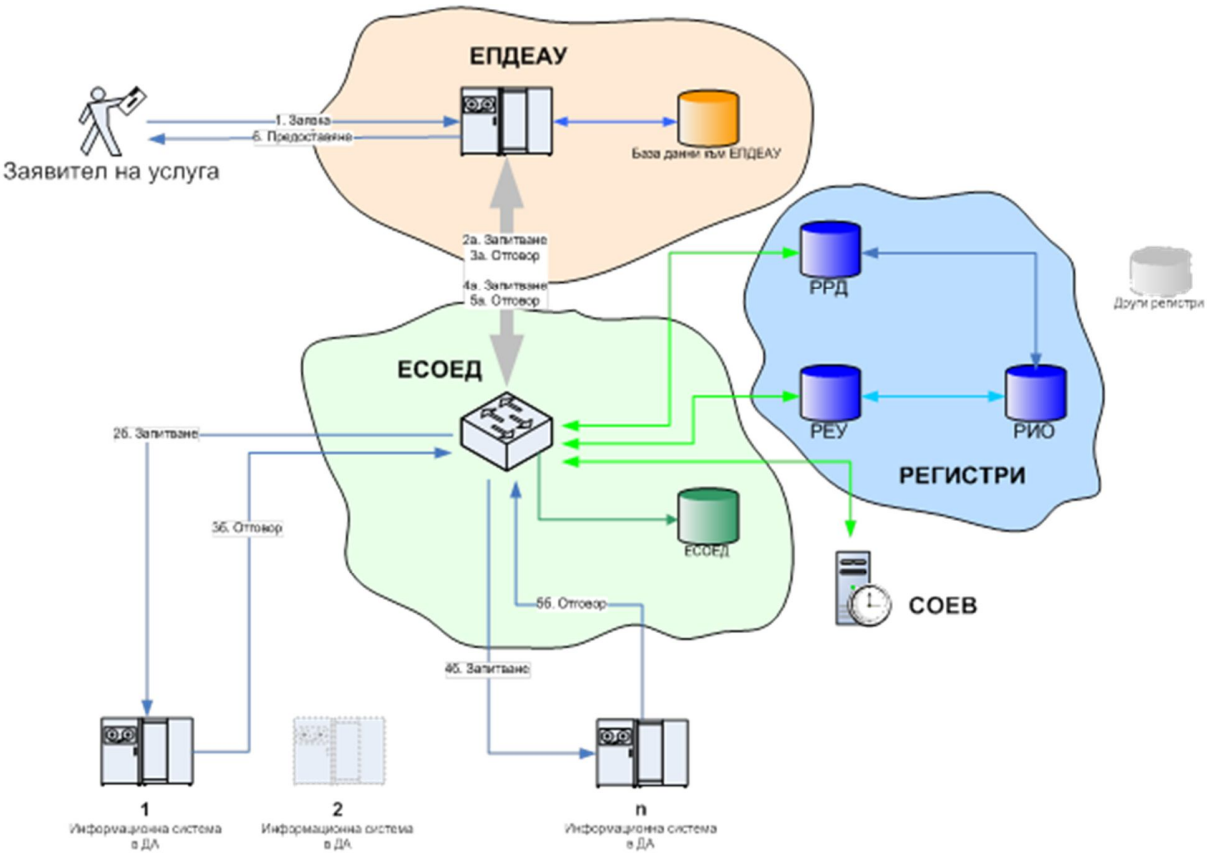
За реализация на електронна услуга се извършват следните стъпки:

- Правен и бизнес анализ на еУслугата
- Реализация/модернизация на АИС
- Дефиниране на информационните обекти на услугата
- Вписване на услугата в регистрите
- Интегриране на АИС с ЕСОЕД и вписване в ЕПДЕАУ

- Провеждане на функционални и интеграционни тестове
- Сертифициране и преминаване в експлоатация
- Поддръжка и актуализация

Примери от развитието на електронното правителство и уеб портали в България

ЕПДЕАУ (Единен портал за достъп до електронни административни услуги) www.egov.bg



Горната схема визуализира начина на работа на единния портал на електронното правителство:

- **Единен Портал за Достъп до Електронни Административни Услуги (ЕПДЕАУ)**

Входната точка на различните целеви групи на електронно правителство към информационната система за е-управление. Тук крайните потребители се идентифицират и отправят заявления за ЕАУ, посредством специализиран потребителски интерфейс. След приемане на заявленията за ЕАУ, ЕПДЕАУ препраща електронни съобщения към доставчика на конкретната ЕАУ чрез ЕСОЕД, която се грижи да се обърне към съответните АИС, които поддържат функционалност и данни, необходими за избраната услуга.

- **Единна среда за обмен на електронни документи (ЕСОЕД)**

Единната среда за обмен на електронни документи играе ролята на маршрутизатор, който препраща заявленията на крайните потребители до съответните АИС посредством електронни съобщения в XML формат.

- **Регистри за оперативна съвместимост (РОС)**

Към момента в интеграционната платформа на електронното правителство са реализирани следните регистри и списъци, определени в Закона за електронното управление и подзаконовите нормативни актове към него:

- **Регистър на регистрите и данните**

Целта на регистъра на регистрите и данните е да поддържа определенията на всички данни (унифицирани и неунифицирани), пакети от данни и информация за регистри и раздели от регистри, които се поддържат в администрациите.

- **Регистър на информационните обекти**

Целта на регистъра на информационните обекти е да поддържа формализираните технологични описания на информационните обекти, събирани, създавани, съхранявани и обработвани от административните органи в рамките на тяхната компетентност.

- **Регистър на електронните услуги**

Целта на регистъра на електронните услуги е да поддържа формализираните технологични описания на електронните административни услуги и на вътрешните електронни административни услуги, предоставяни чрез единната среда за обмен на електронни документи.

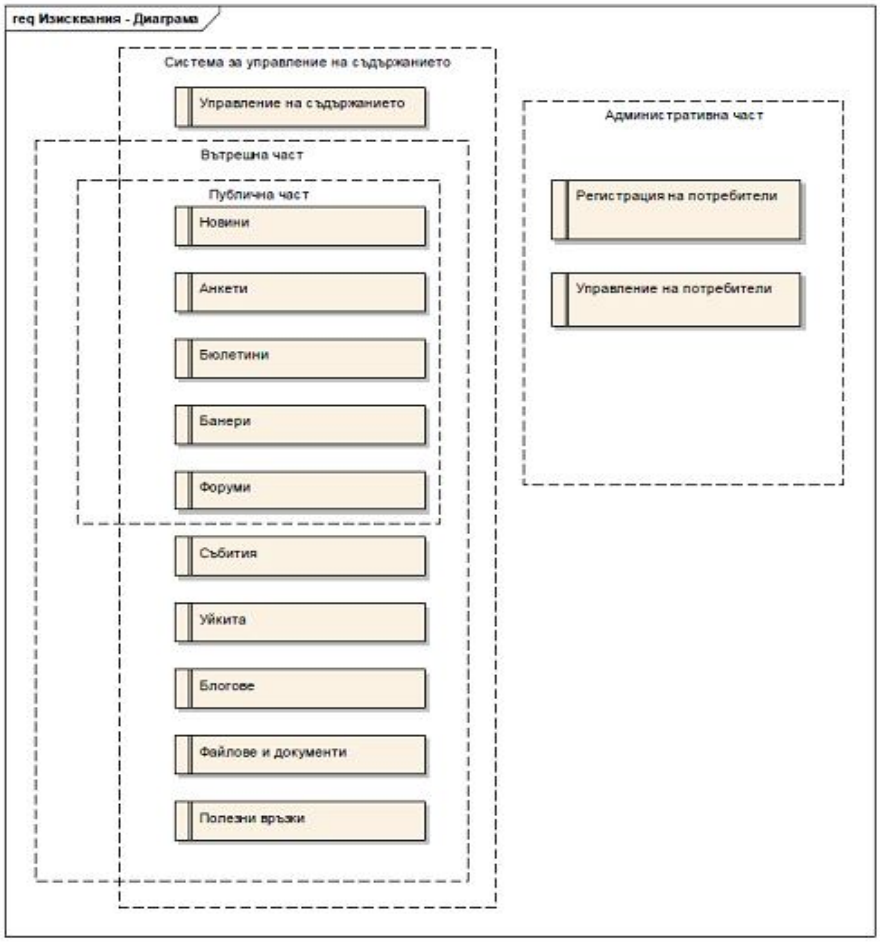


Схема на работата на системата за управление за съдържание на ЕПДЕАУ

Ползи от интегрирано електронно правителство чрез уеб портали

- Автоматизиран обмен на данни в администрацията
- Оптимизиране на администрацията
- „Едно гише“ или единен прозорец
- Предлагане на интегрирани и комплексни услуги
- Ползване на правителствен облак
- Повишаване на сигурността на информацията
- Въвеждане на нови технологии
- Мултиплицируемост на ЕАУ
- Иновативност

Модул 2: Мобилни и облачни технологии за уеб портали

Електронно и мобилно управление

- Дефиниция: сбор от правителствени услуги и приложения, които са достъпни само чрез употребата на клетъчни/мобилни телефони, лаптопи и безжична интернет инфраструктура
- Развитието на М-правителството следва четири основни измерения:
  - Трансформиране на ЕАУ за мобилни платформи
  - Достъп на мобилни адм. служители до информация
  - Интелигентен/гъвкав начин на работа
  - Предоставяне на услуги на гражданите и бизнеса навсякъде и по всяко време

Мобилни приложения в правителствения сектор

- Дефиниция: мобилното приложение е компютърна програма, предназначена да бъде изпълнявана на мобилни устройства като смартфони, лаптопи и др.
- Развитие на мобилните приложения: след първоначално развитие на база WAP протокол през 90-те години, достъпът до мобилни приложения се развива чрез установените след 2008-ма година платформи за разпространение на приложения, които типично се поддържат от собственика на мобилната операционна система (Apple App Store, Google Play, др.)
- Видове мобилни приложения: Мобилните приложения се разделят на три вида: Native (локални) – инсталират се на крайното устройство, работят под една операционна система; Уеб – уеб сайтове, които се изпълняват през браузър, най-често базирани на HTML5 стандарта, платформено независими и Хибридни приложения, които комбинират част от функциите на локалните приложения и уеб браузър, който е вграден в локалното приложение за обработка на HTML код.
- Използването на мобилни приложения в правителствения сектор се разглежда като допълнение на електронното управление. Съществуват редица ограничения по отношение на възможностите за предаване на информация (160 знака SMS сравнено с информацията/визуализацията на съобщение по ел. поща), нужна е налична ИКТ инфраструктура и разработени ЕАУ за оптимално използване, както и хетерогенността на крайните мобилни устройства на потребителите на

мобилни услуги създава предизвикателства пред използването на мобилни приложения.

Предимства на мобилните приложения: достъп до по-широки маси от население (отдалечени региони, възрастно население, възпрепятствани потребители); възможности за гъвкава комуникация; мобилност и достъп 24/7 до услуги през мобилни устройства; персонализация на услугите; пространствено базирани услуги; намаляване на разходи, положителен екологичен аспект.Big Data „ Големи данни“ – как да направим управлението по-бързо и по-умно

”Големите данни“ представляват процесът и неговият резултат при събиране на огромни количества разнородни електронни данни, които са толкова обемни и сложни, че е невъзможна ръчна или автоматизирана обработка с досегашните софтуерни и хардуерни инструменти.

В държавното управление се генерират множество различни по вид данни, което го прави идеална среда за приложение на големите данни.

Характеристики на „големите данни“:

Volume - като „големи данни“ се категоризират данни като тези, които ни биват предоставяни от социалните мрежи, информацията от логовете на сървърите ни, снимки от различни устройства за заснемане, сканирани документи и др.Съхранението и обработката на такова количество и размери данни в ерата на облачните изчисления при изгодни изчислителни ресурси се осъществява при много ниски разходи.

Velocity – описва скоростта на обработваните данни

Variety – данните, с които работи една организация, рядко са във формата, в която могат да бъдат използвани още в самото начало. Технологиите, стоящи зад “големите данни” могат да бъдат използвани, за да извлекат информация от тези първоначално неструктурирани данни и да ги предадат на съответното приложение в „изчистен“ и структуриран вид – било то за последващ анализ или просто като част от функционалността му.

Предизвикателства пред Big Data:

- липса на трансгранична координация
- неадекватна инфраструктура и възможности за финансиране
- недостиг на експерти в областта на данните и на свързаните с тях умения, нехомогенна и прекалено сложна правна среда.

Поява и еволюция на облачните технологии

- Историята на облачните технологии започва като концепция през 50-те години на 20-ти век с архитектура от големи изчислителни мащини (mainframe), където се извършват изчислителните операции, достъпвани през терминални клиенти, предлагащи само комуникационни операции с мейнфрейма. Настоящата концепция се базира на изпълнение на изчислителни или съхраняващи операции на множество далечни локации (облак), което позволява приложният софтуер да бъде достъпен през различни устройства през Интернет.

- Характеристики: гъвкавост на потребителя, поради липса на техн. инфраструктура; намаляване на разходите; независимост от крайното устройство и локацията; улеснена поддръжка; споделяне на ресурси и разходи, подобрена производителност, надеждност; възможност за надграждане; сигурност
- Модели за услуги: 1. Инфраструктура като услуга (IaaS) – трета страна хоства виртуализирани изчислителни ресурси, достъпни през Интернет. 2. Платформа като услуга (PaaS) – трета страна предлага освен базовата инфраструктура, така и надграждащи операционни системи и системни приложения, достъпни през Интернет. 3. Софтуер като услуга (SaaS) – модел за разпространение на услуги, където приложенията се хостват от производител или доставчик на услуги и са достъпни през Интернет.

Типове облаци

Хибриден облак (Hybrid Cloud)
Хибридният облачен модел съществува благодарение на необходимостта на организациите от различни видове облачни модели (Публични, Частни и Обществени) едновременно. Хибридният облачен модел позволява хостването на критично важни приложения в частния облак, докато приложенията с по-ниски изисквания към сигурността и достъпа се ползват в публичната част.
Обществен облак (Community Cloud)
Обществеият облак представлява инфраструктура, която се споделя от няколко организации, които формират общността, споделяйки близки интереси като сигурност, условия за ползване, изисквания за съвместимост и други подобни.Обществеият облак предлага по-висока степен на поверителност, сигурност и политика на съвместимост. Пример за такъв тип облак е проекта Gov Cloud на Google.Според някои анализатори, обществеият облак е поредния хит в света на технологиите. В България обществеият облак би бил добро решение за различни сегменти от електронното правителство.
Комбиниран облак (Combined Cloud)
Комбинирана облачна технология се формира при съединяването на няколко облака. Чрез интегриране на по-голям брой външни и вътрешни доставчици на услуги и приложения този тип облачен модел улеснява достъпа и ползването на услугите в публичния облак, като в същото време спестява определени трудости като например PCI съвместимост.
Вътрешен облак (Cloud of Clouds)
Вътрешен облак (или облак в облаците) представлява мрежа от облачни структури, която е изградена на базата на отворени стандарти и предоставя специфична среда на облачна технология. Тази концепция се изгражда на базата на желанието на няколко доставчици на облачни технологии да формират обща среда, която не съществува като самостоятелен облак.Подобна мрежа представлява интернет пространството, което може да се разглежда като мрежа от много мрежи. По подобен принцип глобален вътрешен облачен модел може да предоставя огромен брой услуги на потребителите което е извън възможностите или приоритетите на един отделен доставчик.

Облак от публични услуги

- Цел: Да се тества и демонстрира добавената стойност на архитектурите, ориентирани към услуги (SOA) и технологията "изчисления в облак" (cloud computing) за предоставянето на по-гъвкави и лесни за разработване и използване публични електронни услуги от страна на публичната администрация.
- Пилотните проекти трябва да тестват и валидират тези технологии и по-конкретно средствата за разработване на нови услуги чрез комбиниране на различни изграждащи блокове и да направят оценка на ползите за публичните администрации и за частните организации, използващи споделени услуги.

Характеристики и условия за пилотни проекти

- Да тестват и валидират иновативни услуги, основани на агрегирането на базови услуги в реални условия в период от 12 месеца.
- Да включват партньори по цялата верига на стойността, при това непременно публични администрации. Базовите услуги могат да се предоставят от членовете на консорциума или от други доставчици извън него.
- Да разработят пълен "бизнес кейс" за преминаване на публичните администрации към архитектура, ориентирана към услуги.
- Да обърнат специално внимание на сигурността и защитата на личните данни.
- Да включват реалистични количествени и качествени индикатори за мониторинг на изпълнението на проекта.

Ползи от правителствения облак

- Значително намаляване консумацията на енергия и екологичния ефект на обществените центрове за данни, след провеждането на обществени поръчки и набавянето на подходящи иновативни решения; подобно развитие ще наложи нови норми и практики за устойчивост на центровете за данни в Европа.
- Значителна подкрепа за развитие и прилагане на технологиите за виртуализация и изчисления в облак с оглед пестене не енергия в центровете за данни в Европа.
- Насърчаване на приемането на стандарти/метрика за измерване на енергийния/екологичен ефект от ИКТ сектора (тъй като центровете за данни формират основата на почти всяка ИКТ инфраструктура).
- Комбинирано с други действия, това ще увеличи конкурентоспособността на европейската индустрията в областта на „зелените" ИКТ.
- По-ефективни и ефикасни административни услуги.
- Стимулиране предоставянето на нови услуги от страна на частния сектор, които са базирани на публична информация и услуги.
- Демонстриране на възможностите за използване на нови архитектури при много различни правни среди и на ползите от тях.
- Демонстриране в дългосрочен план на намаляването на разходите и административното натоварване при предоставяне на електронни услуги.

Модул 3: Въведение в HTML

- **Основна концепция на уеб**
- **HTTP заявки**
- **Структура на HTML**
- **Тагове**
- **Атрибути**
- **Метаданни**

Основна концепция на уеб

Най-общо уеб, представлява система от взаимно свързани хипертекстови документи, достъпни през компютърната мрежа "интернет". Тези документи обикновено са в HTML или XHTML формат и могат да съдържат разнообразна по тип информация(текст, изображения, мултимедийни компоненти и видео). Връзката между тях се осъществява чрез хипервръзки. Представянето им пред потребителите става с помощта на допълнителни програми наречени браузъри (Internet Explorer, Chrome, Firefox, Safari, Opera и др).

HTTP заявки

Какво е HTTP - съкратено от The Hypertext Transfer Protocol (протокол за трансфер на хипертекст) е създаден с цел осъществяването на комуникацията между клиент(браузър) и сървър(сървър на който е качен уеб сайта). Комуникацията се осъществява посредством HTTP заявки (request Message), от страна на клиента, и отговор(Response Message) от страна на сървъра. Съществуват два метода за HTTP заявки към сървъра – GET и POST. За повече информация http://www.w3schools.com/tags/ref_httpmethods.asp.

Структура на HTML документа

HTML документа – съкратено от **H**yper **T**ext **M**arkup **L**anguage (език за маркиране на хипертекст) се състои от съвкупност от елементи, които от своя страна са съставени от тагове – HTML tags (специални думи които имат определено значение за браузъра при показването на документа. Браузърите не показват HTML таговете, а ги използват, за да интерпретират съдържанието на страницата), подредени по определени правила. Всеки

такъв документ започва с декларация която дефинира неговия тип. За HTML5 тя изглежда по следния начин:

```
<!DOCTYPE html>
```

Тя е последвана от <html> таг. В него се съдържа цялата информация в документа. Тази информация се разделя на две части <head> и <body>.

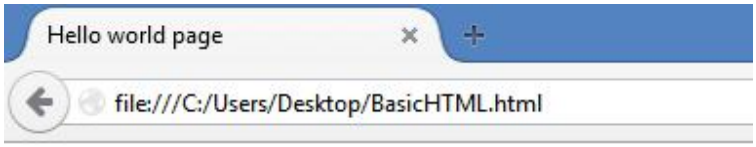
- <head> частта предоставя информация за документа. Там най често се поставят стиловете и скриптовете към сайта.

```
<head>
        <title>Page Title</title>
       <link  rel="stylesheet" href="Style/style.css">
</head>
```

Тази част не се визуализира на екрана, служи за модификация на видимата част от документа. В <head> частта може да се съдържа и <title> тага който от своя страна оказва заглавието на документа(вижда се в таб полето при отваряне на страницата).

```
<title>Hello world page </title>
```

Ето как изглежда под Mozilla:



- <body> частта представлява информацията която се съдържа в страницата. Това може да бъде текст, видео, аудио, картинки и др.

```
<!DOCTYPE HTML>

<html>
    <head>
            <title>Your Website</title>
            <link  rel="stylesheet" href="Style/style.css">
    </head>
```

```
<body>
        <header>
                <h1>Title</h1>
                <nav>
                        <ul>
                                <li>Your menu</li>
                                <li>Your menu</li>
                        </ul>
                </nav>
        </header>

        <section>
                <article class="box-shadow">
                        <header>
                                <h2>first of type / last of type</h2>
                        </header>
                        <p>Pellentesque habitant morbi tristique senectus et netus et
malesuada fames ac turpis egestas.</p>
                </article>
        </section>

        <aside>
                <h2>About section</h2>
                <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies mi
vitae est. Mauris placerat eleifend leo.</p>
        </aside>

        <footer>
                <p>Copyright 2009 Your name</p>
        </footer>
</body>
</html>
```

Това е пример за HTML документ съдържащ всички до момента описани елементи, които ще бъдат визуализирани.

Тагове

Елементите са градивната частица на HTML документа. Всеки елементи се състоят от тагове. Всеки таг има определено значение. Повече за разликите между отделните тагове ще разгледаме по-нататък в книгата.

- Най-често таговете се състоят от две части – отварящ таг и затварящ таг:

<head> - отварящ таг с име “head”

</head> - затварящ таг

- Има и тагове които са самозатварящи се(състоят се от една част която се затваря в края):

Между отварящия и затварящия таг маже да стоят както други тагове така и текст. Получава се така нареченото влагане(един таг се съдържа в друг).

Атрибути

Атрибутите са част от таговете. Те предоставят допълнителна информация за семантиката на съдържанието на елемента. Винаги се дефинират в отварящия таг. Синтаксисът е следния:

attributeName=”attributeValue”

Пример:

 Google

Атрибутите не са общи за всички елементи. Някой елементи имат специфични за тях атрибути.

Модул 4: Форматиране с HTML

- **Форматиране на уеб страници**
- **Списъци**
- **Таблици**
- **Хипервръзки**
- **Изображения**

- **Управление на текстовия поток**

Има доста голям списък от символи които не можем да напишем чрез клавиатурата или са служебни за HTML. За решаването на този проблем съществуват така наречените резервирани символи.

Форматиране на уеб страници

В HTML има елементи със специално значение оказващи влияние на визуализирането на даден текст от документа. Такива са всички от групата "h" (h1,h2,h3,h4,h5,h6), оказващи началото на заглавие, "em", оказващ че текста ще е наклонен, "strong", оказващ удебелен текст и тн.

Списъци

За визуализацията на списъци в HTML документа се използват два основни елемента – "ul" и "ol", които съответстват на unordered list и ordered list. Съществува и трети по-рядко използван вариант – "dl", който представлява списък с дефиниции – списък от термини с описание за всеки термин.

Пример за unordered list (неподреден списък):

```
<ul>
        <li>Your menu</li>
        <li>Your menu</li>
        <li>Your menu</li>
</ul>
```

→

- Your menu
- Your menu
- Your menu

Пример за ordered list (подреден списък):

```
<ol>
        <li>Your menu</li>
        <li>Your menu</li>
        <li>Your menu</li>
```

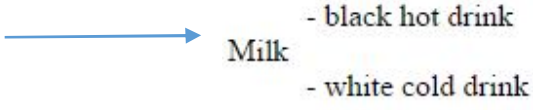
→

1. Your menu
2. Your menu
3. Your menu

```
</ol>
```

Пример за списък с дефиниции:

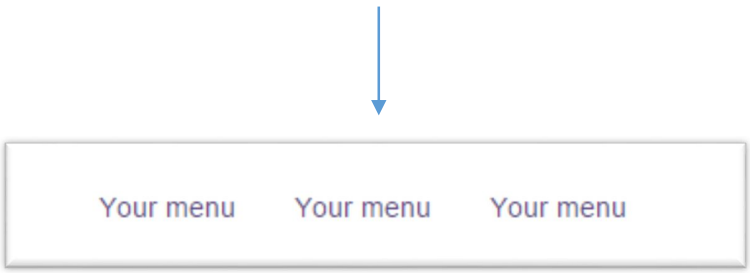
```
<dl>
        <dt>Coffee</dt>
        <dd>- black hot drink</dd>
        <dt>Milk</dt>
        <dd>- white cold drink</dd>
</dl>
```



Списъците се използват за изброяване на множество елементи, но често се употребяват за създаването на навигацията на страницата. С добавянето на малко стилове, от обикновен списък се получава красива навигация. В случая чрез стилизирането на списъка от (1) се получава (2)

(1)
```
<ul>
        <li>Your menu</li>
        <li>Your menu</li>
        <li>Your menu</li>
</ul>
```

(2)



Таблици

Както говори името им, таблиците се използват за таблично оформление и за нищо друго. Не е добра практика да използваме таблици за подреждане на страницата, а това е често

срещана грешка в остарели страници. В новите стандарти съществуват правила за това, които ще разгледаме по нататък.

Таблицата се създава члез отварящ и затварящ “table” таг.

```
<table> </table>
```

Това обаче не е достатъчно. Създавайки таблица ние очакваме тя да има редове и колони.

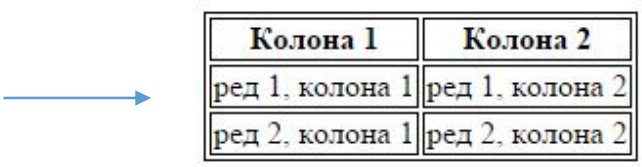
<tr> - дефинира редовете на таблицата

<td> - дефинира клетките на таблицата

<th> - дефинира заглавия на колони

В HTML, тага оказващ клетка се вмъква в тага оказващ ред:

```
<table border="1">
    <tr>
        <th>Колона 1</th>
        <th>Колона 2</th>
    </tr>
    <tr>
        <td>ред 1, колона 1</td>
        <td>ред 1, колона 2</td>
    </tr>
    <tr>
        <td>ред 2, колона 1</td>
        <td>ред 2, колона 2</td>
    </tr>
</table>
```



Колона 1	Колона 2
ред 1, колона 1	ред 1, колона 2
ред 2, колона 1	ред 2, колона 2

Съществуват тагове оказващи смислово разделение на таблицата – thead, tbody и tfoot.

- Thead – използва се за заглавната част на таблицата. В този елемент се поставя “th” мвесто “td”. Позиционира се в най горната част на таблицата.

- Tbody – тялото на таблицата. Позиционира се в най-долната част на таблицата в HTML документа, но се показва веднага след “thead” на страницата.
- Tfoot – позиционира се между “thead” и “tbody”.

Хипервръзки

Най-често се използват за връзка от една страница към друга, но има и възможност за достигане на различни части на страницата посредством хипервръзки. Наричат се още линкове. Започват с отварящ таг <a>(anchor tag) и завършват с затварящия такъв . Притежава един основен атрибут – “href”, оказващ адреса към който сочи линка.

```
<a href=”http://www.google.com” > Google </a>
```

След натискане на мишката върху линка, ще бъдем пренасочени към сайта на google. В случая адреса е абсолютен, което означава че ни води към съществуваща външна за нашия сайт страница. Съществува и друг вид адресация – чрез ралативни адреси. В долния пример “href” атрибута има различна стойност. Не сочи към уеб адрес, а към части от страницата или други документи от проекта.

```
<a href=”/test” > Google </a>          ——→  (1)
<a href=”/page.html” > Google </a>     ——→  (2)
<a href=”Folder1/page.html” > Google </a>  ——→  (3)
<a href=”../index.html” > Google </a>  ——→  (4)
```

(1) – насочва ни към папка която се намира в текущата директория.

(2) – насочва ни към HTML документ (page.html) в текущата директория.

(3) – насочва ни към HTML документ (page.html), намиращ се в папка Folder1 в текущата директория.

(4) – насочва ни към HTML документ (index.html), намиращ се в папка, съдържаща в себе си текущата директория.

```
<a href=”#someIDName” > Go down </a>
```

 Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has…

Тук линка ще сочи към “div” елемента независимо къде се намира в текущата страница и ще скролира страницата до началото на този елемент.

Когато използваме линкове, можем да им зададем допълнително поведение. При клик сочената от линка страница може да се отвори в нов таб, различен прозорец или в текущия таб. Повече информация за това - http://www.w3schools.com/tags/att_a_target.asp .

Изображения

За визуализирането на изображение се използва “img” елемента. Той има атрибут “src”, който оказва адреса на който се намира изображението. Адресът може да бъде абсолютен и релативен.

Релативен:

```
<img src="logo.jpg" alt="My Web site logo" height="100" width="100" />
```

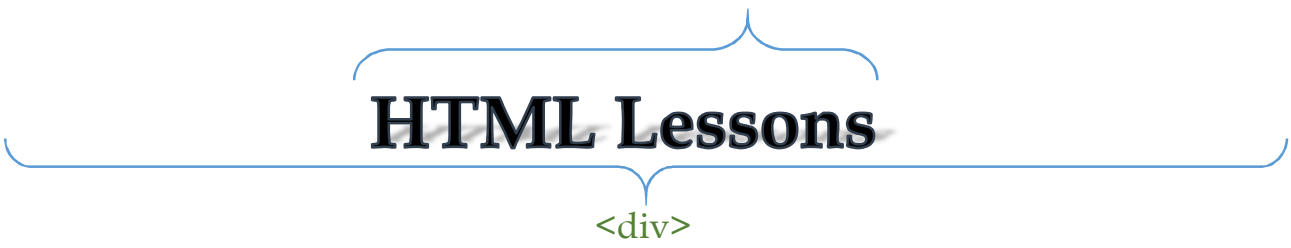
Абсолютен:

```
<img src="htttp://someAddress/someImage.jpg" alt="My Web site logo" height="100" width="100" />
```

Хубаво е винаги да задаваме размери на изображението, независимо от формата – пиксели, проценти и др.

Центриране на изображения – изображението може да се центрира спрямо широчината на своя обграждащ контейнер (родител). Това става като на родителя се зададе стил “text-align:center”. Това ще постави картинката на равни разстояния отляво и отдясно спрямо широчината на родителя и.

```
<div                                                    align="center">
      <img                          src="Example.gif"                    />
</div>
```



По сходен начин с атрибута “text-align” можем да подравним изображението наляво и надясно

```
<div align="left">
        <img src="Example.gif" />
</div>
```


```
<div align="right">
        <img src="Example.gif" />
</div>
```

Управление на текстовия поток

“div” елемент – един от основните в HTML. Блоков елемент (заема цял ред от страницата). Често се ползва за контейнер, обграждащ основното съдържание или отделните части от страницата.

```
<div id=“header”> This is header </div>
```

Това е пример за “div” който обгражда “header” част.

Много е важно да сме наясно от какъв тип е всеки елемент който използваме – “block” или “inline”. Както споменахме, блоковите елементи заемат цялото възможно място на един ред от страницата, докато “inline” елементите заемат само толкова колко им е необходимо за да се визуализират. Затова можем да наредим няколко “inline” елемента един до друг. С правилната комбинация от двата типа елементи можем да наредим до голяма степен нашата страница.

Типичен пример за “inline” елементи са: “span”, “img”, “a”, “input” и др.

Типичен пример за “block” елементи са: “div”, “h1”, “ul”, “p” и др.

Текстът може да се съдържа в доста елементи, но най използваните са: “p”, ”span”, ”h1”, ”a” и др. Можем да му задаваме цвят, шрифт, размер, подравняване, дебелина, стил и др.

Модул 5: Форми и структура на HTML

- **HTML форми**
- **Събиране на потребителски вход**
- **Валидация**
- **Структурни елементи**
- **Навигация и менюта**

HTML форми

Използват се за събиране на потребителска информация. Създават се с “form” елемента:

```
<form> ..form elements.. </form>
```

Събиране на потребителски вход

Формите имат няколко атрибута с определени функции.

“action” атрибута – оказва къде ще бъдат изпратени данните.

“method” атрибута – уточнява как ще бъдат изпратени данните.

Само с “form” елемента нищо не се визуализира. За целта трябва да добавим някой други елементи – “input”, “textarea”, “select” и др. Всеки от тях има определена визия и функция, като най-функционален е input” елемента.

Всеки “input” елемент притежава атрибут “type”, който оказва неговият тип. Най-използваните са следните:

- Бутон

```
<input type=”button” value=”Click me” >
```

- Checkbox - позволява на потребителя да избира между една или повече опции и да оказва своя избор посредством клик върху малка кутийка. В тази кутийка ще се появи отметка. Атрибутат “value” оказва стойността която приема “input” елемента, но тя не се визуализира. За улеснение на потребителя добавяме текст, който ще се визуализира непосредствено след “input” елемента. (
 елемента оказва начало на нов ред – използваме го за по-добра визуализация, двата “input” елемента са на различен ред). Атрибута “checked” оказва че елемента ще се визуализира като селектиран:

```
<input type=”checkbox”checked value=”Apple” > Apple <br>
<input type=”checkbox” value=”Lemon” > Lemon <br>
```

- Text – представлява ткстово поле, предоставящо на потребителя възможност за въвеждане на текст на един ред.

```
<input type=”text”>
```

- Radio – подобно на “checkbox”, позволява на потребителя да избира между няколко възможности, но тук той може да избере само една. Чрез атрибута “name” оказваме кои “input” елементи се отнасят за една и съща група – необходимо е стойността му да е една и съща.

```
<input type=”radio” value=”male” checked name=”gender”> Male <br>
<input type=”radio” value=”female” name=”gender”> Female <br>
```

- Password – текстово поле, чийто символи са скрити. Използва се за пароли при вход (log in) и регистрация в даден сайт.

<input type=”password” value=”password” > → ••••••••

- File – използва се за прикачване на файлове в сайта. За целта се клика върху бутона “Browse”, отваря се прозорец, представляващ папка от нашия компютър. Остава да изберем желания от нас файл и да потвърдим качването му посредством бутона “Open”.

<input type=”file” value=”password” > → Choose File No file chosen

повече информация за “input” елемента - http://www.w3schools.com/tags/att_input_type.asp .

- Textarea – представлява отделен елемент подобен на “input” type=”text”, с тази разлика че потребителя може да въвежда текст на няколко реда<textarea>some text. </textarea> → some text.

- Select – създава падащ списък, предоставящ избор на потребителя подобен на “input” type=“radio”. “Select” елемента пази в себе си опциите от които трябва да избира потребителя. Тези опции се дефинират чрез елемента <option> с атрибут “value” оказваща стойността на тази опция, и текст подсказващ на потребителя за стойността на опцията.

```
<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
</select>
```

→ Volvo ▾ (1) → Volvo ▾ / Volvo / Saab (2)

(1) – представлява списъкът който се визуализира при зареждане на страницата.

(2) – начина по който се визуализира списъкът след клик върху него.

Валидация

Свързана е с проверка дали въведената информация е в правилния формат и е използваема преди изпращането на данните към сървъра. Потребителският вход може да варира в точност, качество и намерение. Client-side валидацията подобрява потребителското усещане. Server-side валидацията е все още наложителна.

Някои проверки по време на валидация:

- Задължителните полета са попълнени
- E-mail адресите са валидни
- Датите са коректни
- Не се среща текст в числово поле

Автоматичната валидация на входните данни означава, че браузъра проверява данните на потребителския вход. Нарича се **client-side validation** (валидация при клиента/потребителя).

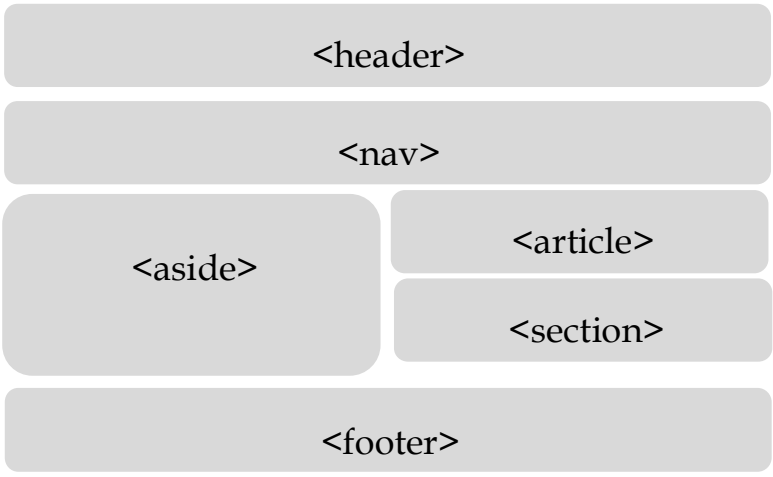
Server-side validation(валидация при сървъра) имаме, когато сървъра валидира данните, които са получени от потребителския вход на форма.

Повече за валидацията на форми може да прочетете http://www.the-art-of-web.com/html/html5-form-validation/ .

Структурни елементи

Структурните елементи се използват в налагащия се вече HTML5 стандарт.

Представляват съвкупност от елементи които задават семантика на страницата (разделят страницата на смислени части). Основните и най-често използвани са “header”, “section”, “article”, “aside”, “footer”, “nav”. Останалите може да видитена http://www.w3schools.com/html/html5_new_elements.asp. Тъй като са нови, не се подържат от някой стари браузъри.



Това е примерна структура на една HTML5 страница.

Навигация и менюта

Съвременния подход за създаване на навигация в страница е свързан с използването на семантичния елемент “nav”. Той смислово оказва че всичко намиращо се между отварящия и затварящия таг е част от навигация. Пример за използването му в страница:

```
<nav>
        <ul>
                <li> first list</li>
                <li> second list</li>
                <li> third list</li>
        </ul>
</nav>
```

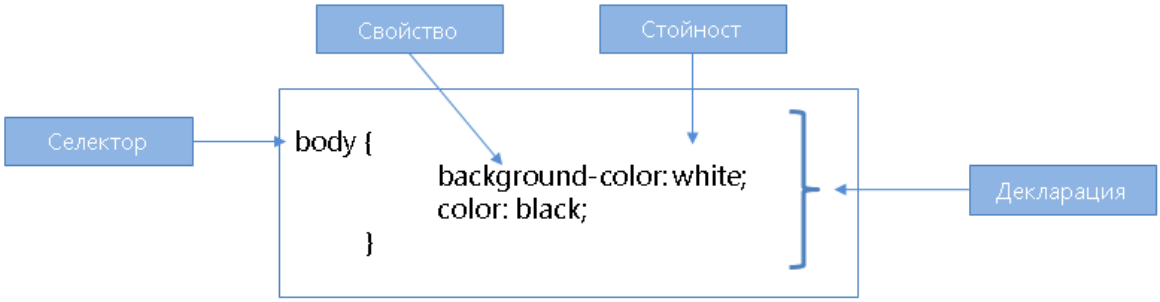
Модул 6: Контрол на цвета и типографията

- **Въведение в CSS**
- **CSS селектори**
- **Методи за прилагане на стилове**
- **Позициониране на блокови елементи**
- **Block layout модел**
- **Цветове и градиенти**

Въведение в CSS

CSS - Cascading Style Sheets (каскадни листа със стилове) е език за описване на стилове. Използва се предимно върху HTML документи, но може да се приложи и върху XML документи - eXtensible Markup Language (разширяем маркиращ език). Създаден е с цел разделение между структурата на документа и неговото визуално представяне. Поддържа се от всички браузъри. Най-новият стандарт е CSS3 (най-новата версия на CSS). Подържа се само от нови браузъри и при това частично.

CSS свойства



Когато се говори за оформление на съдържание чрез CSS трябва да се вземат предвид следните основни термини:

Селектор – това е израза, който чрез определени синтактични правила указва върху кои елементи ще се приложи стиловото оформление

Свойство – указва коя от характеристиките на указаните чрез селектора елементи трябва да се промени

Стойност – задава новата стойност на характеристиката, която се променя

Декларация – Съвкупността от селектора, всичките свойства които трябва да бъдат променени и техните нови стойности

CSS селектори

За да зададем даден стил върху елемент, трябва изрично преди това да го посочим. Това става чрез името на тага, “class” или “id” или комбинация между трите. Израза чрез който оказваме елемента върху който искаме да приложим определен стил се нарича селектор.

Примерни селектори:

- div.className - оказва “div” елемент с клас “className”
- p span – търсения елемент е “span” съдържащ се в “p”

- p span#idName.className – тук отново имаме “span” съдържащ се в “p”,

но трябва да има class=”className” и id=”idName”.

След като сме селектирали конкретен елемент, следва да зададен стилове. Всички стилове се описват между къдрави скоби - { some styles }. Всеки стил започва с ключова дума (какво искаме да променим стилово) последвана от двуеточие и стойност за стила.

Пример:

```
div.className  {
        width: 150px;
}
```

Този стил задава ширина 150 пиксела на “div” елемент с class=”className”. С цел по-добра четимост е хубаво да оставяме коментари в нашия код, както за HTML така и за CSS. В CSS коментарите за ограждат в “/* some comment*/”. Коментарите не се четат от браузърите и не влияят на документа.

Тежест на селекторите:

Всеки селектор има определена тежест. Ако имаме различни стилове за един и същи елемент ще се изпълни този с по-голяма тежест. Ако са с еднаква тежест ще се изпълни този който е дефиниран последен. Нека “a” е броя на селекторите с “ID”, “b” е броя на селекторите с “class” и “c” е броя на селекторите с име на таг.

```
*               /* a=0 b=0 c=0 -> specificity =   0 */
LI              /* a=0 b=0 c=1 -> specificity =   1 */
UL LI           /* a=0 b=0 c=2 -> specificity =   2 */
LI.red          /* a=0 b=1 c=1 -> specificity =  11 */
#content        /* a=1 b=0 c=0 -> specificity = 100 */
```

Това е начинът по който лесно можем да пресмятаме тежестта на нашите селектори.

Не е добра практика селекторите ни да са дълги:

div.mainContainer section.firstSection div.someDiv p span.lastSpan

Начини за прилагане на стилове

Има 4 метода за прилагане на стилове върху документ.

- Чрез атрибута **style** за конкретен HTML element:

```
<p style="color:blue;"> some text </p>
```

- Чрез елемента **<style>** в **<head>** секцията за прилагане на стилове на конкретната страница

```
<style type="text/css">
    p { color: blue; }
</style>
```

- Елемент **<link>** за връзка с външен CSS файл:

```
<link rel="stylesheet" type="text/css" href="mystyles.css" media="screen">
```

- Чрез javascript

Основния и препоръчителен метод е използването на външен файл където да са описани всички файлове.

Дължини и мерни единици

Когато се задават стойности за големина и оразмеряване, в CSS могат да се използват различни мерни единици. Те се делят на относителни и абсолютни.

Относелните мерни единици задават големината спрямо големината на друг елемент, който се стилизира.

Абсолютните мерни единици задават точна големина.

Когато се задава стойност на характеристика за размер трябва задължително да се зададат 2 компонента – числова стойност и мерна единица. Изключение се допуска само когато числовата стойност е 0.

Видове мерни единици:

Px – стойността означава брой пиксели – големината на изображението на екрана ще зависи от разделителната способност

Pt – точки. Големината е 1/72 от инч. Често се използва при печат.

% - процент от големината на променяната характеристика, на елемента, който съдържа променяният елемент

Em – коефициент за големина на шрифтове. Текущата стойност на шрифта винаги е 1em. Задавйки 2em шрифта ства два пъти по-голям.

Vh – Viewport Height – 1% от височината на видимото поле на браузъра. 50vh се равнява на половината от височината на прозореца.

Vw – Viewport Width – 1% от вирочината на видимото поле на браузъра.

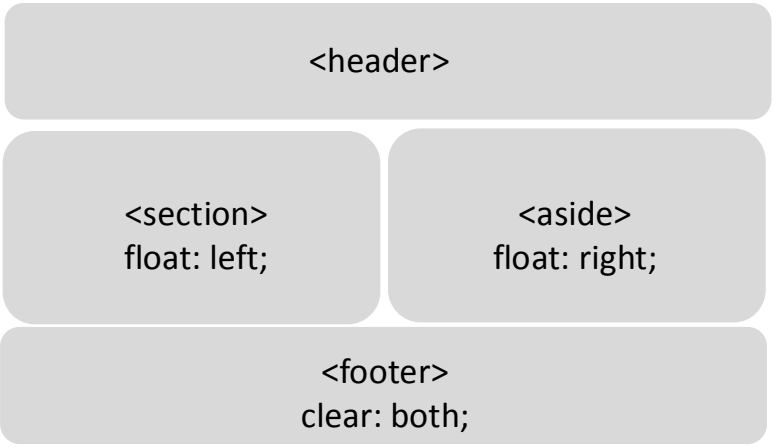
За повече информация за единиците за оразмеряване:
http://www.w3schools.com/cssref/css_units.asp

Позициониране на блокови елементи

- Позициониране чрез “float”:

“float” е CSS свойство което определя начина на позициониране на даден елемент. Възможните му стойности са “left” и ”right”. Това свойство може да се прилага както върху inline елементи така и върху “block” елементи. При задаване на “float” върху “block” елемент, той подобно на “inline” елементите, той заема единствено мястото от реда, необходима за неговата визуализация в лявата или дясната част на реда, в зависимост от стойността на свойството. Важно е след използване на “float” позициониране да се направи така нареченото “зачистване”. На родителя на елемента който е позициониран, се прилага свойството “clear” със стойност “both”. Алтернативния вариант е да поставим празен “div” елемент след позиционирания и да му зададем същото свойство.

```
section {
        float: left;
        margin: 10px;
        width: 600px;
}
aside {
        float: right;
        margin: 10px;
        width: 320px;
}
```



- Позициониране чрез свойството “position” – ще го разгледаме по подробно в следващи раздели.

Block layout модел

CSS3 добавя поддръжка на някои нови методи за подредба на съдържанието.

- Разделение на колони:

```
column-count: 3;
column-gap: 5rem;
column-rule: 1px solid black;
column-width: 100px;
```

- Flexbox

```
display: flex;
lex: 1 | auto;
```

Цветове и градиенти

- Цветове – на повечето елементи в HTML може да се задава цвят, посредством свойствата “color” или ”background-color”. Color – използва се за задаване цвета на текст в даден елемент. Background-color – определя цвета на фона на даден елемент.

Стойностите на тези свойства са цветове записани по определени правила. Най-използваните са: RGB/RGBA - Red, Green, Blue (Червено, Зелено, Синьо) / (Red, Green, Blue, Alpha – Червено, Зелено, Синьо, Прозрачност); шестнайсетичен запис(#000000); HSL/HSLA – Hue, Saturation, Lightness (оттенък, насищане, осветеност)/ Hue, Saturation, Lightness, Alpha (оттенък, насищане, осветеност, прозрачност)

- RGB/RGBA - първото число показва стойността на червиния цвят, второто на зеления, третото на синия и ако има четвърта стойност – прозрачност. Стойностите на цветовете са от 0 до 255. Колкото по висока е стойноста толкова по наситен е цвета.

color: rgb(255,0,0); - червен звят → 

color: rgba(2550,0,0.5); - червен цвят с 50% наситеност → 

R – red

G – green

B – blue

A – alpha (прозрачност)

- Шестнайсетичен запис

color: #ff6600; - оранжев цвят → 

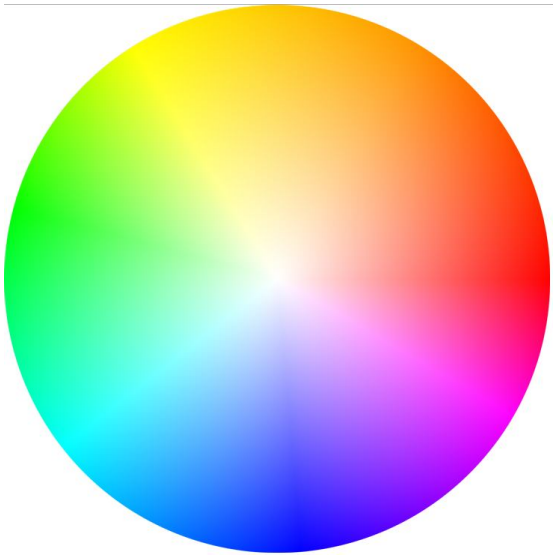
FF – подобно на rgb оказва колко наситено е червеното

66 – подобно на rgb оказва колко наситено е зеленото

00 – подобно на rgb оказва колко наситено е синьото

- HSL/HSLA - Hue (оттенък) представлява градус от 0 до 360 в колелото на цветовете. 0 или 360 са червен цвят, 120 – зелен, 240 – син. Saturation представляна ниситеността на цвета – от 0 до 100%. Lightness – осветеност, задава се от 0 до 100%.

Колело на цветовете →



color: hsl(240, 100%, 50%); - тъмно синьо →

color: hsl(120, 100%, 50%, 0.5); - зелено с прозрачност →

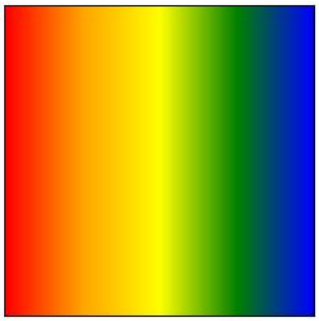
Повече информация за шестнайсетичния запис http://www.w3schools.com/html/html_colors.asp, а за всички http://www.w3schools.com/cssref/css_colors_legal.asp. Тези линкове да се поставят в пълнота

Цветовете може да се използват и с предварително предефинирани имена – red, green, yellow, blue, pink …

color:red - → some text

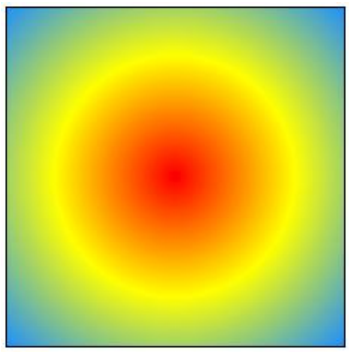
- Градиенти – представляват преливане на цветове от един в друг в определен ред и посока. Биват два вида “linear” и “radial”
    - Linear:

```
background: linear-gradient(direction, start-color, [mid-color-list,] end-color);
background: radial-gradient(top right, ellipse, red, blue);
background: -prefix-linear-gradient(left, red, orange, yellow, green, blue);
background: -linear-gradient(to right, red, orange, yellow, green, blue);
background: -linear-gradient(to right, red, orange 50%, yellow, green, blue);
```



    - Radial:

```
background: radial-gradient(red, yellow, rgb(30, 144, 255));
```



Промяна на стила на шрифта

Както съдържанието има основна роля за интернет, така типографията е най-важната част при изграждането на дизайна на сайта. В CSS се обръща голямо внимание на работата с текст.

Шрифтове:

Най-важната част от оформлението на текста е шрифта, с който той да бъде изписан. Ето някои основни свойства, които се прилагат при оформяне на шрифтовете:

-font-family – използва се за да се даде име на шрифта, който трябва да бъде приложен. Задава се списък от имена, като ако не бъде намерен първият, се прилага вторият и т.н. Ако никой от зададените шрифтове не бъде намерен браузъра избира кой шрифт да бъде зададен.

font-family : Arial, Candara, Verdana, sans-serif; → Some text

font-family : Georgia, Corbel, "Times New Roman", serif; → Some text

font-family : Consolas, "Courier New", monospaced; → Some text

-font-size – задава размера на шрифта чрез височината на реда

font-size : 16px;

-font-style – може да зададе дали шрифта да е italic или oblique (наклонен текст)

font-style: italic; → *Some text*

font-style: oblique; → *Some text*

-font-weight – задава удебеление на текста – стойностите може да са както думи, така и числата 100, 200, 300, 400, 500, 600, 700, 800, 900

font-weight : bold; → **Some text**

font-weight : normal; → Some text

font-weight : 800; → **Some text**

Тези четри свойства могат да бъдат обединени в едно – font:

font: italic bold 12px/30px Georgia, serif ; → ***Some text***

Типографски свойства:

-letter-spacing – задава разстояние между отделните букви

letter-spacing : 1.2em; → S o m e t e x t

letter-spacing : -3px; → Sometext

-line-height – увеличава или намалява празното разстояние между редовете:

line-height : 16px;

line-height : normal; /* Стойност по подразбиране */

line-height : 1.2;

line-height : 120%;

-text-align – задава в коя част от екрана да бъде подравнен текста:

text-align : left;

-text-decoration – служи за подчертаване на текста – отдолу, отгоре или в средата

text-decoration : underline; → Some text

-text-transform – чрез него може да се укаже текста да бъде визуализиран само с главни или само с малки букви:

text-transform : lowercase;

Създаване на свързан style sheet

Стиловете, които се прилагат върху съдържанието на html документ могат да бъдат разделени на два вида

а) вътрешен – дефинира се и се използва само в текущия документ (когато се налага да се форматира само една страница).

б) външен – текстов файл, съдържащ форматиращи атрибути (само CSS код). Към него трябва да има препратка от всички страници, които трябва да имат този външен вид. Използва се, когато искате сайтът да има еднотипен външен вид, независимо коя страница е отворена.

Предимства на външните CSS

а) стиловете могат да бъдат използвани в много документи;

б) тъй като форматиращият код се намира в един общ документ, то:

- не се налага ползващите го страници постоянно и поотделно да го зареждат поради кеширането на този файл;

- използва се по-малко код и страниците се зареждат по-бързо;

- добавянето на съдържание към уеб сайтовете е много по-бързо и лесно;

- по-опростено актуализиране и поддържане на сайтовете.

Модул 7: CSS селектори

- **CSS селектори**
- **HTML наследяване**
- **Групиране на селектори**
- **Псевдо елементи и псевдо класове**

CSS селектори

В предишни раздели обяснихме тежестта на селекторите, но не стана много ясно каква е разликата между class и id.

- Класове:

Ако искаме да зададем еднакви стилове на много елементи най-добре е да използваме класове. Класовете са CSS метод чрез който оказваме, че един или повече елементи са от една група. Какъвто и стил да напишем, той ще се отрази на всички елементи с този клас.

HTML:

```
<div class=”className1”> some text</div>
<div class=”className2”> some text</div>
<p class=”className1”> some text</p>
```

CSS:

```
.className1 {
        color: blue;
        font-size: 16px;
        text-decoration: underline;
}
```

Резултат:

some text

some text

some text

Независимо че първият елемент от групата на класа “className1”, е “div” а втория е “p”, стиловете се прилагат и за двата елемента.

- ID:

За разлика от класовете, на страницата не може да има две ID-та с еднакво име. Също така всеки елемент може да има само едно ID.

```
<div class=”idName1” class”className1”> some text</div>
```

Както се вижда от примера може да имаме едновременно клас и Id на един и същи елемент.

CSS:

```
(1) div#idName1 { color:red; }
(2) #idName1 { color:red }
(3) div.className1 { color:red }
```

(1) (2) и (3) са еквивалетни но са с различна тежест на селекторите.

Съществуват и други подходи за селектиране на елементи:
* - селектира всичко

[attribute] –

```
a[target="_blank"] {
   color: yellow;
}
```

Ако приложим горния стил за следния HTML:

```
<a href="#" target="_blank">Some link</a>
<a href="#" target="_parent">Some link2</a>
```

Получаваме → Some link Some link2

HTML наследяване

HTML наследяването и каскадния механизъм на CSS определят как браузърите да прилагат правилата за стилизация. Всички HTML елементи имат родител с изключение на <html>. Наследяването представлява вземането на някой от стиловете на родителя. Някой стилове

се наследяват други не. Каскадният механизъм определя кой стил да се приложи, когато конфликтни правила се прилагат на същия елемент.

Групиране на селектори

Ако искаме да стилизираме няколко елента по един същи начин, но те не притежават един и същи клас, се налага да изброяваме:

```
h1, div.clasName1, p { color:red; }
```

Текста във всеки “h1” елемент, всеки “div” с class=”className1”, всеки “p” ще бъде червен.

Псевдо елементи и псевдо класове

Псевдо елементи - позволяват стилизирането на конкретни части от даден елемент.

```
p::first-line {
   color: red;
}
```

Променя цвета на първия ред от даден текст от “p” елементи. По подобен начин може да се промени стила на първата буква(::first-letter), да се постави нещо непосредствено преди дадения елемент(::before), да се промени цвета на маркират текст(::selection).

Псевдо класове – използват се за селектиране на определени елементи от страницата, в случай че не можем да ги достъпим с обикновен клас или id.

Някой елементи имат специфични псевдо класове. “input” елемента е типичен пример.

- input**:enabled** – селектира всички контроли, с които е възможно да се работи
- input**:disabled** – селектира всички контроли, с които не може да се работи
- input**:checked** – селектира всички контроли, които са (отбелязани/маркирани)

- First child

```
p:first-child {
   color: blue;
}
```

Селектира всеки “p” елемент, който е първи наследник на някой елемент.

- First of type

```
div p:first-of-type {
   background: #ff0000;
}
```

Задава червен цвят на първия “p” елемент, чийто родител е “div” елемент.

Повече за псевдо елементите и класовете може да видите http://www.w3schools.com/css/css_pseudo_classes.asp. Да се изпише пълният линк.

Модул 8: Проектиране на раздели на съдържанието

- **CSS Box Model**
- **Inline и Block елементи**
- **Размери на елемент и позициониране**
- **Overflow и resize**

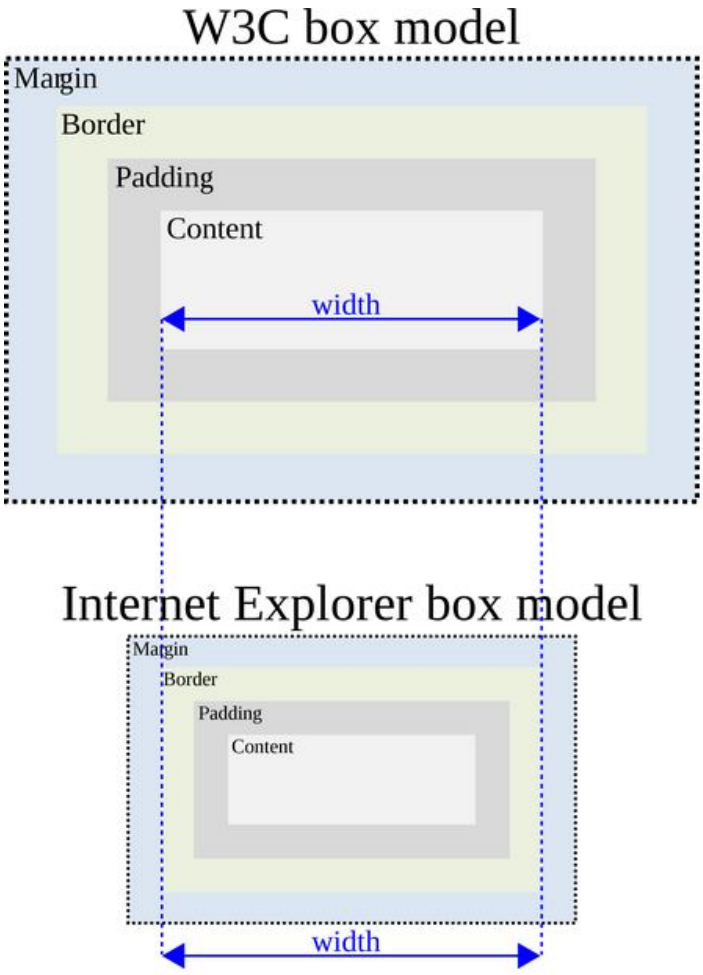
CSS Box Model

Box модела представлява съвкупност от 4 концентрични полета. Това са съдържанието на даден елемент, неговия padding, border и margin.

- Margin – чрез него се задава разстоянието между два или повече съседни елемента.
- Border – определя рамка около елемента.
- Padding – определя разстоянието между съдържанието на даден елемент и border-а му.
- Content – не е css свойство. Представлява съдържанието на самия елемент.

Съществуват два box модела, този на Internet explorer и този на W3C- World Wide Web Consortium (основната световна организация по стандартизация). Долната графика показва основните разлики между тях. По подразбиране в Internet explorer се използва Internet explorer box model (border-box). Във всички останали браузъри по подразбиране се използва

W3C box model (content-box). Можем да променяме box mode-а като променим стойността на css свойството “box-sizing”, с желаната от нас.



Inline и Block елементи

Както споменахме и по-рано Inline елементите заемат място колкото им е необходимо за да се визуализират, докато Block елементите заемат целия ред. Затова спокойно можем да

нареждаме Inline елементи един до друг. Това може да направим и с Block елементи, но само с използването на позициониране, в противен случай елементите се нареждат един под друг. Типични Inline елементи са: <a>, , <b>, , <i>, и <mark>. Типични блокови елементи са: <p>, , <form>, <header>, <nav>, <ul>, <li> и <h1>.

Размери на елемент и позициониране

- Реални размери на елементите:
  - Сантиметри
  - Милиметри (mm): 10 милиметри = 1 сантиметър
  - Инчове (in): 1 инч = 2.54 сантиметра
  - Picas (pc): 1 pica = 12 points = 1/6 от инч
  - Points (pt): 1 point = 1/72 от инч
  - Pixels (px): 1 pixel = 1/96 от инч

- Относителни размери:
  - На шрифтове
    - em: 1em = текущият размер на шрифта на текущия елемент
    - ex: 1ex = височината на малко x, или 0.5 em ако не може да се изчисли.
    - ch: 1ch = широчината на символа 0
    - rem: 1rem = широчината на шрифта на html елемента(16px default)
  - Свързани с видимата част от екрана (viewport)
    - vw: 1vw = 1% от широчината на viewport
    - vh: 1vh = 1% от височината на viewport
    - vmin: vmin = по-малкото от vh и vw

Повече http://kyleschaeffer.com/development/css-font-size-em-vs-px-vs-pt-vs/ .

Позициониране на елементи

Свойството за позициониране в CSS, което не сме разгледали е position. Използва се със следните стойности: absolute, relative, static, fixed.

- Static е нормалното състояние на елементите. Трябва да знаем че всички останали трябва да използваме само тогава когато не можем да се справим по друг начин.
- Absolute – оказва отместване от ръба на контейнер, който има relative-на позиция. Изрично трябва да се зададе двойка стойности за отместването на несрещуположни величини. Най-често се задава двойката отместване отгоре – top с отместването отляво – left.

- Relative – подобно на absolute, с разликата че отместването се извършва от нормалната позиция на елемента.

Overflow и resize

- Overflow – свойство в css, което се използва за скриване или показване на съдържанието, в случай че не се побира в елемента. Възможните стойности на това свойство са – visible, hidden, scroll, auto.
    - Visible – съдържанието което не се събира в елемента е видимо.
    - Hidden – съдържанието което не се събира в елемента е скрито.
    - Scroll - съдържанието което не се събира в елемента е скрито, но може да бъде видяно посредством скролиране. В този случай и хоризонталния и вертикалния скрол са видими, незамисимо дали има нужда от тях. За да оправим този проблем, съществуват по-конкретни свойства(overflow-x, overflow-y).
    - Auto – ако има нужда добавя скрол.

  Overflow-x – показва хоризонтален скрол при нужда.

  Overflow-y – показва вертикален скрол при нужда.

  HTML:

```
<div>
        <p> HTML content </p>
</div>
```

  CSS:

```
div {
        background-color: green;
        overflow: scroll;
}
```

Модул 9: Анимации с CSS

- **CSS преходи**
- **Управление на преходите**
- **Transition timing**
- **Трансформации на елементи**
- **Анимации**
- **Програмно стартиране на анимации**

Анимациите са често срещани в съвременните страници. Те са ефектни и красиви.

CSS преходи

Преходите ни позволяват да променим стойността на определена свойства от css по плавен начин за определено време. Преди да започнем трябва да знаем че преходите и всички свойства използвани за преходи не се поддържат от стари браузъри.

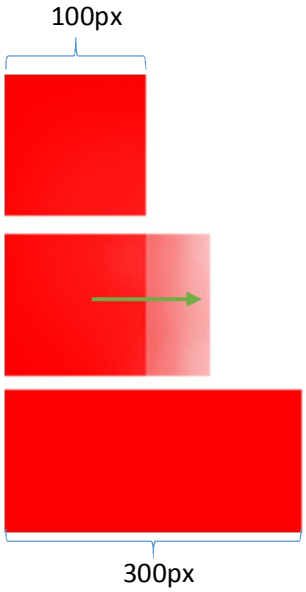
Property	IE	Chrome	Firefox	Safari	Opera
transition, transition-delay, transition-duration, transition-property, transition-timing-function	10.0	26.0, От 4.0 с “-webkit-” префикс	16.0, От 4.0 с “-mox-” префикс	6.1, От 3.1 с “-webkit-“ префикс	12.1, От 10.5 с “-o-“ префикс

В таблицата-се вижда от коя версия на най-използваните браузъри се поддържа transition properties. Посочени са и по стари версии на тези браузъри които поддържат transition, но с помощта на така наречените префикси. За всеки браузър префиксите са различни:

```
-webkit-transition: width 2s;  /* Safari */
transition: width 2s;
```

Нека разгледаме един пример за transition:

```
div {
   width: 100px;
   height: 100px;
   background: red;
   border:1px solid black;
   -webkit-transition: width 2s;  /* For Safari 3.1 to 6.0 */
   transition: width 2s;
}
div:hover {
   width: 300px;
}
```



Този код оказва че при поставяне на курсора над div елемента, широчината на този елемент от 100px става 300px, като това се случва плавно за 2 секунди. Демо може да видите http://www.w3schools.com/css/tryit.asp?filename=trycss3_transition1 .

С помощта на останалите свойства от групата на transition, можем да усложним прехода:

- transition-property – оказва свойството спрямо което ще приложим преход (width, height…). Повече за него може да видите http://www.w3schools.com/cssref/css3_pr_transition-property.asp .
- transition-duration – оказва продължителността на прехода. Повече за него http://www.w3schools.com/cssref/css3_pr_transition-duration.asp .
- transition-timing-function – оказва по какъв начин ще варира скоростта на прехода. За повече информация: http://www.w3schools.com/cssref/css3_pr_transition-timing-function.asp.
- transition-delay – определя забавянето преди началото на прехода. Повече http://www.w3schools.com/cssref/css3_pr_transition-delay.asp .
- transition – използва се за кратък запис на всички изброени до тук.

Трансформация на елементи

Използва се чрез свойството “transform”. С негова помощ можем да извършваме двуизмерни и триизмерни трансформации на елементи. Както и преходите, трансформациите не се подържат от всички браузъри:

Property	Chrome	IE	Firefox	Safari	Opera
transform(2d)	36.0, От 4.0 с “-webkit-” префикс	10.0, От 9.0 с “-ms-” префикс	16.0, От 3.5 с “-moz-” префикс	От 3.2 с “-webkit-” префикс	23.0, От 15.0 с “-webkit-” префикс
Transform(3d)	36.0, От 12.0 с “-webkit-” префикс	10.0	16.0, От 10.0 с “-moz-” префикс	От 4.0 с “-webkit-” префикс	23.0, От 15.0 с “-webkit-” префикс

(2D)

- translate – премества текущата позиция на даден елемент в зависимост от зададените параметри по хоризонтала и вертикала.

```
div {
    -ms-transform: translate(50px,100px); /* IE 9 */
    -webkit-transform: translate(50px,100px); /* Safari */
    transform: translate(50px,100px);
}
```

Този код ще премести div елемента 50px в дясно и 100px надолу.

- rotate - завърта даден елемент по часовниковата или обратно на часовниковата стрелка посока, в зависимост от зададени градуси като параметри.

```
div {
   -ms-transform: rotate(20deg); /* IE 9 */
   -webkit-transform: rotate(20deg); /* Safari */
```

```
  transform: rotate(20deg);
}
```

Този код ще завърти div елемента с 20 градуса по часовниковата стрелка.

- scale – преоразмерява даден елемент, като го увеличава или намалява, в зависимост от зададен параметър.

```
div {
  -ms-transform: scale(2,3); /* IE 9 */
  -webkit-transform: scale(2,3); /* Safari */
  transform: scale(2,3);
}
```

Този код увеличава широчината два пъти и височината три пъти.

- skew – наклонява даден елемент по хоризонталата или вертикалата.
  - skewX() – наклон спрямо хоризонталата.

```
div {
    -ms-transform: skewX(20deg); /* IE 9 */
    -webkit-transform: skewX(20deg); /* Safari */
    transform: skewX(20deg);
}
```

  - skewY() – наклон спрямо вертикалата.

3D

- translate3d – използва се за триизмерна транслация
- scale3d – използва се за триизмерно преоразмеряване
- rotate3d – използва се за триизмерна ротация
- perspective – използва се за перспектива

Анимации

Не се поддържа от всички браузъри:

Property	IE	Chrome	Firefox	Safari	Opera
		От 4.0 с	16.0	От 4.0 с	От 15.0 с
	10.0	“-webkit-”	От 5.0 с	“-webkit-”	

animation, @keyframes — префикс "-moz-" префикс — префикс "-webkit-" префикс

```
/* The animation code */
@keyframes example {
   from {background-color: red;}
   to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
   width: 100px;
   height: 100px;
   background-color: red;
   animation-name: example;
   animation-duration: 4s;
}
</head>
<body>
<div></div>
</body>
</html>
```

Този код променя цвета на div елемента преливайки от червен в жълт за определен интервал.

Свойства за задаване на анимацията:

- animation-name – определя име на анимацията, което се използва в @keyframes
- animation-duration – определя колко време е необходимо за изпълнение на анимацията
- animation-timing-function – определя скоростта(ускорение) за изпълнение на анимацията
- animation-delay – определя време което се изчаква преди стартиране на анимацията
- animation-iteration-count – определя колко пъти трябва де се изпълни анимацията
- animation-direction – оказва посоката на изпълнение на анимацията.

Модул 10: Изработка на адаптивен потребителски интерфейс

- **Поддръжка на различни устройства**
- **Media Types (типове медиини файлове)**
- **Media queries (заявка за медиен файл)**
- **Stylesheet (лист със стилове) за печат**

Поддръжка на различни устройства

Изисква се от съвременните страници да бъдат красиви, функционални но все по често се говори за поддръжката на различни устройства. Това се налага, поради засиления интерес на хората към новите технологии и по-конкретно към смартфони, таблети, четци и др. Тези устройства предоставят достъп до интернет и се очаква че страниците трябва да са удобни за боравене на тях, независимо от размерите и вида на устройството.

Добра практика е да се създава страницата първо за мобилно устройства и чак след това за desktop. Всичко трябва да е с необходимите размери, да е четимо и удобно за боравене с пръсти.

Media Types

HTML използва атрибута media(медия), за да наложи употребата на даден stylesheet (лист със стилове) за даден тип устройство. В CSS съществува сходна функционалност - @media.

Типове:

- **speech**: Speech synthesizers (синтезатори на реч)
- **braille**: Braille tactile feedback screen readers (брайлови четци)
- **embossed**: брайлови принтери
- **handheld**: мобилни устройства
- **print**: Print preview:екрани и печатане

- **projection**: проектори
- **screen**: компютърни екрани
- **tty**: Teletypes (телетекст)
- **tv**: телевизори и други екрани с малка резолюция и ограничена възможност за скролиране
- **all**: за всички устройства

Media Queries

Чрез media queries (заявка за медиен файл) можем да зададем различно поведение на страницата, в зависимост от вида на устройството. Самата media queries наподобява селектор за видовете устройства, според ширината, височината, резолюцията, ориентацията и др на екрана им. Стиловете за определено устройство се записват вътре в media queries, между къдрави скоби. Пример за media queries:

```
@media screen and (max-width: 800px)  {
        p {
                color:red;
        }
}
```

Задава червен цвят на текста във всеки “p” елемент, за устройства с максимална ширина на дисплея 800 пиксела.

- **width, height**: широчина и височина на видимото поле
- **device-width, device-height**: широчина и височна на екрана на устройството(или на хартията при печат)
- **orientation:** ориентация
- **resolution:** разделителна способност
- **aspect-ratio**: съотношение между височина и широчина на видимото поле
- **device-aspect-ratio**: съотношение между широчина и височина на екрана
- **color**: колко бита информация за цвят поддържа дисплея

- **color-index**: колко цвята поддържа дисплея
- **scan**: метод на сканиране на телевизора - progressive или interlace.
- **grid**: grid или bitmap. Обикновено устройствата са bitmap базирани. Пример за grid устройство е TTY терминал или телефон с течнокристален дисплей, който потдържа само един шрифт с еднаква ширина на буквите. При grid устройство се задава стойност 1, в противен случай 0.

Повече информация http://www.w3schools.com/cssref/css3_pr_mediaquery.asp .

Дефиниране на Style Sheets за печат

- Добавят се стилове за печат, които отговарят са изгледа на елементите при разпечатване
    - Задава се **print** media type в CSS правилата
- Изпълняват се следните оптимизации
    - Премахват се header, footer навигация, background, графики и анимации
    - Задава се големината на шрифта и се премахват ефектите
    - Поставя се съдържанието само в една колона
    - Задава се големината и позицията на разпечатваната страница

Модул 11: Въведение в JavaScript

- **Структура и употреба**
- **Функции и оператори**
- **Библиотеки**
- **Връзка между JavaScript и HTML**
- **JSON**
- **Document Object Model**
- **jQuery**

Структура и употреба

Javascript е интерпретативен език за програмиране, чрез който можем да добавим функционалност в нашата страница. Поддържа обектно-ориентиран и функционален стил на програмиране. Подобно на стиловете, и функционалността за дадена страница е хубаво да се отдели в отделен файл. Използва се за добавяне, премахване, промяна на текст и елементи от страницата, валидация на данни, и много други. Подобно на други езици за програмиране javascript, притежава променливи, оператори, функции, цикли, класове и др. Съществуват и коментари. Има два вида коментари в javascript – за конкретен ред или за точно определена последователност от символи, независимо от редовете.

// коментар за текущия ред

/*многоредов коментар …….
…….. ……край на коментара*/

Всички променливи се дефинират чрез запазената дума “var”.

```
var thisVariable = 3;
```

Променлива със стойност 3. Javascript автоматично определя типа на променливата избирайки от три дефинирани за този език : string (низ), number (числен), boolean (булев). Променливите могат да приемат и стойностите undefined (недефинирана) и null (неопределен). Jacascript подържа различни видове оператори - аритметични, за присвояване, за сравнение, булеви и условни.

Функции и оператори

- Функции – в javascript дефинирането им може да става по няколко начина в зависимост от целите и предпочитанията ни. Повече http://www.w3schools.com/js/js_functions.asp . Аргументите са достъпни само в тялото на функцията. Функциите могат да връщат резултат. В тялото на функцията може да се декларират локални променливи. Глобалните променливи дефинирани извън функцията са достъпни от всички функции.
- Условни оператори – оказват изпълнението на дадено действие, в случай че е изпълнено определено условие и друго действие, ако условието не е изпълнено.
    - If (ако)
    - Switch (превключи)

Повече за тях http://www.w3schools.com/js/js_if_else.asp и http://www.w3schools.com/js/js_switch.asp.

- Цикли – изразяват повтарящо се действие да настъпването на определено действие (край на цикъла).
  - While (докато)
  - Do while (повтаряй докато)
  - For (за)

  Повече информация http://www.w3schools.com/js/js_loop_for.asp и http://www.w3schools.com/js/js_loop_while.asp .

- Сложни типове – javascript поддържа сложни типове - string (низ), date (дата), array (масив), RegExp?????????????

Библиотеки

Javascript е изключително разпространен в днешно време и поради тази причина постоянно се появяват нови библиотеки. По известни са jQuery, jQuery UI, Kendo UI, WinJS, Bootstrap, Angular JS и др. В този раздел ще разгледаме по-подробно jQuery.

Връзка между JavaScript и HTML

Подобно на CSS, javascript може да се добави към страницата по няколко начина. Най-често използвания е добавянето му във външен файл, който се декларира в “body” тага, в най-долната му част.

```
<body>
    <!-- some thml -->
    <div class="test" id="element">       some text</div>
    <!--  some html -->
    <script src="Script/script.js"></script>
</body>
```

src атрибута оказва адреса на който се намира нашия script файл, спрямо текущата директория.

JSON

JSON – javascript object notation (език за означаване на обекти посредством javascript) е текстово базиран отворен стандарт създаден за човешки четим обмен на данни. Произлиза от скриптовия език JavaScript, за да представя прости структури от данни и асоциативни масиви, наречени обекти. Въпреки своята връзка с JavaScript, това е езиково независима спецификация, с анализатори, които могат да преобразуват много други езици в JSON. JSON е формат за сериализация на обекти.

```
var attendees = [
   {
      "name": “Eric Gruber",
      "currentTrack": "1"
   },
   {
      "name": “Martin Weber",
      "currentTrack": “2"
   }
]
```

Document Object Model

DOM (обект за модел на HTML документа) предоставя програмен API – Application Programming Interface (приложно-програмен интерфейс) за контролиране на браузъра и достъп до съдържанието на страницата:

- Намирането и определяне на стойностите на елементите на страницата
- Обработка на събития за контрол на една страница
- Промяна на стилове, свързани с елементи
- Сериализиране и десериализиране на страница като XML документ
- Валидиране и актуализиране на уеб страници

За да промените елемент на дадена страница трябва да изпълните следните стъпки:

- Създаване на нов обект, съдържащ новите данни.

- Намиране на родителския елемент, който трябва да съдържа новият обект.
- Добавяне, вмъкнете или промяна на данните в новият обект.

Следния код:

```
var node = document.createElement("LI");
var textnode = document.createTextNode("Water");
node.appendChild(textnode);
document.getElementById("myList").appendChild(node);
```

създава списъчен елемент(li) и го добавя към списък с id=”myList”. За премахване на елементи се използва метода “removeChild”, а за премахване на атрибут – “removeAttribute”.

jQuery

JQuery е javascript библиотека. Използва се за улеснение на достъпа до елементите от HTML документа, като по този начин позволява лесно изграждане на динамична функционалност. За да използваме jquery е необходимо да използваме jquery.js файл, който можем да изтеглим https://jquery.com/ , и да декларираме в най-долната част в body елемнта, непосредствено преди нашия script файл.

Може да използваме jquery за задаване на стилове:

```
<script type="text/javascript">
   $(document).ready(function () {
      $("h2").each(function () {
         this.style.color = “red";
      });
   });
</script>
```

За да се определи кои елементи ще бъдат премахвани или модифицирани се използва селектор. JQuery селекторът се състои от част оказваща изрично че ползваме jquery – “$” и CSS селектор:

```
$(“p.className1”).addClass(“className2”);
```

Селектираме всеки “p” елемент с class=”className1” и му добавяме “className2” клас. Други често използвани методи са append, detach, replaceWith, val и др.

```
$( ".inner" ).append( "<p>Test</p>" );
```
- добавя параграф в стойността на елемент с клас “inner”

```
p = $( "p" ).detach();
```
- премахва първият параграф от DOM и присвоява информацията свързана със всичките му характеристики на JavaScript обекта с име “p”

```
$( ".third" ).replaceWith( $( ".first" ) );
```
- заменя елемента с клас “third” с елемент с клас „first”

Node JS

Node.js е платформа, разработена на open-source engine (отворен код)за JavaScript на Google – V8. С помощта на Node.js можем да създаваме бързи и мащабни мрежови приложения. Node.js използва събития, неблокиращ I/O – input/output (вход/изход) модел, който позволява лекота и ефективност. Перфектен е за приложения, които имат голям интензитет на прехвърляне на данни в реално време.

Node е базиран на събития и е асинхронен. Събития, като например HTTP заявка, ще извика JavaScript функция, която ще свърши малко работа и ще пусне други асинхронни задачи като връзка с базата или теглене на съдържание от друг сървър. Щом тези задачи бъдат пуснати, същата функция за извикване на събитие ще спре и Node ще заспи. В момента, в който нещо друго се случи – например установяване на връзка към базата отговор на сървъра, се извиква callback функция и се изпънява още JavaScript код, който може да пуска още асинхронни задачи. По този начин, Node.js ще разпредели много дейности за много паралелни работни потоци. Ето защо Node се справя толкова добре, когато трябва да управлява хиляди конекции едновременно.

За повече информация:

http://bg.wikipedia.org/wiki/Node.js

http://help.superhosting.bg/node-js.html

Knockout JS

Knockout е библиотека, изработена върху JavaScript. Тя се основава на три основни принципа.

1) Проследяеми променливи (observables) и проследяване на зависимостите.

2) Декларативно указване на свързаност (declarative bindings).

3) Шаблони (templates).

Чрез проследяемите променливи се постига синхронизация на модела съдържащ данните за представяне (ViewModel) с потребителския интерфейс (UI), т.е. при възникване на промяна в модела тя автоматично се отразява визуално в интерфейса. Дефинирането на такъв обект става аналогично на дефиницията на обикновен JavaScript обект, с малки специфики. Указването на това как визуално трябва да бъде представен моделът, т.е. на връзката между отделната променлива и нейното визуално представяне става декларативно чрез специалния „data-bind" атрибут. В контекста на предходния пример: Библиотеката предоставя набор от дефинирани правила за свързаност (bindingHandlers), т.е. конкретен начин на обработка на стойността при визуализиране или прочитане от UI. Те могат да бъдат използвани в по-голямата част от стандартни сценарии, които се срещат при повечето уеб приложения. Друго голямо предимство е възможността за разширяемост и предефиниране на вече предоставена функционалност. Всеки, който използва този продукт, има опцията да дефинира собствени правила или да предефинира стандартните, които се предоставят. След като са дефинирани всички зависимости и правила за дадения обект (ViewModel), те се прилагат чрез извикване на конкретна функция:

За повече информация:

http://en.wikipedia.org/wiki/Knockout_(web_framework)

http://knockoutjs.com/

Модул 12: AJAX

- **HTTP**
- **Проследяване на комуникацията**
- **Fiddler**
- **AJAX – предимства и недостатъци**
- **AJAX чрез jQuery**

HTTP

HTTP (протокол за трансфер на хипертекст) е мрежов протокол, от приложния слой на OSI модела, за пренос на информация в компютърни мрежи. HTTP определя 8 различни клиентски метода за заявки:

HEAD — иска изпращане на заглавията отговарящи на посочения с URL ресурс. Отговорът на сървъра е идентичен с този на GET, но е с липсващо тяло.

GET — с него клиентът прави заявка за ресурс, зададен чрез URL. Могат да се изпращат и ограничено количество данни, закодирани директно в самия URL.

POST — позволява клиентът да изпрати данни на сървъра. Тази заявка обикновено се генерира при изпращането на уеб формуляр, а данните могат да бъдат: текст, написан от потребителя във формуляра; файл на клиентския компютър и др.

PUT — качва файл, който в бъдеще ще отговаря на посочения URL.

DELETE — изтрива посочения ресурс.

TRACE — сървърът връща получената заявка със статус OK. Позволява да се провери в какъв вид пристига заявката при сървъра и дали (и как) е била модифицирана по трасето от междинни прокси сървъри.

OPTIONS — сървърът трябва да отговори с поддържаните от него клиентски методи, съответстващи на зададения URL, или с поддържаните от сървъра методи като цяло, ако е зададено * вместо URL.

CONNECT — използва се при комуникация през прокси.

За примери за цялостни HTTP завки:

http://www.opencalais.com/httpexamples

Fiddler

За проследяването на тези HTTP заявки е хубаво да използваме допълнително приложение. Такова приложение е fiddler. При презареждане на страницата той проследява всички заявки и ги визуализира, показвайки типа на заявката и получения резултат. Може да изтеглите безплатно fiddler от http://www.telerik.com/fiddler .

AJAX

AJAX - asynchronous javascript and xml (асинхронен javascript и xml) представлява похват в уеб разработките за създаване на интерактивни уеб приложения. Предимството на Ajax е, че посредством използването му уеб страниците се зареждат по-бързо. Посредством асинхронен обмен на малки порции данни “зад кадър” може да се променят части от уеб страницата. По този начин се намалява количеството информация, която се трансферира между сървъра и клиента. Асинхронността позволява да не бъде необходимо да се

презарежда цялата страница отново. По този начин се повишава интерактивността, скоростта и функционалността на страниците.

AJAX и jQuery

JQuery предоставя някой полезни метода за AJAX. По използвани от тях са:

jQuery.load, jQuery.get, jQuery.post и jQuery.ajax.

- jQuery.load – може да се използва за зареждане на съдържание(файлове и др) в елементи от страницата. Следния код:

```
$("button").click(function(){
   $("#div1").load("demo_test.txt");
});
```

добавя съдържанието на “demo_test.txt” файла към “div” елемент с id=”div1”.

- jQuery.get – изпраща HTTP GET заявка и взима резултат от заявката:

```
$("button").click(function(){
   $.get("demo_test.asp", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
   });
});
```

- jQuery.post – може да се използва за зареждане на данни от сървъра, посредством HTTP POST заявка:

```
$("button").click(function(){
   $.post("demo_test.asp", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
   });
});
```

- jQuery.ajax – може да се използва с цел промяна на даден елемент. Следния код:

```
$("button").click(function(){
   $.ajax({url: "demo_test.txt", success: function(result){
      $("#div1").html(result);
```

```
        }});
    });
```

променя съдържанието на “div” елемент, посредством AJAX заявка.

Всички методи може да видите http://www.w3schools.com/jquery/jquery_ref_ajax.asp .

Модул 13: Структура на JavaScript

- **Област на действие на променливите**
- **Създаване на прости обекти**
- **Object Literal Notation**
- **Конструктори**
- **Прототипи**
- **Капсулация**
- **Наследяване на данни и функционалност**

Област на действие на променливите (scope)

В JavaScript има две области на действие на променливите:

- Глобална – достъпна е навсякъде, дефинира се със запазената дума “var”, освен ако не искаме да дефинираме глобална променлива в тялото на функция. Тогава пропускаме “var”.

```
var variableName1 = "some value";   //глобална променлива
function myFunction() {
   variableName2 = "some value";    //глобална променлива
}
```

- Локална, в тялото на дадена функция – достъпна е само в тялото на функцията. Дефинира се със запазената дума “var”.

```
function myFunction() {
   var variableName2 = "some value";    //локална променлива за тази фунция
}
```

Не се подържа област на действие в блок от код. Ако се декларира променлива в блок от код, нейният обхват е в цялата функция.

Какво правим ако имаме две променливи с еднакви имена с еднакъв scope(обхват). Javascript има механизми за справяне с този проблем.

- Immediate functions (непосредствени функции) – това са функции, които се дефинират еднократно на мястото, където се прибягва до тяхното ипълнение. Immediate function се дефинира чрез следният синтаксис:

```
(function () {


})();
```

- Namespaces – това са области от имена на променливи и обекти, които в JavaScript се реализират чрез дефиниране на главен обект, който да съдържа необходимите имена.
- Strict mode – режим на изпълнение на кода, чрез който се избягват системни грешки. Активира се за целият скрипт или за отделни функции чрез фразата „use strict”

Когато имаме две променливи с еднакво име но но различна област на действие, локалната променлива “скрива” глобалната:

```
foo = "foobar";   //глобална променлива
var bar = function(){
   var foo = foo || "";    //локална променлива
   return foo;
}
bar();
```

Функцията “bar()” връща празен string(низ), защото в scope-а на функцията ние на практика присвояваме на локалната променлива, стойността на самата променлива, а не тази на глобалната променлива. Тъй като преди това не сме задали стойност тя остава undefined и се изпълнява втората част от реда и нашата променлива приема стойност “” – празен string (низ).

Създаване на прости обекти

Обектите са сложен тип данни в javascript. Създаването на нови обекти може да стане по няколко начина.

```
var object1 = new Object();

var object2 = {};
```

Това са два празни обекта. Обектите могат да притежават специални полета - properties(свойства) и methods(методи). С тяхна помощ можем да опишем даден предмет, събитие и др, разбираемо не само за машина, но и за човека. Чрез полетата и методите оказваме, че даден обект притежава определени свойства, характерни само за този конкретен обект, също така чрез тях задаваме поведения на обекта, в зависимост от определени събития.

Object Literal Notation

Предоставя по-лесен начин за създаване на обекти и дефиниране на методи и полета към тях.

```
var object1 =  {
        propertyName: “some value”,
        propertyName2: 56
        methodName: function(value) {
                //some body
        }
}
```

Най-общо полетата представляват променливи, а методите – функции.

Конструктори

Представлява метод, чрез който се задават стойностите на полетата (properties) на даден обект. Осигурява по-добра сигурност на данните – може да се добави валидация на данните.

- this - специална запазена дума в повечето езици за програмиране, която сочи към текущия обект. Използва се при конструкторите:

```
var Account = function (id, name) {
```

```
    this.id = id;
    this.name = name;
    this.balance = 0;
    this.numTransactions = 0;
};
```

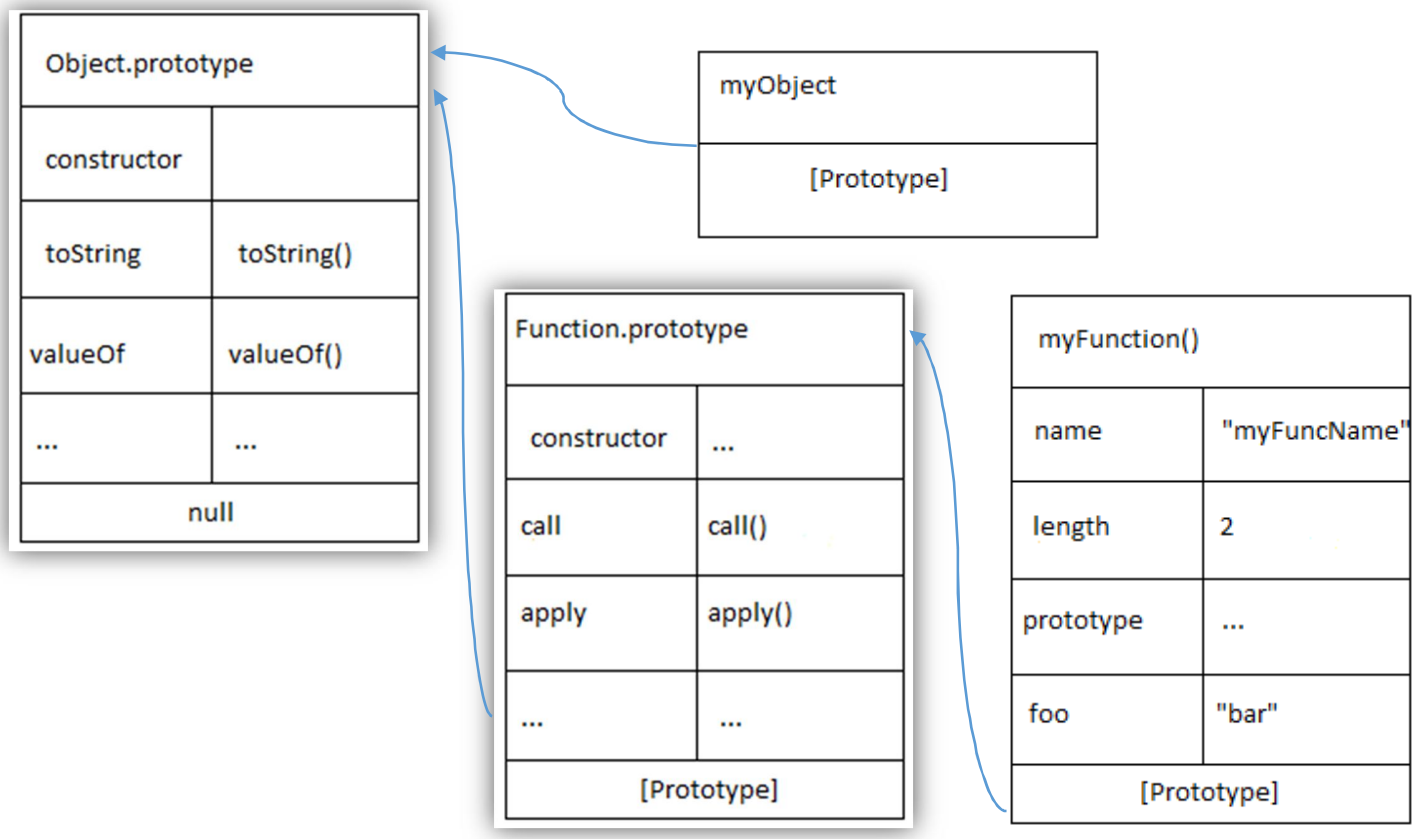
Конструкторът се извиква при създаването на обекти:

```
var acc1 = new Account(1, "John");
var acc2 = new Account(2, "Mary");
```

Прототипи

Всички обекти в javascript притежават прототип. Прототипът също е обект. Всички обекти в javascript наследяват своите полета и методи от своя прототип. Функциите и обектите имат различни прототипи. Обектите наследяват object.prototype обекта, а функциите наследяват function.prototype обекта. Function.prototype обекта от своя страна наследява object.prototype обекта. Всеки наследник може да ползва методите и полетата на своя родител а също така да ги променя, както и да създава нови такива.

Фиг. 1



Капсулация

Използва се за скриване на част от информацията в даден обект с цел по-голяма сигурност. Прави невъзможно за потребителите на даден обект да променят неговото вътрешно състояние по неочакван начин. За да се променят тези стойности се използват вътрешните методи на този обект. По този начин дефинираме така наречените private(скрити) променливи които не се виждат от "външния свят".

```
var Person = function(name, age)
{
   // Private поле.
   var _name, _age;

   // метод достъпен от вън, чрез който променяме private полето.
   this.getName = function()
   {
      return _name;
   }
   …
}
```

Наследяване чрез Chaining Prototypes

Промер за това наследяване е фиг. 1.

Подбробно обяснение за chaining prototypes може да видите http://www.objectplayground.com/ .

Модул 14: Реализиране на графика чрез CSS и JavaScript

- **Създаване на интерактивни графики чрез SVG**
  - **Фигури**

- **Фон**
- **Градиенти**
- **Трансформация**

- **Рисуване на графики, използвайки Canvas API**
  - **Градиенти**
  - **Трансформация**

SVG - Scalable Vector Graphics (мащабируема векторна графика) и Canvas са методи за графично изобразяване на обекти в страници. Могат да се използват за анимации и др. Основната разлика между SVG и Canvas е че чрез SVG се създават мащабилуеми векторни изображения, докато с Canvas се създават растерни изображения.

Създаване на интерактивни графики чрез SVG

SVG е базиран на XML маркиращ език за описване на двуизмерна векторна графика с възможност за включване и на растерни изображения. Възможностите на SVG могат да се разделят на 14 раздела:

- **Пътеки** — могат да се създават прави линии, многосегментни линии, криви на Безие.
- **Основни форми** — включва правоъгълници, окръжности, елипси, линии, многосегментни линии, полигони. Всички те могат да бъдат създадени и чрез пътеки, но готовите форми са по-лесни за използване.
- **Текст** — използва се Уникод кодировка. Върху символите могат да бъдат прилагани ефекти. Текстът може да бъде двупосочен, вертикален и разположен по протежение на крива.
- **Оцветяване** — SVG формите могат да бъдат запълвани и очертавани с цвят. Запълването, освен това, може да бъде и с градиент или шарка, степента му на прозрачност също може да се задава. Очертанията от пътеки, линии, полигони имат и свойството наконечник (marker) който представлява графичен обект поставян в краищата на линии или по ъглите на сложни очертания — най-често използван за създаване на стрелки. Може да се прилага и разнообразие от пунктировки задавани чрез дължини на отделните съставящи ги чертици.
- **Цвят** — задава се чрез своите RGB (червена, зелена, синя) компоненти.

- **Градиенти и шарки**. Градиентите са линейни и радиални и могат да включват неопределен брой цветове. Шарките се дефинират чрез растерни или векторни обекти които се повтарят по x, y или и двете координати.
- **Изрезки, маски и съчетаване**. Очертания на пътеки, текст и основни форми могат да се използват за дефиниране на изрезки и маски. Обектите попадащи в очертанията на изрезките се изобразяват, а тези извън тях — не. Маските пък задават и прозрачността на обектите попадащи в тях — прозрачността на всеки пиксел от обекта използван като маска дефинира прозрачността на обектите към които е приложена маската. Самите обекти използвани като маски или изрезки не се изобразяват.
- **Филтърни ефекти** — разнообразни графични трансформации които могат да се прилагат върху графичните обекти.
- **Интерактивност** — в този раздел на спецификацията се дефинират отделните събития (движение на мишката, натискане на клавиш и др) в отговор на които могат да се задействат анимации или скриптове.
- **Свързаност** — за описание на отношенията между отделни документи се използва езика XLink, това включва и възможност за създаване на еднопосочни препратки по подобие на HTML.
- **Скриптиране** — всеки аспект на SVG документа е достъпен за скриптиране чрез документен обектен модел (DOM). По подразбиране се използваECMAScript, а скрипта се изпълнява в отговор на възникнало събитие както е дефинирано в раздел Интерактивност на спецификацията.
- **Анимиране** — с помощта на анимиращи елементи се описват промени на графичните елементи във времето. Алтернативно, анимация може да се постигне и посредством скриптиране.
- **Шрифтове** — могат да се използват външни файлове с шрифтове по подобие на HTML. Има възможност и за вграждане на глифите на шрифта в документа чрез SVG шрифтове.
- **Метаданни** — има възможност за добавяне на информация описваща самия документ като автор, описание и др.

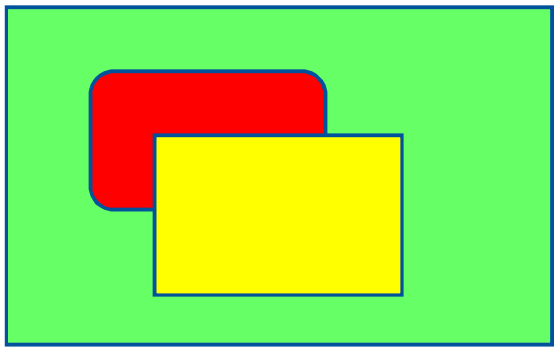
Можем да създадем SVG елемент чрез тага svg:

<svg> … </svg>

Посредством специфични за svg свойства създаваме изображения. Тези свойства представляват елементи и се записват вътре в svg елемента.

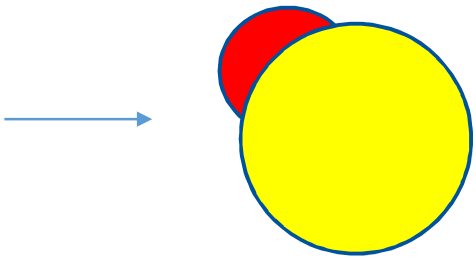
```
<svg xmlns="http://www.w3.org/2000/svg">
<rect x="50" y="50" width="100" height="75"
        rx="20" ry="20" fill="red" stroke="blue" />
<rect x="75" y="75" width="100" height="75,,
         fill="yellow" stroke="blue" />
</svg>
```

Получава се следната фигура:



“x” и ”y” се използват за кординати по хоризонталата и по вертикалата. “width” и ”height” определят съответно широчината и височината на всеки елемент. За да зададем определен цвят като фон можем да използваме запазената дума “fill” последвана от цвета който желаем. С “stroke” можем да окажем цвета на бордъра ограждащ фигурата. За някой фигури се налага използването и на допълнителни свойства – “cx”, “cy”, “r”, “ry”, “rx”:

```
<circle cx="120" cy="80" r="40"
        stroke="blue" fill="red" />
<circle cx="160" cy="120" r="60"
        stroke="blue" fill="yellow" />
```



и

```
<ellipse cx="150" cy="60" rx="110" ry="30"
        stroke="blue" fill="red" />
<ellipse cx="150" cy="140" rx="110" ry="30"
        stroke="blue" fill="yellow" />
```

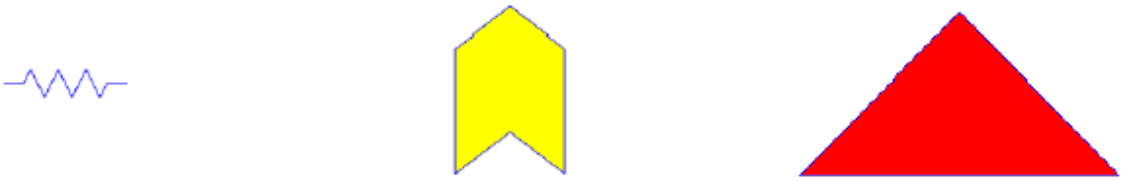
“cx” и “cy” съответстват на позицията на центъра по хоризонталата и вертикалата спрямо svg елемента.

“r” съответства на радиуса на окръжност.

“rx” и “ry” съответстват на радиусите на елипса.

С svg могат да се създават и по-сложни фигури – “polyline”, “polygon”, “path” и др.

polyline polygon path



- Градиенти и шаблони – и тук градиентите биват “linear” и “radial”. Задават се по сходен начин.

“linear” градиент

```
<svg>
    <linearGradient x1="0%" y1="0%" x2="100%" y2="0%">
            <stop offset="0.2" stop-color="yellow" />
            <stop offset="0.5" stop-color="green" />
            <stop offset="1.0" stop-color="red" />
    </linearGradient>
</svg>
```

“radial” градиент

```
<svg>
    <radialGradient id="gradient2" fx="0.3" fy="0.3">
            <stop offset="0.1" stop-color="yellow" />
            <stop offset="0.7" stop-color="red" />
    </radialGradient>
</svg>
```

“x1” и “y1” представляват началната точка на градиента във фигурата.

“x2” и “y2” представляват крайната точка на градиента във фигурата.

<stop> представлява списък от междинни цветове.

Offset отместване спрямо целия градиент.

Stop-color – използвания цвят.

“fx” и “fy” – позиция на фокусната точка.

шаблони - дефинират се чрез <pattern> елемента.

```
<svg>
    <pattern patternUnits=”userSpaceOnUse” width=”100” height=”100”>
            <image xlink:href=”wales.png" x=”0” y=”0” width=”100” height=”100” />

    </ pattern >
</svg>
```

patternUnits – контролирра как ще се показва изображението

Рисуване на графика с Canvas API

Canvas се използва за рисуване на растерни изображения. По подразбиране canvas елемента няма нито рамка нито фон. Притежава начални размери от 300px ширина и 150px височина. Манипулацията върху canvas става с помощта на javascript. Важно е да запомним че canvas не помни какво сме нарисували вече, т.е. ако искаме да редактираме някакво фигура, трябва да я нарисуваме отново като приложим желаните от нас промени. Canvas елементите не се влияят от CSS. Тъй като е базиран на растерни изображения, canvas не е подходящ за екрани с висока резолюция, защото за тази цел трябва повече пиксели да се запълнят, което от своя страна е бавно. В такива случаи е по-добре да използваме SVG.

В началото трябва да създаден canvas елемент и да му зададен id.

```
<canvas id="myCanvas">No canvas support</canvas>
```

В зависимост от изискванията които имаме може да определим колко ще е голям самия елемент (width height) и дали ще има border, background-color, padding или margin. Следва програмната част (в отделен js файл), която за canvas елемент с id=”myCanvas” би трябвало да изглежда по следния начин:

```
var canvas = document.getElementById('myCanvas');
var context = canvas.getContext('2d');
```

Необходимо е първо да достъпим canvas елемента с помощта на javascript селектора getElementById. Запазваме го в променлива, на която изрично задаваме че ще работим с двуизмерни изображения –“canvas.getContext('2d');”. От тук нататък следва добавянето на пътечки, фигури и др, както и самото им стилизиране посредством javascript.

```
context.fillStyle = "red";
context.fillRect(20, 20, canvas.width - 40, canvas.height - 40);
```

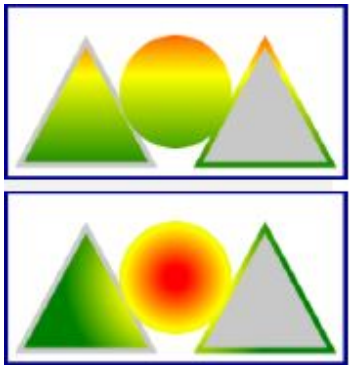
Тези два реда добавят правоъгълник с червен цвят.

Повече за възможните фигури които може да нарисувате чрез canvas може да намерите http://www.w3schools.com/html/html5_canvas.asp и http://www.html5canvastutorials.com/tutorials/html5-canvas-element/ .

- Градиенти и шаблони

Градиент:

```
var grad = ctx.createLinearGradient(x1, y1, x2, y2);
var grad = ctx.createRadialGradient(startCx, startYy,
                                    startRad, endCx, endCy, endRad);
grad.addColorStop(fraction1, color1);
grad.addColorStop(fraction2, color2);
…
ctx.fillStyle = grad;
```



шаблон:

```
var image = document.getElementById("anImageElement");
var pattern = ctx.createPattern(image, "repeat");
ctx.fillStyle = pattern;
```

Модул 15: Разработване на интерактивни страници чрез HTML5 APIs

- **Работа с файлове**
- **Drag-and-drop (провлачване)**
- **HTML видео**
- **HTML аудио**
- **Геолокация**

Работа с файлове

HTML5 API дава възможност на приложенията да достъпват до локалната файлова система. Има четири основни типа структури:

- **Blob** – бинарни данни. Повече https://developer.mozilla.org/en-US/docs/Web/API/Blob .
- **File** – предоставя информация за файла и дава достъп до съдържанието му. Повече https://developer.mozilla.org/en-US/docs/Web/API/File .
- **FileList** – масив от файлове. Обект който се връща от файл свойството на input елемента. Повече https://developer.mozilla.org/en-US/docs/Web/API/FileList .
- **FileReader** – методи за прочитане на данни от **blob.** обект чрез който уеб приложенията асинхронно могат да четат съдържание от локален файл. Повече https://developer.mozilla.org/en-US/docs/Web/API/FileReader .

- Прочитане на текстов файл – посредством изпълнение на следните стъпки, чрез JavaScript:
  - Да се вземе **File** или **Blob** обект (<input type="field">) или чрез провлачване в браузъра (drag-and-drop)
  - Да се създаде **FileReader** обект и да се прихванат събитията **load**(събитие, което възниква при успешно зареждане на файла) и **error**(събитие, което възниква при проблем при зареждането)
  - Да се извика функцията **readAsText**() от създаденият **FileReader** обект

- Във събитието load да се достъпи до текстовото съдържание чрез **result** параметъра
- Във събитието error да се имплементира подходяща обработка за възможните грешки

- Прочитане на бинарен файл:
  - Да се вземе **File** или **Blob** обект (<input type="field">) или чрез drag-and-drop
  - Да се създаде **FileReader** обект и да се прихванат събитията **load** и **error**
  - Да се извика **readDataAsURL**() от **FileReader**
  - Във събитието load да се достъпи до текстовото съдържание чрез **result** параметъра
  - Във събитието error да се имплементира подходяща обработка за възможните грешки

Drag-and-drop (провлачване)

Функционалност от HTML5 стандарта. Позволява преместването на обект от едно място на страницата до друго, посредством click, drag и drop. Всеки видим HTML елемент може да бъде позволен за провлачване.

Първото което трябва да направим е да добавим HTML атрибута "draggable" със стойност "true", за елемента който искаме да преместим. След това трябва да зададен какво ще се случи при когато елемента се провлачва. Това можем да направим с атрибута "ondragstart", който трябва да извиква функция описваща поведението на елемента при drag. В тази функция трябва да опишем типа на елемента (данните) и стойността и. Това става чрез метода "setData("type",value)". Следва определянето на поведението на елемента в който ще "пуснем" (drop) нашия елемент. Поведението на този елемент се определя от функция извикана от атрибута "ondragover". Функцията трябва да позволи на елемента да бъде поставен на елемент с дефинирана функция за "OnDragOver" събитието. Това става възможно с премахване на нормалното поведение на този елемент - използваме "event.preventDefault()" метода. Остана само да опишем поведението на елемента в който поставяме, при самото поставяне (drop). Трябва да му добавим "ondrop" атрибут, извикващ функция, описваща това поведение:

```
<!DOCTYPE HTML>

<html>
```

```
<head>
<style>
#div1 {width:350px;height:70px;padding:10px;border:1px solid #aaaaaa;}
</style>
<script>
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag the W3Schools image into the rectangle:</p>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)" width="336"
height="69">
```

```
</body>
</html>
```

В този пример взет от w3schools.com, са показани всички стъпки.

HTML видео

Има няколко варианта да добавим видеоклип към нашия сайт:

- Чрез <embed>

```
<embed src="intro.swf" height="200" width="200">
```

- Чрез <object>

```
<object data="intro.swf" height="200" width="200"></object>
```

- Чрез <video>

```
<video width="320" height="240" controls>
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
         Your browser does not support the video tag.
</video>
```

Елементът “video” специфицира, че добавяме видеоклип. Трябва да окажем източник на клипа, дори повече от един в случай че формата не се поддържа от браузъра. Това става чрез “source” елемента, както се вижда в примера. Съществуват три формата които се поддържат от повечето браузъри: MP4, WebM, Ogg.

За повече информация за поддръжката на различните аудио формати в различните видове и версии браузъри, също и всички атрибути които може да използваме при вмъкване на видео:

http://www.w3schools.com/tags/tag_video.asp

HTML аудио

Използва се “audio” елемента. Също е част от HTML5 стандарта. Чрез него можем да добавим звуков файл в нашата страница.

```
<audio controls>
        <source src="horse.ogg" type="audio/ogg">
```

```
        <source src="horse.mp3" type="audio/mpeg">
        Your browser does not support the audio element.
</audio>
```

За повече информация:

http://www.w3schools.com/html/html5_audio.asp

HTML5 Геолокация

Функционалност от HTML5 която се използва за установяване позицията на потребителя.

Изрично трябва да позволим установяването на нашата позиция в браузъра който ползваме, в противен случай няма да получим желания резултат.

Пример:

```
<script>
        var x = document.getElementById("demo");
        function getLocation() {
                if (navigator.geolocation) {
                    navigator.geolocation.getCurrentPosition(showPosition);
                } else {
                    x.innerHTML = "Geolocation is not supported by this browser.";
                }
        }
       function showPosition(position) {
            x.innerHTML = "Latitude: " + position.coords.latitude +
           "<br>Longitude: " + position.coords.longitude;
       }
</script>
```

Има няколко техники, които могат да бъдат използвани при определянето на местоположението:

- IP адрес
- GPS позиция
- Wi-Fi
- Клетките на мобилният оператор
- Задаване от потребителя

Модул 16: Какво представляват CMS системите

- Какво е CMS
- Видове CMS
- Популярни CMS
- CMS Hosting
- SharePoint 2013

Въведение в CMS

- Content Management System (Система за управление на съдържанието, CMS) е компютърна програма, която позволява публикуването и редактирането на съдържания, както и поддръжка на главен интерфейс.
- Улеснява изграждането на динамичен уебсайт
- Употреба
    - Блогове
    - Новинарски сайтове
    - Електронни магазини
- Намаляване на ръчно написаният код

Основната функция и използването на системи за управление на съдържанието е да се представи информация в уебсайтове. Има изключително голям избор от такива системи и всяка една от тях е различна сама по себе си. Едни от тях предоставят само необходимите за обикновения потребител функции по лесен и достъпен начин, а други предоставят много възможности за манипулация на данните и с тях може да се прави абсолютно всичко. Повечето системи за управление включват публикуване, форматиране, преглед, индексиране, търсене и извличане на текстове. Системата може да се използва за централно хранилище, съдържащо документи, филми, снимки, телефонни номера, научни данни и др. Тя може да се използва за съхранение, контролиране, преразглеждане, семантично обогатяване и публикуване на документацията.

Основно разделение на видовете CMS

- Разграничават се по функционалност и по леснота на употреба

- CMS, които дават възможност да се прави абсолютно всичко със съдържанието, за сметка на трудността на работа и поддръжка
- Лесни и интуитивни CMS – те съдържат само най-необходимото

- Избор в зависимост от:
  - Мащаба на проекта
  - Време и бюджет за реализация на проекта

В днешно време уеб разработчиците разполагат с богато разнообразие от системи и технологии, които могат да им помогнат при разработката на интернет приложение. Някои от тях разполагат с разширена функционалност, докато други са разработени за решаването на конкретни проблеми. Когато трябва да се вземе решение коя технология да се използва при разработка, задължително трябва да се вземе предвид доколко тази технология е лесна за използване или научаване и дали функционалността, която предлага не надхвърля значително нуждите на проекта. Избор на многофункционална технология може да забави изпълнението на проекта.

Основни характеристики

- Управление на съдържанието на уеб сайтове
- Функционалност
  - Публикуване
  - Форматиране
  - Преглед
  - Индексиране
  - Хранилище на данни
    - Документи
    - Филми
    - Снимки
    - Телефонни номера

CMS служи за създаване и поддръжка на уеб сайтове. Те дават възможност за използване на предварително разработена функционалност, която е обща за голям набор от сайтове. В повечето случаи за даден сайт е нужно да може да бъде добавяно и обновявано

съдържанието от потребители, които не са участвали в разработката. Те трябва да разполагат с инструменти за преглед на съдържанието и режим за редактиране. Тъй като често се налага качване на файлово съдържание в интернет, един от основните инструменти на CMS е възможността за споделяне и управление на документите. Друго предимство, което CMS в повечето случаи осигуряват е възможността за използването на предварително подготвен дизайн.

Уеб система за управление на съдържанието

- Пакет или самостоятелно приложение за управление на съдържанието на уеб страниците
    - Създаване
    - Съхраняване
- Уеб съдържание
    - Събиране и индексиране на съдържание
    - Чужди езици
    - Контрол на HTML съдържанието

Уеб системата за управление на съдържанията е пакет или самостоятелно приложение за създаване, управление, съхраняване и използване на съдържанията в уеб страниците. Уеб съдържанието включва текст и вградени графики, снимки, видео, аудио файлове, както и програмен код. Уеб системата може да събира и индексира съдържания, избира и сглобява съдържания по време на работа или да доставя съдържания за определени потребители по определен начин, като например друг език. Този вид системи обикновено позволяват на клиента да контролира HTML-базирани съдържания, файлове документи и уеб хостинг планове, въз основа на системата и дълбочината на нишата, която тя обслужва.

Компонентна система за управление на съдържанието

- Специализирани в създаването на документи от съставни части
- Системата Component CMS (Компонентна CMS, CCMS. Разделя информацията на гранулярно ниво, вместо на документи. Отделни заглавия, изображения и съдържание се разделят на единици информация)
    - Използва DITA XML (Darwin Information Typing Architecture eXtensible Markup Language)
    - Потребителите сглобяват индивидуални теми в една карта
    - Компонентите могат да се използват с други документи

- Не са подходящи за големи компании

Компонентната система за управление на съдържанието е специализирана в създаването на документи от съставни части. Като например CCMS, която използва DITA XML позволява на потребителите да сглобяват индивидуални теми в една карта (документ). Тези компоненти могат да се преизползват с друг документ или с друго множество такива. Това гарантира, че съдържанието е наблюдавано и в целия набор документация. Все пак този вид система не е подходяща за големи организации, защото те имат възможност да си поръчат система, отговаряща на техните изисквания.

Повече информация за CCMS и DITA:

1. https://en.wikipedia.org/wiki/Component_content_management_system
2. https://en.wikipedia.org/wiki/Darwin_Information_Typing_Architecture

Joomla

- Безплатна, написана на PHP (скриптов език за уеб програмиране)
- Възможности
    - Кеширане на страници
    - Блогове
    - Анкети
    - Търсене
    - Езикова локализация
- Преведена на 60 езика, сред които и български
- Разширения
    - Компоненти
    - Плъгини (допълнителни приложения, които разширяват функционалността)
    - Шаблони
    - Модули
    - Езици

Joomla! е безплатна система за управление на съдържанието с отворен код, написана на PHP, за публикуване на уеб съдържание. Използва база данни MySQL и техники на Обектно-ориентирано програмиране ООП. Joomla включва/съдържа възможности като кеширане на страници (page caching) за подобряване на изпълнението, Rich Site Summary(подробно резюме на

сайта) четци (на англ. RSS feeds), версии на страниците за печат, предаване на кратки новини (т.нар."news flashes"), блогове, анкети (web polls), уеб сайт търсене и езикова локализация.

Шаблоните за Joomla са многостранни разширения, които са отговорни за оформлението, дизайна и структурата на задвижваните от Joomla сайтове. Докато CMS управлява съдържанието, шаблоните управляват външния вид, усещането за елементите на това съдържание и цялостния дизайн на задвижваните от Joomla сайтове. Съдържанието и дизайна в шаблоните на Joomla са отделни и могат да се редактират, променят и изтриват поотделно. Шаблонът е мястото, където е зададен дизайнът на позициониране на основните елементи за сайт на Joomla. Това включва, мястото където потребителите поставите различни елементи (компоненти, модули и допълнителни приложения), които са отговорни за различните видове съдържание. Ако шаблонът може да бъде персонализиран от потребителя, потребителят може да промени разположението на съдържанието в сайта, например, да постави главното меню от дясната или лявата страна на екрана.

За дапълнителна информация: http://joomla.bg/

PrestaShop - CMS за електронна търговия

- Локализация
    - Език
    - Валути
- SEO оптимизация
- Висока защита
- Статистики
- Разплащане
    - Paypal – популярна система за електронно разплащане
    - Moneybookers – друга система за финансов трансфер в/чрез интернет
- Каталог
- Клиенти

Prestashop е безплатна платформа за онлайн магазин. Базирана е на PHP и MySQL (или друг тип система за бази данни), като с нея лесно можете да изработите онлайн магазин сами. Ще са ви нужни елементарни познания по File Traqnsfer Protocol (протокол за обмен на файлове по мрежа, FTP), MySQL и поне някога да сте работили с блог или форум.

Добавянето на различни допълнителни възможности към магазина става с т.нар. модули. Има, както безплатни, така и платени модули. За България, модулите, които са най-необходими са безплатния Prestashop модул за доставка на Еконт експрес и Prestashop

модула за разплащания през ePay.bg. Prestashop разполага с множество безплатни модули за разплащания, като един от най-популярните е модула за разплащания през PayPal. Модулът за разплащния през ePay.bg, обаче се закупува допълнително.

Предимствата на платформата за онлайн магазин Prestashop, е че е с много повече възможности от почти всички платформи, които се предлагат на българския пазар. В сравнение с българските платформи за онлайн магазини, като SummerCart, Prestashop освен, че е безплатна разплага с огромен набор от статистики, най-добри практики за SEO (Search Engine Optimisation – практики за изграждане на електронно съдържание, така че то да бъде по-лесно обработвано от търсачките), менажиране на потребители, служители, стилове и модули за функционалност и т.н. Почти всеки месец излиза нова версия, като обновяването на старите версии е напълно безплатно. За разлика от комерсиалните решения, всякакви рестрикции за лицензиране на магазина за точно определен домейн отсъстват. Работата на цяло общество от специалисти и отворения код на основната платформа превръщат Prestashop в една от водещите платформи за електронна търговия в световен мащаб.

Сайтът по подразбиране при инсталацията на PrestaShop излежда по следният начин:

CMS Hosting – физическото местоположение на сайтовете изградени чрез CMS

- CMS се различават по
    - вид и технология на изграждане

    - фунционалностти който предлагат
- Бази данни
- Програмна структура
- Hosting в зависимост от изискванията на системата
    - Нормални динамични сайтове

В зависимост от технологията, която е избрана при разработване на приложението трябва да се избере и подходящ доставчик на хостинг услуги(съхранение на уеб съдържанието). Изключително важно е доставчика да поддържа сървърната технология, върху която е изграден сайта. По-известните технологии и CMS се поддържат от повечето провайдери.

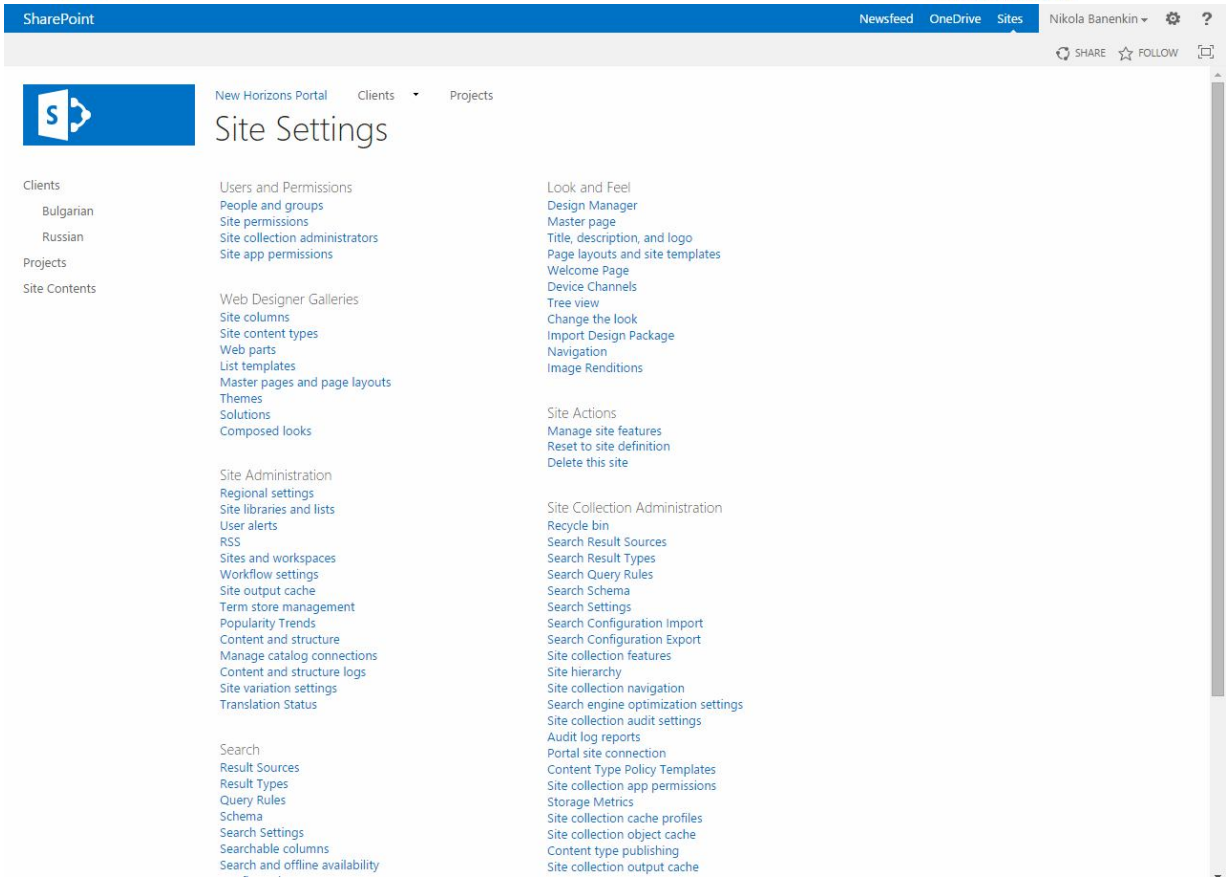
Полезни насоки при избор на хостинг провайдер:

http://vhod2.com/novina-9_vajni_faktora_pri_izbor_na_web_hosting.html

Какво е SharePoint

- Платформа за създаване и управление на уеб съдържание – сайтове, библиотеки с документи, записи и др.
- Web Application Framework (Библиотека, която е проектирана да подпомага разработването на динамични уебсайтове, WAF)
- Първата версия е интегрирала:
    - Intranet (отделена частна мрежа)
    - Content Management (управление на съдържание)
    - Document Management (управление на документи)
    - Уеб базирана конфигурация
- PowerShell (система за автоматизация на управлението на задачите и настройване на продуктите на Microsoft)

Организациите използват SharePoint за създаване на уеб сайтове. Можете да го използвате като сигурно място за съхранение, организиране, споделяне и достъп до информация от почти всяко устройство. Всичко, от което имате нужда, е уеб браузър, като напр. Internet Explorer, Chrome или Firefox. Това е технология с много възможности – използва се за разработване на вътрешни мрежи, споделяне на документи или фирмени портали.



Изграждане на CMS

Има три начина, по които можете да започнете CMS проект:

1) Стратегически подход, насочен върху стойността – Организациите могат да изберат да започнат с този подход, когато те смятат, че той е най-разбран за техните бизнес изисквания и приоритети.

Един от начините да използват този подход е да документират бизнес и техническите услуги, които са доставени в каталога за услугите. Определени услуги, взаимоотношения и композиции могат да бъдат определени и документирани в CMS. След като приключат, информацията за тази услуга може да се използва, за да се приведе в съответствие с бизнеса и да бъде основа за структурата на CMS.

2) Функционално фокусиран подход – Компаниите могат да изберат да използват този подход, когато те вече са закупили инструменти и технологии за прилагането, откриването и картографирането на услугите и инфраструктурата.

Тези инструменти се използват, за да събират и предоставят списък на приложения и техните взаимоотношения. След като първоначалният списък е пълен, приложенията може да бъдат съпоставени с услугите, включително със състава на всяка услуга.

Този подход създава основа за документиране и определяне на услуги, тъй като отношенията са ключов компонент, събрани по време на откриването. Подобно на стратегическия подход за създаване на стойност, техническия каталог на услугите може да бъде създаден с помощта на информация за прилагане на връзките.

3) Подход, фокусиран върху технологията и стратегията – организациите обикновено започват с този подход, когато те имат достъп и могат да използват съществуващите инструменти за мониторинг и управление, с които колекционират физически и логически конфигурации и инфраструктура като съществуващи сървъри, маршрутизатори и комутатори, инсталирани в цялата заобикаляща среда.

Тази информация се използва като отправна точка за изграждане на CMS и улеснява създаването на основните елементи на конфигурацията, които могат да бъдат използвани за връзка между приложенията и определенията на услугите.

Повечето организации имат различни инструменти, използвани за управление на информационната инфраструктура и тези бази данни се считат за отделни бази данни за управление на конфигурацията (CMDB – Configuration Management DataBases), които могат да бъдат използвани като основополагаща информация, за да се интегрира CMS.

CMS шаблони и допълнителни приложения

Шаблоните за CMS системите представляват допълнително разработен дизайн, върху който може да се разработи портала и да се добави информацията. Тъй като тези шаблони в повечето случаи се продават на повече от един клиент след като бъдат разработени, за не голяма сума може да се намери професионален дизайн. Могат да се намерят и безплатни шаблони.

Шаблоните могат да бъдат:

Front-End – чрез тях се променя изгледа на сайта за потребителите

Back-End – чрез тях се променя изгледа на административният панел. Back-End шаблоните се използват много по-рядко.

CMS плъгините представляват функционалност, която лесно може да се интегрира в портала, който се разработва. Чрез плъгин може да се реализират например: разплащане през epay, компоненти за коментари, филтри за информация. Общо взето всичко, което не е предоставено от конкретната CMS система. В зависимост от това на каква CMS система ще бъдат инсталирани, плъгините могат да бъдат разработени чрез различни технологии.

Плъгини за WordPress:

https://www.ait-themes.club/wordpress-plugins-bg/

Възможно е съществуващи плъгини или шаблони да бъдат развити и дообработени според различни изисквания, за да се доразвие вече съществуваща система.

Редактиране и управление на съдържание

В повечето случаи, когато се използва CMS система, тя е изградена от две основни съставни части – потребителски сайт и административен панел. Възможно е съдържанието да бъде променяно директно през потребителският сайт – когато има реализиран форум или блог с коментари.

По-голямата част от съдържанието обаче се контролира от административният панел. През него могат да се създават потребители със съответните права, да се добавят статии или продукти в зависимост към какво е насочен порталът, който се разработва. Административните панели често разполагат и с диаграми, които визуализират статистики свързани с поведението на потребителите в уеб портала. На базата на тези статистики може да се вземе решение за промяна на съществуващата информация или на структурата и навигацията в портала.

SharePoint продукти

- SharePoint Online
- SharePoint Foundation
- SharePoint Server
- OneDrive for Business
- SharePoint Designer

SharePoint Online Базирана на облак услуга, доставяна от Microsoft, за фирми от всякакъв калибър. Вместо да инсталира и разполага локално SharePoint Server, всяка фирма може да се абонира за план на Office 365 или за отделната услуга на SharePoint Online. Вашите служители ще могат да създават сайтове за споделяне на документи и информация с колеги, партньори и клиенти.

SharePoint Foundation Базовата технология за всички сайтове на SharePoint. SharePoint Foundation (преди Windows SharePoint Services) е безплатна за локално разполагане. Можете да използвате SharePoint Foundation за създаването на много типове

сайтове, където можете да работите съвместно по уеб страници, документи, списъци, календари и данни.

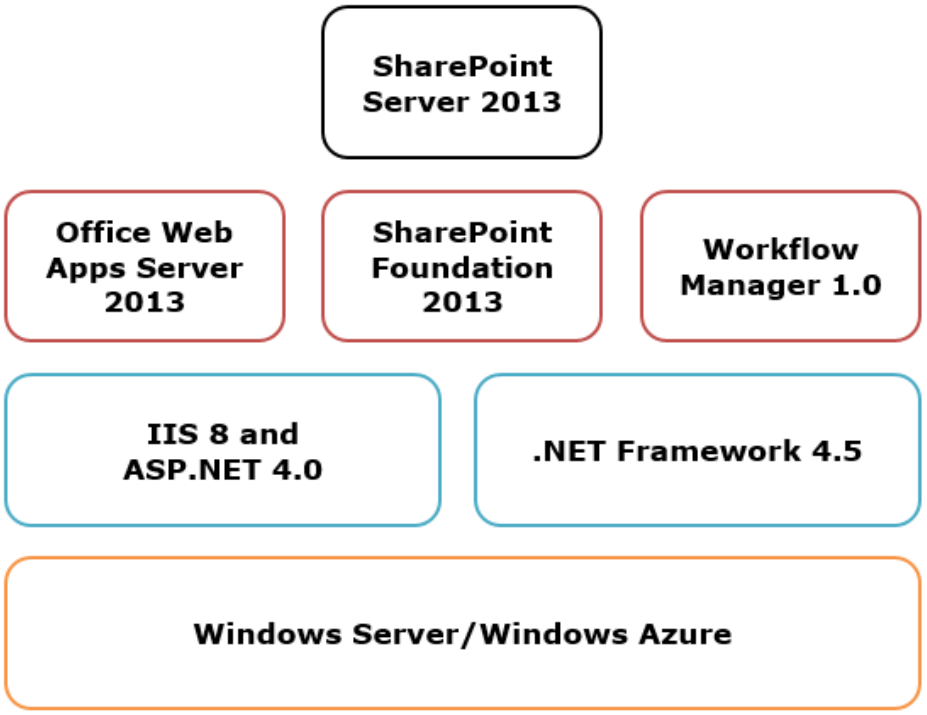
SharePoint Server Организациите могат локално да разполагат и управляват SharePoint Server. Той включва всички функции на SharePoint Foundation и предлага допълнителни функции и възможности, като управление на корпоративно съдържание, бизнес разузнаване, корпоративно търсене, персонални сайтове и информационни канали.

SharePoint Designer 2013 Безплатна програма. Използвайте я за създаване на мощни решения за работни потоци и редактиране на външен тип съдържание за решение за външни данни въз основа на услугите за бизнес свързване.

Синхронизиране на папки на OneDrive за бизнеса Настолна програма, която можете да използвате за синхронизиране на офлайн версия на екипен сайт или библиотека на OneDrive за бизнеса с папка на компютъра..

Технологии в SharePoint

SharePoint се базира изцяло на технологии, разработени от Microsoft. Инсталацията на SharePoint Server изисква работеща платформа – Microsoft Azure или PC с операционна система Windows Server. Използват се още компоненти и части от функционалността на .NET Framework 4.5 (платформа за разработка на приложения на Microsoft), IIS(Internet Information Services – сървърно приложение на Microsoft за предоставяне на информация по мрежа, което работи с повечето популярни протоколи) и ASP.NET (Active Server Pages – технология на Microsoft за разработване на уеб приложения)



Възможности на SharePoint

- Колаборация (съвместна работа)
- Enterprise Content Management (управление на документите в големи организации, ECM)
- Web Content Management (управление на съдържание в интернет, WCM)
- Social computing (функционалности на социалните мрежи)
- Search (Търсене в базата от данни)
- Business Intelligence (управление и използване на големи бази данни със статистическа информация, BI)

Използвайки SharePoint Server вие можете да изградите web базирана среда със следните възможности:

- Всеобщо достъпен Интернет сайт
- Интранет сайт за организация и за всеки един от отделите и.
- Екстранет сайт за вашите клиенти или партньори
- Общ сайт за вашият отдел Продажби
- Проектен сайт за вашият екип разработчици
- Система за управление на документите, която е съвместима с ISO-9000.
- Лична страница за всеки един потребител, където те могат да съхраняват информация и изграждат линкове към техните екипни сайтове
- Цифрово хранилище за съхраняване на бизнес чувствителна информация. Например индикатори за определена възвръщаемост
- Място за търсене и намиране на всеки тип информация независимо къде тя реално се намира
- Хранилище за всякаква правна информация гарантираща нейната сигурност

Списъка може да расте все повече и повече тъй като SharePoint е доста гъвкаво приложение и само нашето въображение би ни ограничило за неговото приложение. Също така работата с него е изключително забавна и впечатляваща понеже в лесни стъпки можем да изградим доста сериозни сайтове.

Модул 17: Защо имаме нужда от CMS

- Цели на уеб портала
    - Информацията да се актуализира лесно

- Актуалната информация да е достъпна за потребителите

- Разнообразно съдържание в портала
  - Документи
  - Изображения
  - Streaming Media (управление на мултимедия в реално време)
  - Съобщения
  - Електронна поща

Целта на уеб сайта на дадена компания е да осигури актуална информация и съдържание за широка публика, с възможност за бърза промяна и добавяне на информацията, за да могат да се отразят своевременно новите продукти и новините около компанията. Бизнесът се променя бързо, появяват се нови промоции и се променят предлаганите продукти. Успешният уеб сайт гарантира, че с потребителите се споделя точната информация. Променя се не само големината на предлаганата информация, но и типа на съдържанието – като документи, снимки, съобщения и т.н. За да може разработчика да се справи максимално бързо и лесно с тези предизвикателства той трябва да използва CMS. Чрез CMS се улеснява изграждането на уеб приложения, които използват голям набор от информация и функционалност.

Статични уеб сайтове

- Не са приложими за бизнеса
- Информацията се обновява сложно
- Навигацията не е достатъчно изчистена

До преди няколко години беше масова практика да се изработват статични уеб сайтове. Специфичното за тези сайтове е, че няма страница или модул за администриране, от където собственикът на сайта може да променя съдържанието му.

Статичният сайт се състои от електронни страници, реализиране чрез Hyper Text Markup Language (език за описание и дизайн на уеб страници, HTML) и Cascading Style Sheets (скриптов език за описание на стилове, CSS). В някои случаи се добавя и JAVA(обектно-ориентиран език, съвкупност от технологии за изграждане на приложения) или Java Script технология. При избора на статичен сайт е важно да сте предвидили малко на брой страници, в които да поместите информацията за фирмата си, защото поддръжката на сайта може да се извършва само от специалист дизайнер, и това генерира допълнителни разходи за Вас. Ако сайтът Ви ще съдържа голямо количество информация, която ще се променя често, е по-добре да поръчате динамичен уеб сайт.

Предимствата на статичния сайт са пълна свобода при реализиране на дизайна на сайта и по-бързо зареждане на страниците в браузъра.

За съжаление, изборът на статичен сайт ограничава собственикът му в актуализирането на информация по сайта. За промяната дори на една буква, е необходима намеса на дизайнер или програмист. Това само по себе си не е рентабилно за клиента, защото генерира постоянно допълнителни разходи.

Предимства на статичния уеб сайт:

- Изработва се по-бързо и е значително по-евтин
- Дава по-голяма свобода при изработването на дизайн, съобразен с предпочитанията на клиента
- Всяка една от вътрешните му страници може да е с различен дизайн
- Зарежда се бързо и има по-ясна навигация

Недостатъци на статичния уеб сайт:

- Не могат да се създават нови страници
- Актуализацията и промените по него могат да бъдат направени само от уеб специалист
- При статичните уеб сайтове не могат да бъдат направени по-сложни фунционалности, като например филтрирания, сортирания и др.

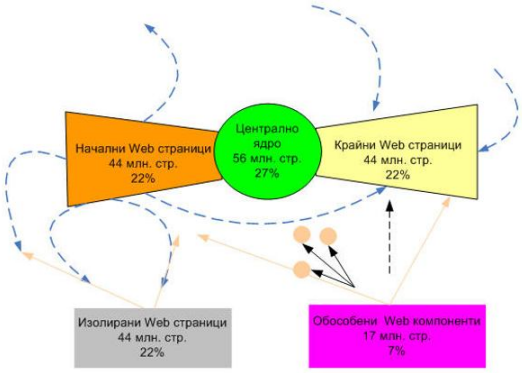
Основи на софтуерната архитектура

- Приложенията непрекъснато се развиват
- Колкото по-сложно е едно приложение, толкова по-важно е да се обърне внимание на архитектурата
- Въведени са много абстракции
    - Процедурни езици
    - ООП
    - Domain specific languages (компютърни езици, които са създадени за работа с конкретно приложение)
- Софтуерната архитектура въвежда абстракции от много високо ниво

Софтуерна архитектура представлява процеса за изработка на архитектурата на софтуера, при което се избират технологиите които ще се използват, стандарт за писане на код, инструменти и платформа. Разделят се сложните задачи на по-прости и лесни за изпълнение. Разделяне на компоненти и описание на тяхната функционалност.

Уеб архитектура

- Уеб архитектурата е необходима за:
    - Ефективна обработка на информацията от търсещите системи
    - По-добра ориентация на потребителите
- Карта на уеб пространството



През 2000 година компаниите AltaVista, IBM и Compaq, разработват математически модел на „карта" на Web пространството.

За разработка на модела са изследвани с помощта на търсачката AltaVista над 600 млн. Web-страници и 1,5 млрд. връзки, поместени в тези страници. Като резултат е определено, че топологията на Web-пространството съответства на модела „ папионка" (Bow Tie), състоящ се от следните компоненти:

Централно ядро, или възел на папионката - съставлява Web-страници, взаимосвързани, така тясно, че следвайки хипервръзките на всяка от тях в крайна сметка можеш да попаднеш на на всяка от тях (27%);

Начални Web-страници (22%), съдържащи хипервръзки, които водят към ядрото, но нямат обратна връзка от ядрото към тях.

Крайни Web-страници (22%) - до които може да се достигне с връзки от ядрото, но е невъзможно връщането назад;

Изолирани Web-страници (22%) от централното ядро. Това са свързани с хипервръзки страници от други категории или съединени страници не влизащи в ядрото;

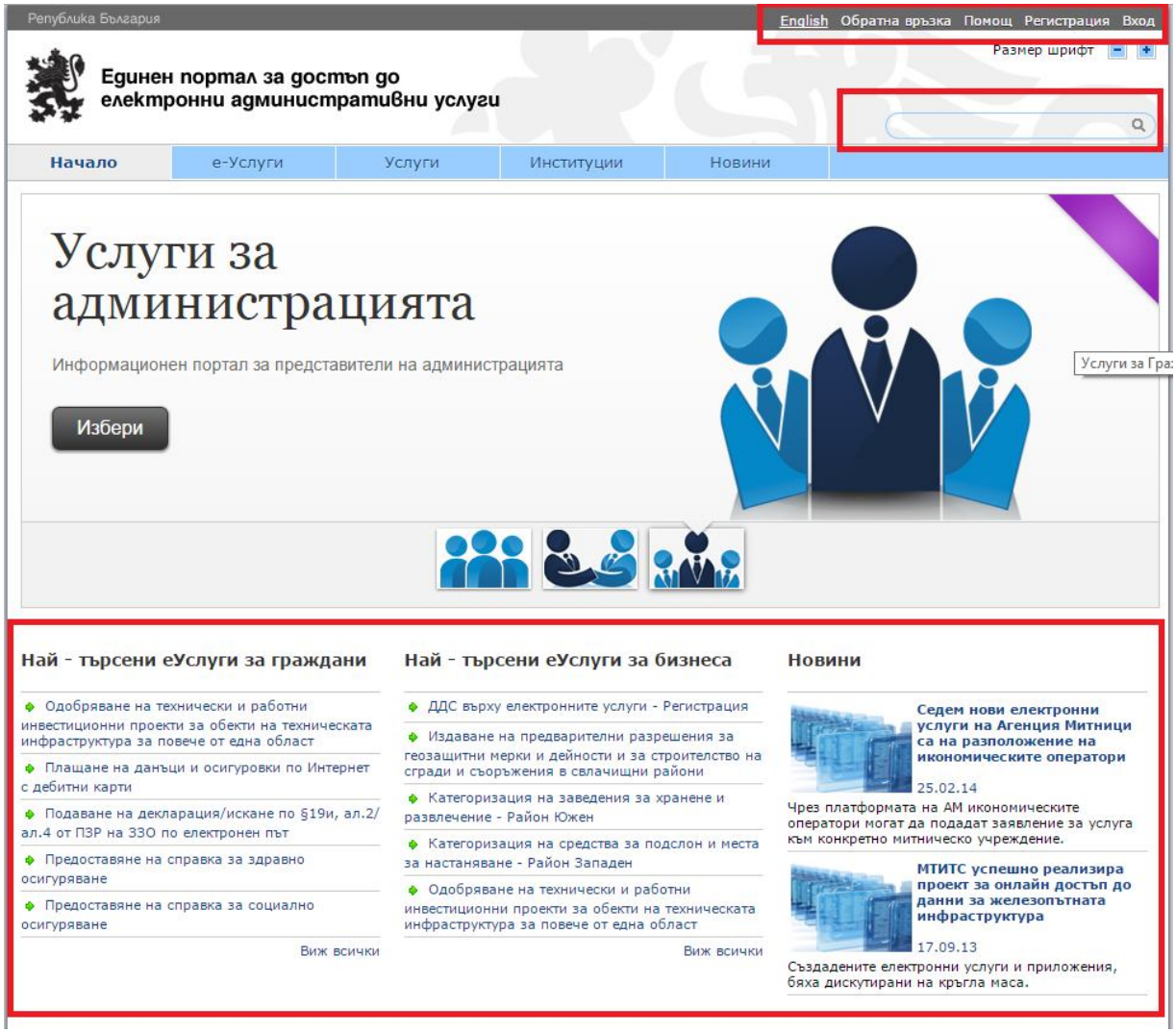
Изолирани острови Web-страници (7%), които не се пресичат с останалите ресурси на Интернет. Единствения начин да се стигне до тях е, ако е известен адресът им.

Сигурност, архитектура и мобилност на уеб портала

С нарастването на възможностите, които ни дава развитието на информационните технологии, сигурността, архитектурата и мобилността на уеб приложенията се превръщат във все по-значими въпроси. CMS системите успешно успяват да се справят с нивото на сигурност – разполагат със сравнително прости системи за определяне на правомощията на потребителите, а също и са сравнително добре защитени от хакерски атаки. С течение на времето сигурността става все по-добра.

От архитектурна гледна точка има известни ограничения, но вътрешната структура на приложенията, разработени чрез CMS е добре организирана. Прехвърляне на решението от една CMS на друга би бил трудоемък процес, но технологиите, на които се базират по-популярните CMS системи не са много. Това би могло да спомогне разработчиците на приложението бързо да се ориентират в новата система. От друга страна повечето CMS системи разполагат и с подобни модули.

Практически примери от правителствени уеб-портали:

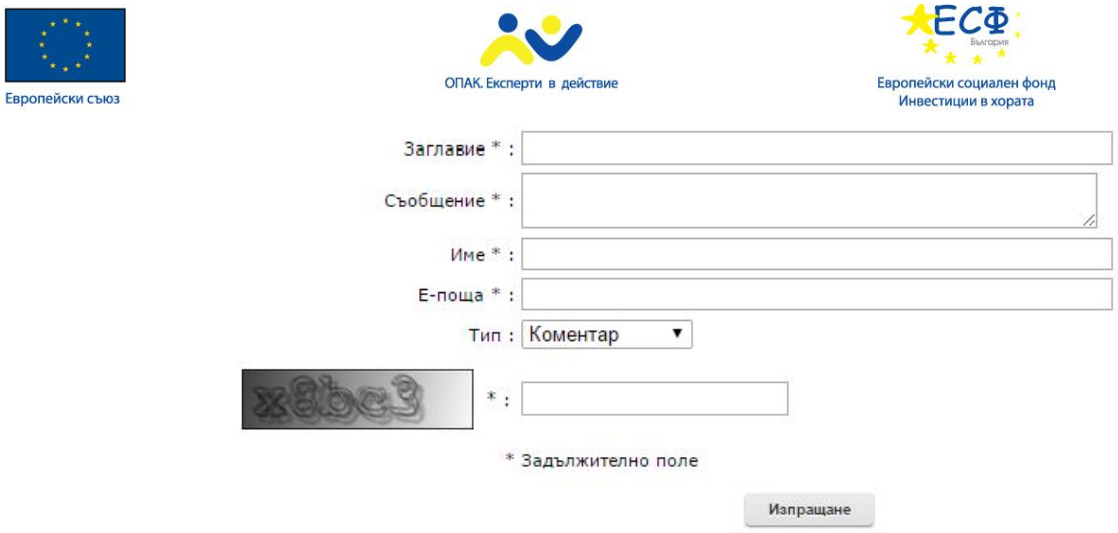


На сайта www.egov.bg се виждат някои от белезите характерни за динамичните уеб сайтове:

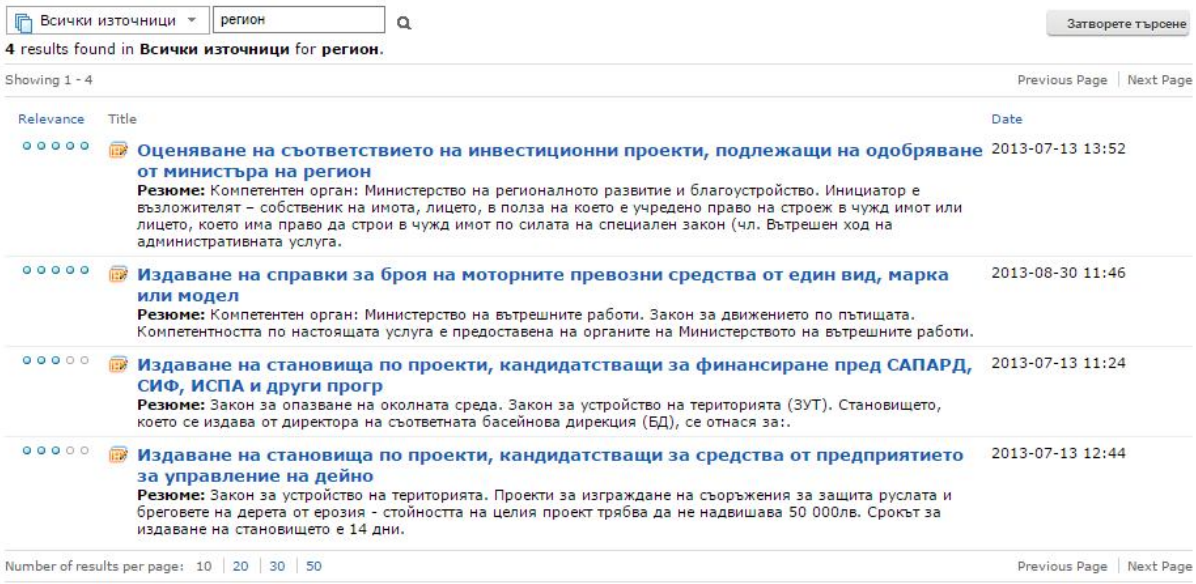
Възможност за регистрация и вход на потребител – чрез регистрирането на потребители е възможно да се ограничи предоставянето на определена информация на конкретният потребител или да се селектира информация, конкретно насочена към неговите интереси според профила му.

Възможност за промяна на език – на представеният сайт има възможност за смяна на езика на Английски и Български

Възможност за изпращане на обратна връзка – визуализира се форма за изпращане на информация към администраторите на съдържанието на сайта



Възможност за търсене в съдържанието на портала – чрез въвеждане на текст в полето за търсене извежда систематизирано резултати



Възможност за извеждане на динамична информация на главната страница – най-търсените услуги за гражданите и бизнеса и актуалните новини.

Изисквания за инсталация SharePoint 2013

- Сървър за бази данни:
    - Един от продуктите:
        - The 64-bit edition of Microsoft SQL Server 2012.
        - The 64-bit edition of SQL Server 2008 R2 Service Pack 1

- 64-bit edition на Windows Server 2008 R2 Service Pack 1 (SP1) Standard, Enterprise, or Datacenter or the 64-bit edition на Windows Server 2012 Standard или Datacenter
- Hotfix: ASP.NET (SharePoint) race condition in .NET 4.5 RTM:
  - Windows Server 2008 R2 SP1 (KB 2759112)
  - Windows Server 2012 (KB 2765317)
- Microsoft .NET Framework version 4.5

- За front-end web сървър или application сървър
  - The 64-bit edition на Windows Server 2008 R2 Service Pack 1 (SP1) Standard, Enterprise, или Datacenter or the 64-bit edition на Windows Server 2012 Standard or Datacenter.
  - Hotfix: ASP.NET (SharePoint) race condition in .NET 4.5 RTM:
    - Windows Server 2008 R2 SP1 (KB 2759112)
    - Windows Server 2012 (KB 2765317)
- Microsoft SharePoint Products Preparation Tool инсталира:
  - Web Server (IIS) role
  - Application Server role
  - Microsoft .NET Framework version 4.5
  - SQL Server 2008 R2 SP1 Native Client
  - Microsoft WCF Data Services 5.0
  - Microsoft Information Protection and Control Client (MSIPC)
  - Microsoft Sync Framework Runtime v1.0 SP1 (x64)
  - Windows Management Framework 3.0 which includes Windows PowerShell 3.0
  - Windows Identity Foundation (WIF) 1.0 and Microsoft Identity Extensions (previously named WIF 1.1)
  - Windows Server AppFabric
  - Cumulative Update Package 1 for Microsoft AppFabric 1.1 for Windows Server (KB 2671763)

Когато инсталирате SharePoint 2013 на отделен сървър можете да го конфигурирате така, че да изпълнява определена задача. След като е приключила инсталацията и изпълнението на SharePoint Products Configuration Wizard вие ще разполагате с инсталирани бинарни

файлове, конфигурирани настройки за сигурност, настройки на регистъра на Windows, настроена конфигурационна база от данни и подготвен уеб сайт на Централната администрация. След това може да се стартира приложението **Farm Configuration Wizard**, за да се настрои поведението на фермата, да се избере функционалността, която ще се използва и за да се направи първият Site collection(съвкупност от сайтове – основен градивен елемент от логическата архитектура на SharePoint. По-подробно описание в секцията „Логическа архитектура на SharePoint). Това приложение може да бъде стартирано и в бъдещ момент или настройките да се зададат ръчно.

Минимални хардуерни изисквания

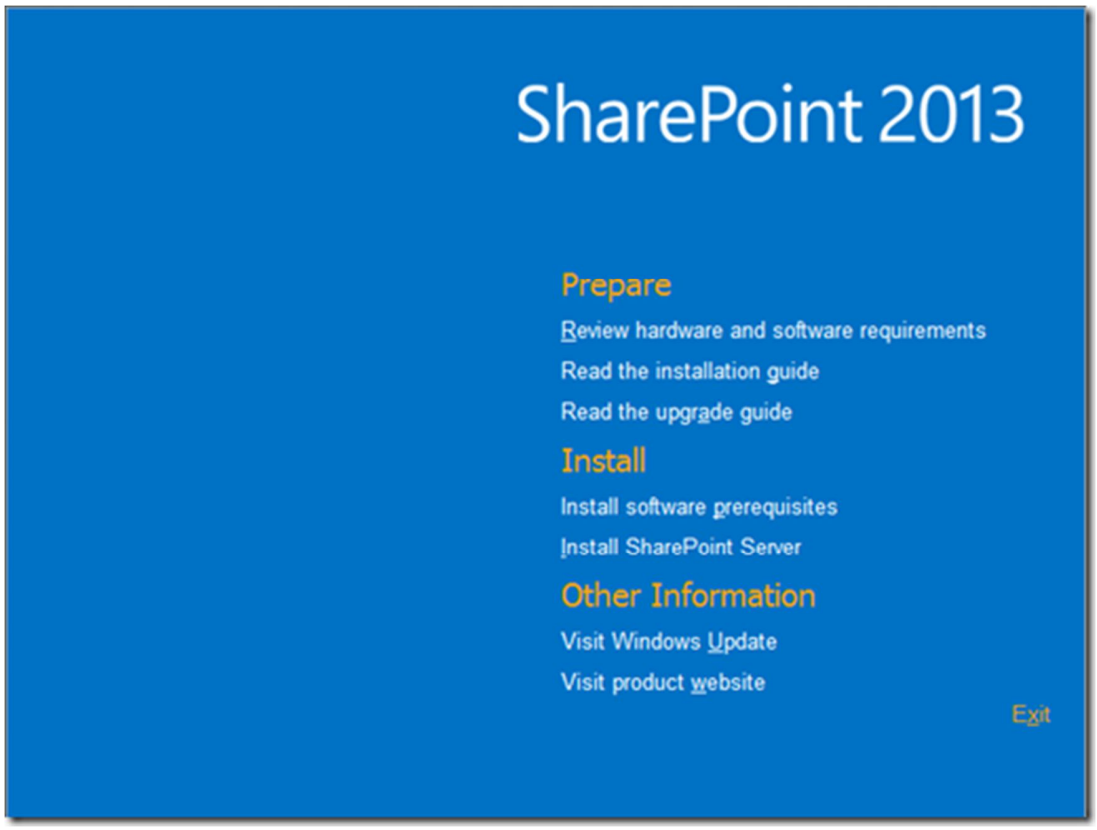
За сървър за база данни:

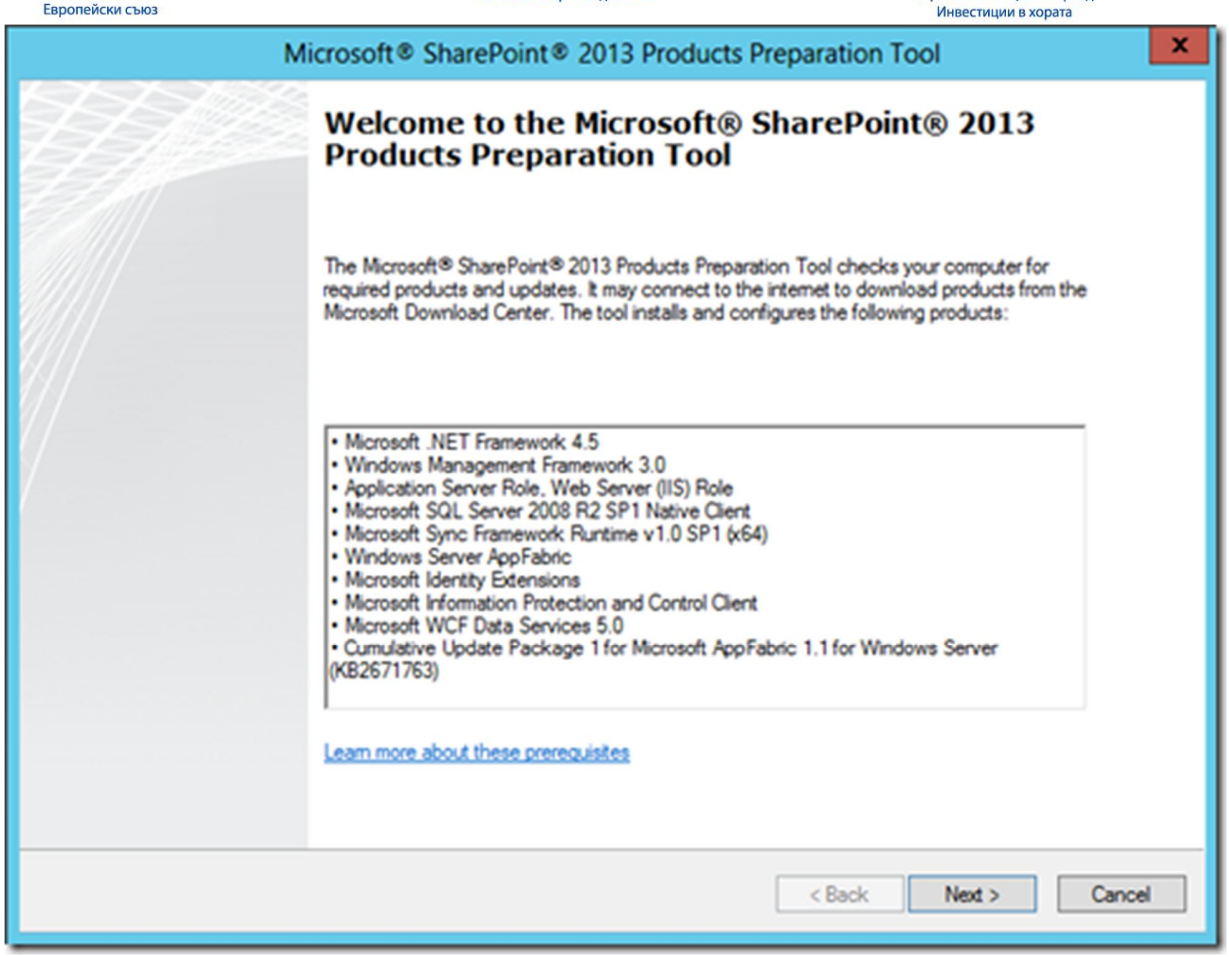
Компонент	Минимални изисквания
Процесор	64-bit, four-core, 2.5 GHz minimum per core (ако базата данни е много по-голяма от средното се препоръчват 8 ядра)
RAM	16 GB
Дисково пространство	80 GB

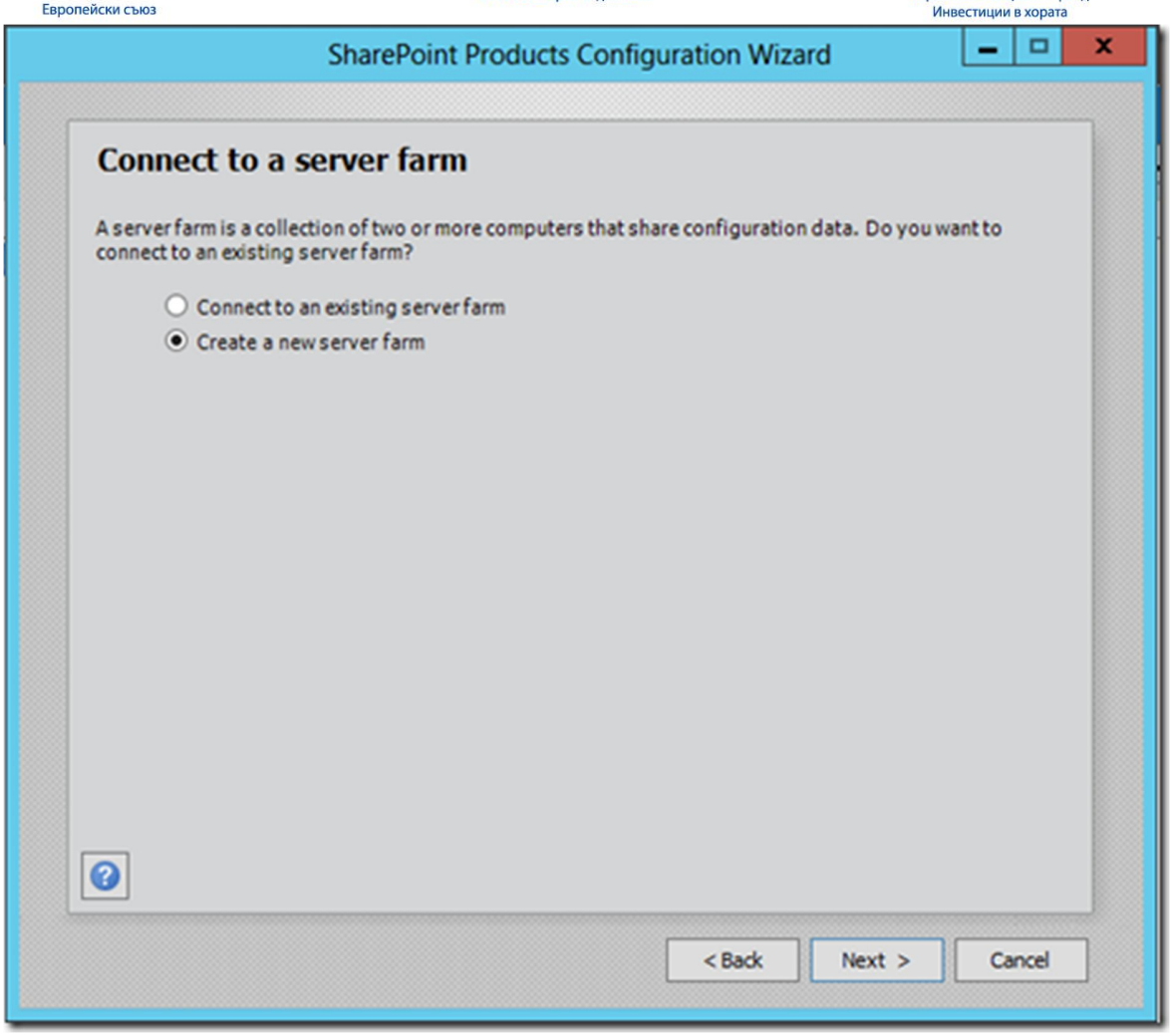
За web front end сървър:

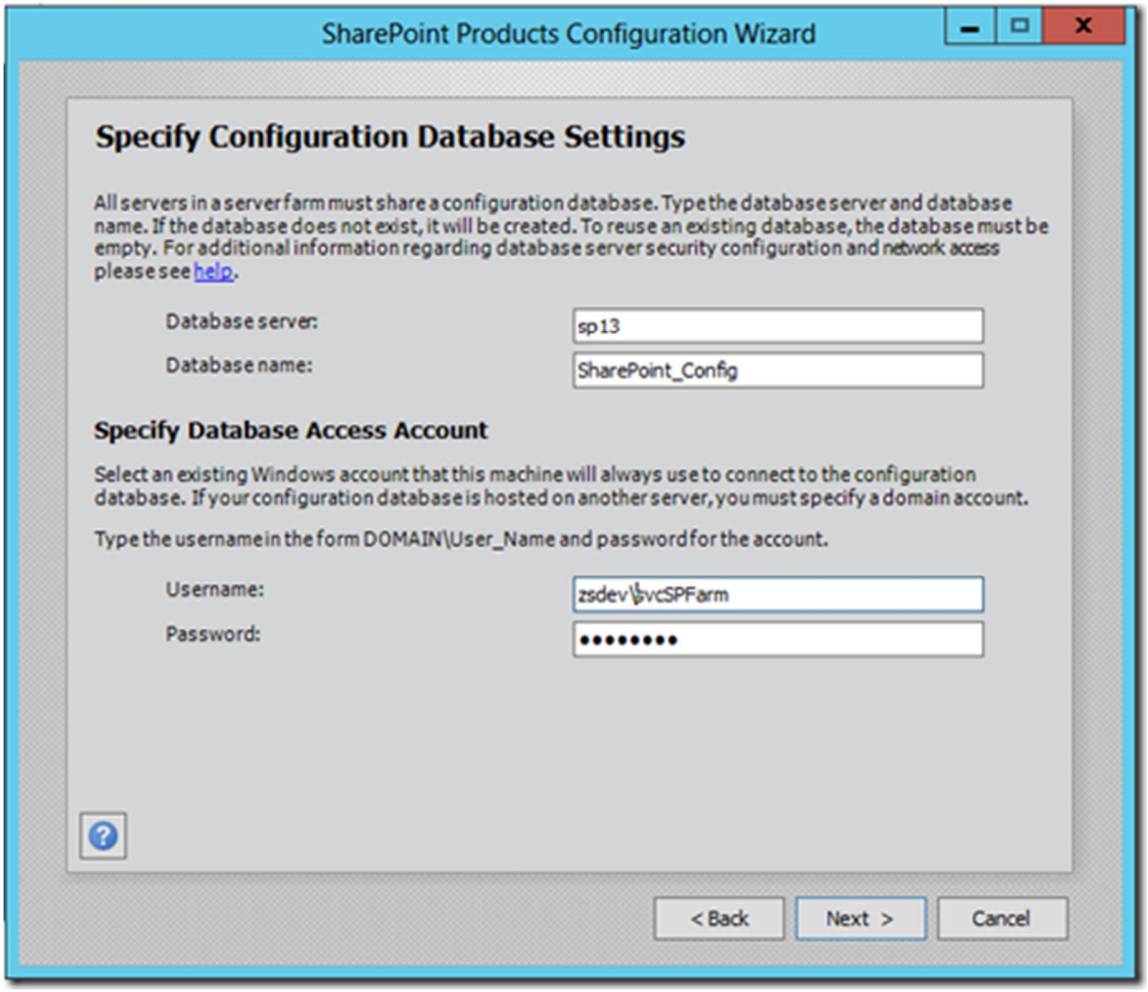
Компонент	Минимални изисквания
Процесор	64-bit, четриядрен, 2.5 GHz на ядро
RAM	16 GB
Дисково пространство	80 GB

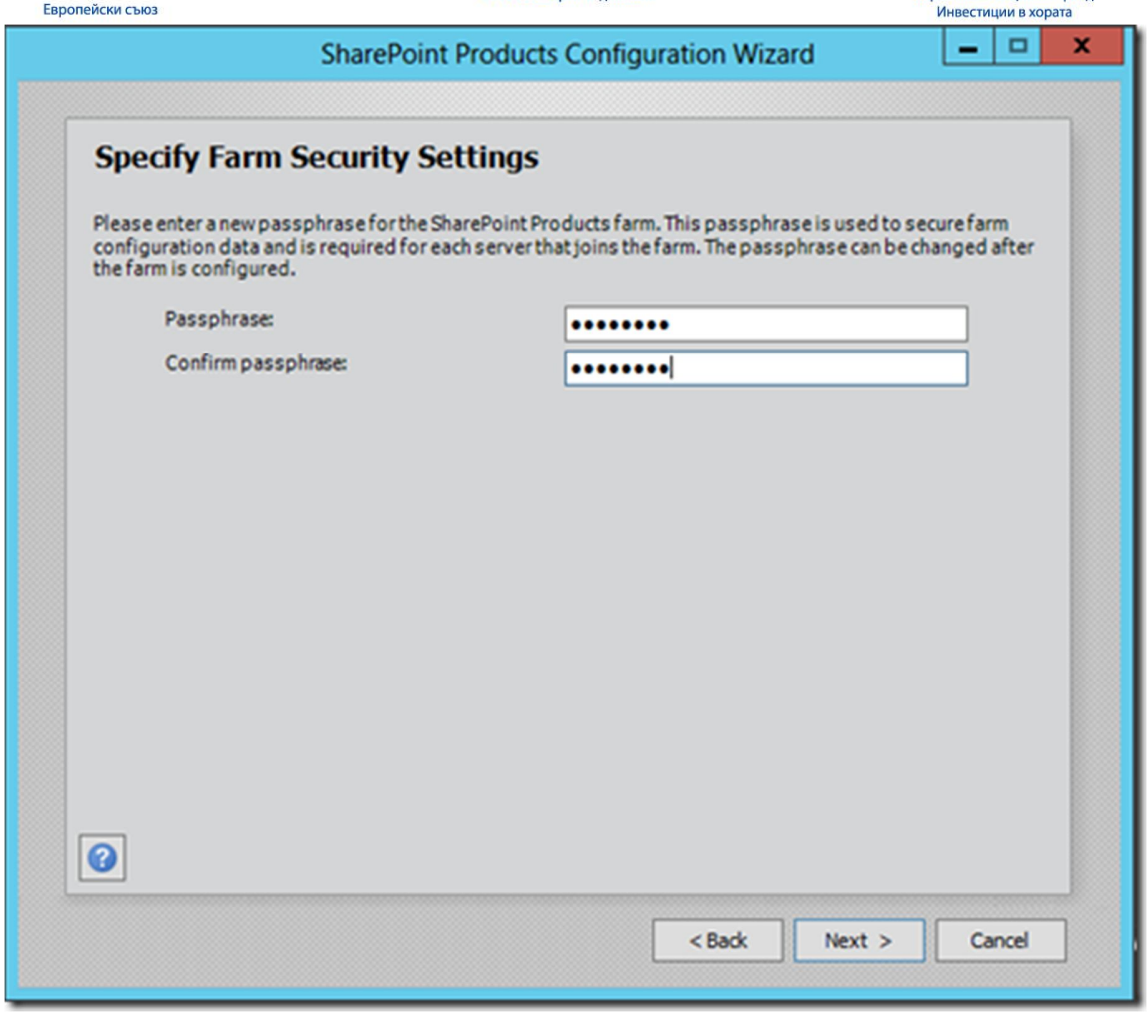
Инсталиране на SharePoint

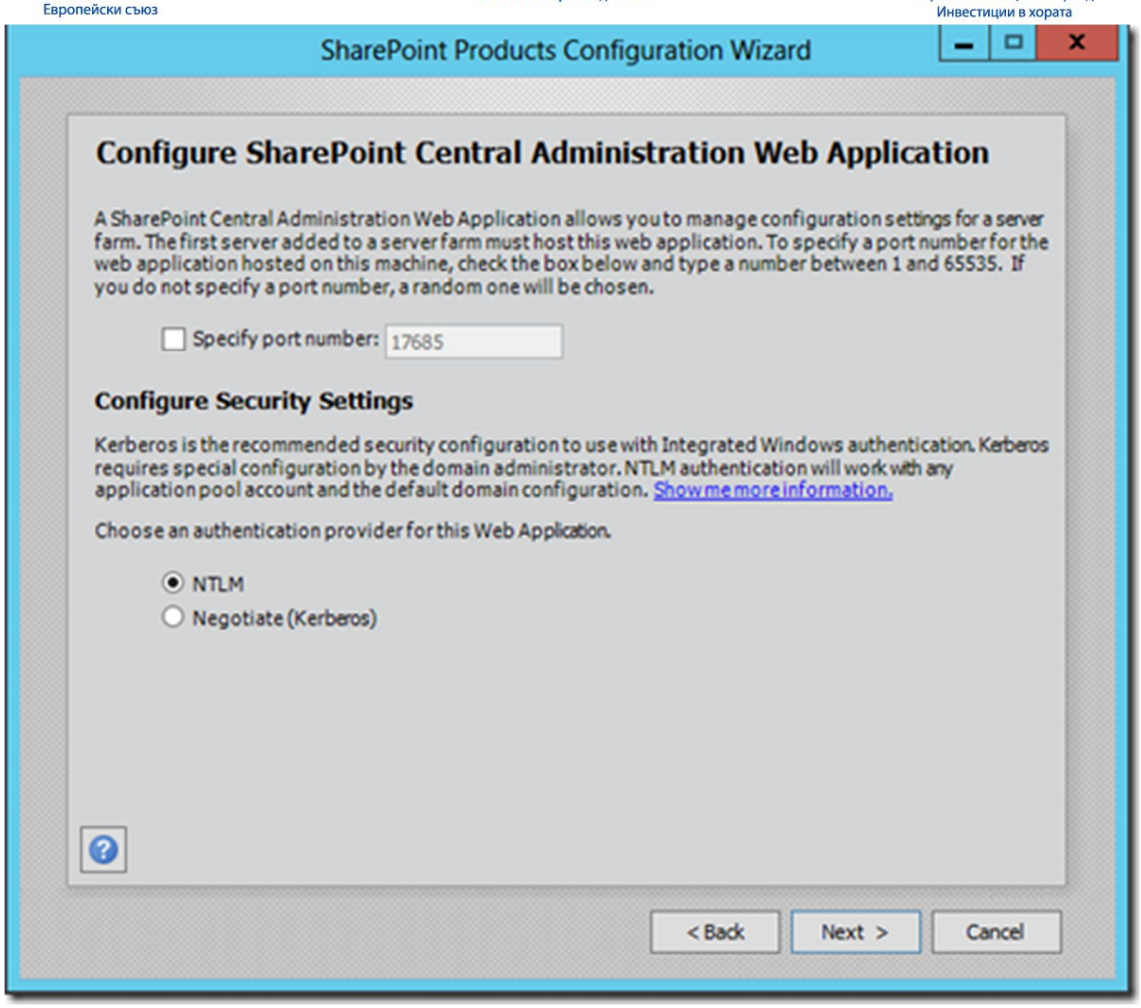


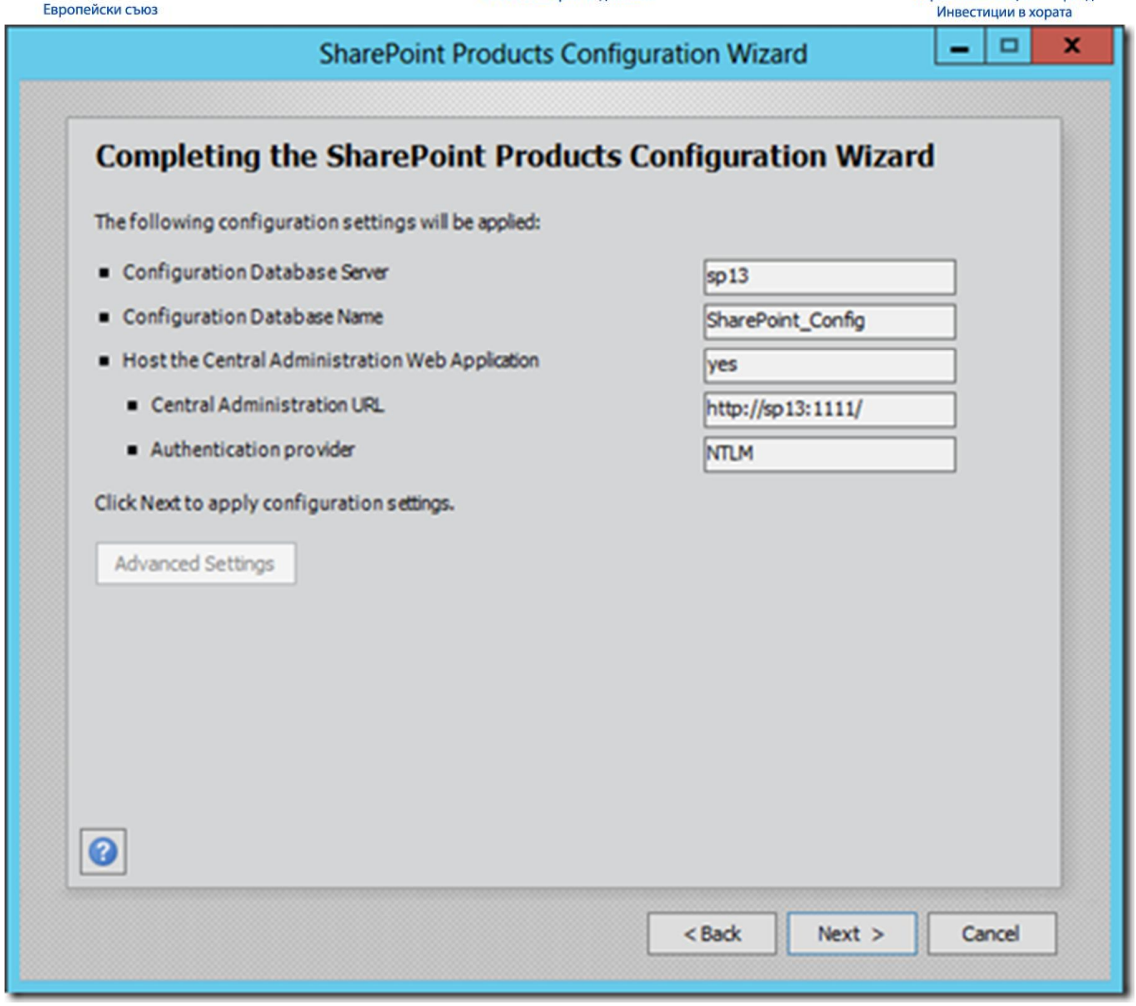

Microsoft® SharePoint® 2013 Products Preparation Tool
Welcome to the Microsoft® SharePoint® 2013 Products Preparation Tool
The Microsoft® SharePoint® 2013 Products Preparation Tool checks your computer for required products and updates. It may connect to the internet to download products from the Microsoft Download Center. The tool installs and configures the following products:
• Microsoft .NET Framework 4.5
• Windows Management Framework 3.0
• Application Server Role, Web Server (IIS) Role
• Microsoft SQL Server 2008 R2 SP1 Native Client
• Microsoft Sync Framework Runtime v1.0 SP1 (x64)
• Windows Server AppFabric
• Microsoft Identity Extensions
• Microsoft Information Protection and Control Client
• Microsoft WCF Data Services 5.0
• Cumulative Update Package 1 for Microsoft AppFabric 1.1 for Windows Server (KB2671763)
Learn more about these prerequisites
< Back
Next >
Cancel

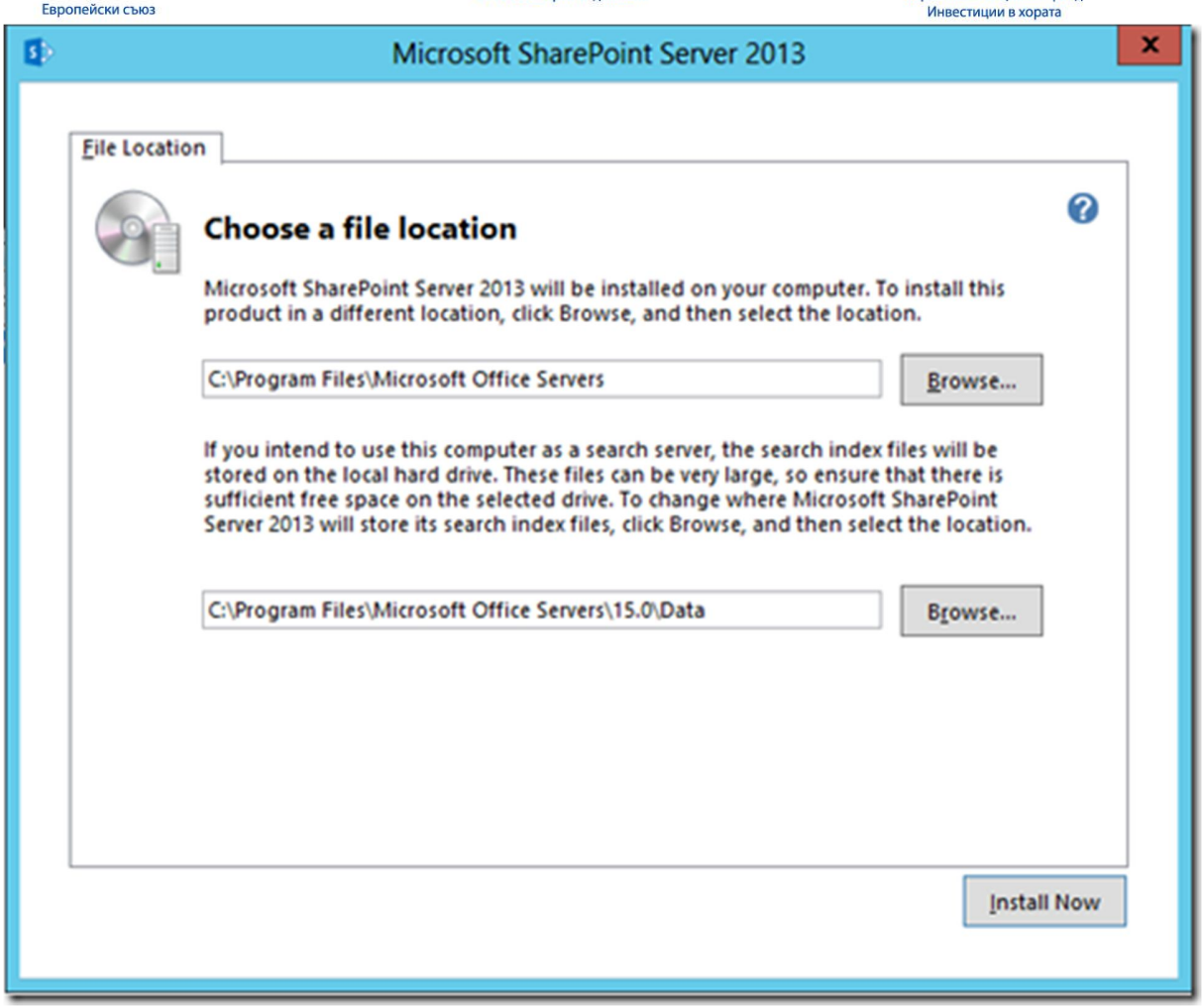

SharePoint Products Configuration Wizard
Connect to a server farm
A server farm is a collection of two or more computers that share configuration data. Do you want to connect to an existing server farm?
Connect to an existing server farm
Create a new server farm
< Back
Next >
Cancel


SharePoint Products Configuration Wizard
Specify Configuration Database Settings
All servers in a server farm must share a configuration database. Type the database server and database name. If the database does not exist, it will be created. To reuse an existing database, the database must be empty. For additional information regarding database server security configuration and network access please see help.
Database server:
sp13
Database name:
SharePoint_Config
Specify Database Access Account
Select an existing Windows account that this machine will always use to connect to the configuration database. If your configuration database is hosted on another server, you must specify a domain account.
Type the username in the form DOMAIN\User_Name and password for the account.
Username:
zsdev\svcSPFarm
Password:
< Back
Next >
Cancel


SharePoint Products Configuration Wizard
Specify Farm Security Settings
Please enter a new passphrase for the SharePoint Products farm. This passphrase is used to secure farm configuration data and is required for each server that joins the farm. The passphrase can be changed after the farm is configured.
Passphrase:
Confirm passphrase:
< Back
Next >
Cancel


SharePoint Products Configuration Wizard
Configure SharePoint Central Administration Web Application
A SharePoint Central Administration Web Application allows you to manage configuration settings for a server farm. The first server added to a server farm must host this web application. To specify a port number for the web application hosted on this machine, check the box below and type a number between 1 and 65535. If you do not specify a port number, a random one will be chosen.
Specify port number:
17685
Configure Security Settings
Kerberos is the recommended security configuration to use with Integrated Windows authentication. Kerberos requires special configuration by the domain administrator. NTLM authentication will work with any application pool account and the default domain configuration. Show me more information.
Choose an authentication provider for this Web Application.
NTLM
Negotiate (Kerberos)
< Back
Next >
Cancel


SharePoint Products Configuration Wizard
Completing the SharePoint Products Configuration Wizard
The following configuration settings will be applied:
Configuration Database Server
sp13
Configuration Database Name
SharePoint_Config
Host the Central Administration Web Application
yes
Central Administration URL
http://sp13:1111/
Authentication provider
NTLM
Click Next to apply configuration settings.
Advanced Settings
< Back
Next >
Cancel



Логическа архитектура на SharePoint

Много от софтуерните архитекти се съсредоточават върху физическата хардуерна архитектура и не обръщат необходимото внимание на логическата архитектура. Тя се занимава с документирането на структурата на решението, което е изградено за да покрие определени изисквания. Логическата архитектура задава изисквания за отделяне на частите на проекта в отделни структури, например информация за различните департаменти или потребителски вход. Тези изисквания не се отнасят до определени технологии или платформи. Въпреки че може да има изискване за използване на база от данни например, дизайнът на логическата архитектура не указва специален софтуер за управление на структурата на базата от данни.

Основни компоненти на логическата архитектура на SharePoint

Ферми(Farms) – организират архитектурата на най-високо ниво.

Service applications – осигуряват ресурси които могат да бъдат споделени между сайтове във ферма или между отделни ферми

Application pools – в IIS application pool е група от един или повече URLs които се обслужват от определен процес на операционната система. Когато се създава Web Application или Service application в SharePoint 2013 може да се избере съществуващ application pool или да се създаде отделен.

Web applications(уеб приложения) – web application реализира групиране на функционалност и общи настройки между колекции от сайтове. Всеки web application има IIS уеб сайт

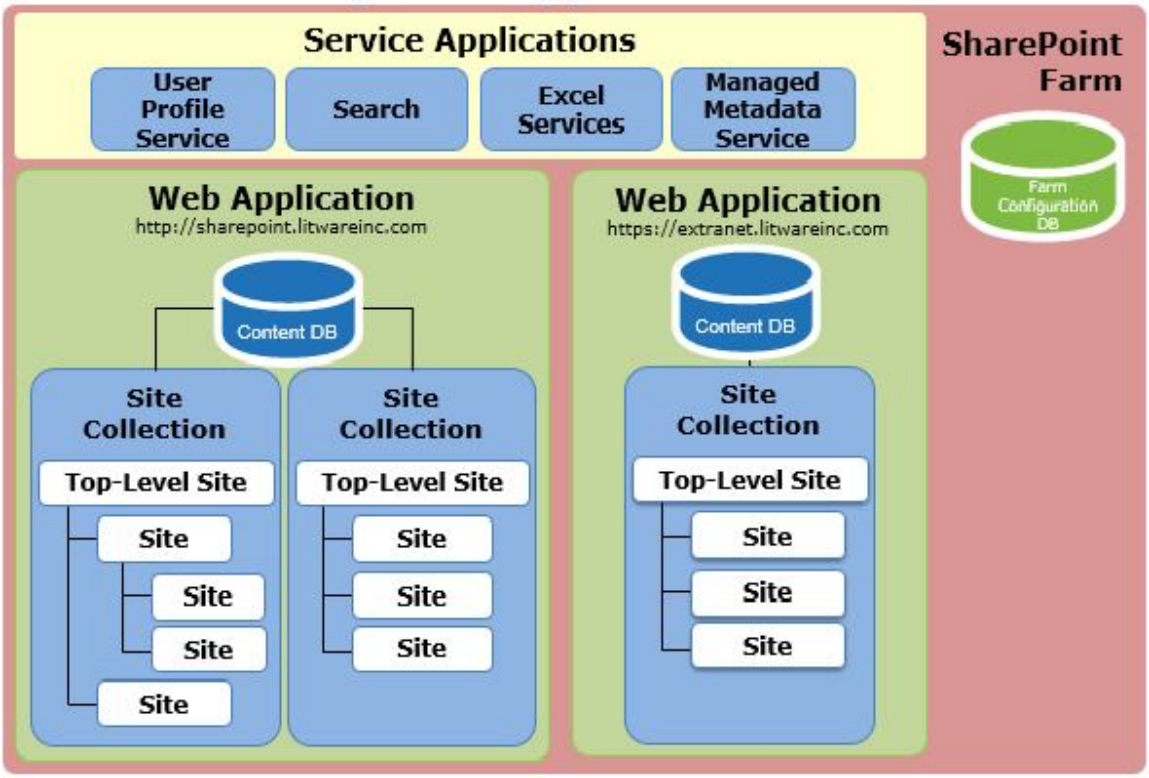
Зони – Зоните се използват за да се осигури достъп до един и същ web application през различни URL-и. За отделните зони могат да се зададат различни настройки през IIS

Content Databases(База данни със съдържание) – По подразбиране, всички данни, които се въвеждат в сайтовете се намират в content databases

Site collections – Това е логически обособено множество от сайтове, които имат общ top-level сайт и споделят определени административни настройки

Сайтове(Sites) – логически обособена част от съдържанието, която има свои контейнери с данни и нива на правомощия за различните потребители. Може да съдържа една или повече страници

Списъци и библиотеки(Lists и Libraries) – контейнери за данни, които съхраняват елементи с еднаква структура като контакти, задачи или документи



Модул 18: Нуждата от уеб портали и системи за управление на съдържанието

- Нуждите на бизнеса
- Възможности на CMS
- Категоризация на CMS
- Примери
- Информационна архитектура на SharePoint 2013
- Йерархия на обектите
- Йерархия на библиотеките
- Site columns и Content types
- Страници

Нуждите на бизнеса

- Предназначение на корпоративен уеб сайт:
    - Предоставяне на актуална информация за дейността на корпорацията

- Предоставяне на актуална информация за предлаганите продукти
- Възможност за лесна промяна и добавяне на информация
- Селектиране на най-значимата информация

- За реализиране на изискванията се използват системи за управление на съдържанието

Целта на уеб сайта на дадена компания е да осигури актуална информация и съдържание за широка публика, с възможност за бърза промяна и добавяне на информацията, за да могат да се отразят своевременно новите продукти и новините около компанията. Бизнесът се променя бързо, появяват се нови промоции и се променят предлаганите продукти. Успешният уеб сайт гарантира, че с потребителите се споделя точната информация. Променя се не само големината на предлаганата информация, но и типа на съдържанието – като документи, снимки, съобщения и т.н. За да може разработчика да се справи максимално бързо и лесно с тези предизвикателства той трябва да използва CMS. Чрез CMS се улеснява изграждането на уеб приложения, които използват голям набор от информация и функционалност.

Кой уеб сайт е успешен

- Информация
- Услуги
- Организация на съдържанието
- В последните години се увеличава и обема и типовете информация, която е достъпна в интернет
    - По-мощен хардуер
    - По-добро развитие на CMS

Търсенето на информация е основното предназначение на уеб-страниците и на интернет като цяло. Именно това е основната цел на всички сайтове – да предоставят желаната информация на интернет потребителите. Предоставянето на полезна информация поражда конкуренция между отделни интернет страници, ето защо авторите на всяка страница се стремят да публикуват повече и по-актуална информация, за да привлекат вниманието на потребителите.

Дизайнът на дадена интернет страница оказва голямо влияние върху мнението, което си изграждат потребителите. Едно от първите неща, които забелязват повечето посетители, е външния вид на уебстраницата.

Когато създавате интернет страница трябва да обърнете внимание на един много важен детайл – скоростта, с която зарежда тя. Една голяма част от интернет потребителите не обичат да чакат и просто прекъсват зареждането на бавно зареждащите страници. Ето защо трябва да се погрижите вашия сайт да се отваря колкото се може по-бързо и безпроблемно.

Възможности на CMS

- CMS и допълните разработки към тях предоставят множество функционалности
    - Динамично публикуване на информацията от базата данни
    - Допълнителни услуги
        - Системи за резервация
        - Системи за плащания
    - Различни шаблони (templates) за изглед
- Повечето функционалности са предварително програмирани
- Изпълняване на изискванията по възможно най-лесният начин

Вградени функции

Много от CMS програмите имат вградени редица полезни, често използвани в интернет страниците фукнции. Ето пример за някои от тях:

Поле за търсене: нещо като малка Google търсачка, но търсеща само сред страниците на сайта. Това позволява на потребителите да намерят бързо във сайта съдържанието, което ги интересува.

Статистика на страниците: информация за броя посещения на всяка една от страниците на сайта.

Менюта: можете да включвате избрани заглавия в менюта, които помагат в ориентирането на потребителите в сайта.

Форма за регистрация: автоматично записване на регистрираните се към сайта потребители. Полезно е например ако искате само регистрирани потребители да имат достъп до определени статии или привилегии (оставяне на коментари, гласуване за статии и др.)

Автоматично показване на най-новите и/или най-популярните статии. Тази информация е полезна както за честите потребители на сайта, така и за новодошлите.

Възможност за добавяне или спиране на статии с едно кликване.

Гъвкавост в организацията на страниците по категории.

Вграден HTML редактор, който улеснява писането на статиите: можете да пишете страниците без да познавате HTML.

Допълнителни компоненти

Много от CMS програмите са изградени с възможност за добавяне на отделни компоненти (plug-in). Това дава възможност за добавяне на функции, които не се поддържат от основната програма. В интернет често можете да намерите стотици, ако не и хиляди компоненти за по-известните CMS програми: картинни галерии, компоненти за архивиране на целия сайт, автоматичен превод на страниците, електронна търговия и какво ли още не. Повечето компоненти са безплатни като самите CMS програми, за които са направени.

Шаблони

Много от CMS програмите работят с т.нар. шаблони. Шаблонът определя как ще изглеждат страниците: какъв да е размера на страницата, как да изглеждат менютата, хипервръзките, заглавията и параграфите и т.н. CMS програмите имат вградени няколко шаблона, но за по-популярните CMS програми в интернет се предлагат стотици безплатни шаблони.

Ако случайно нищо от вградените или предлаганите в интернет безплатни шаблони за CMS не Ви допадне, то сред платените (цените варират доста, но много от дизайнерските разработки могат да се намерят за 50-ина долара) почти със сигурност ще намерите много точно съответстващ на нуждите дизайн.

Разбира се, шаблоните винаги могат да бъдат дообработени и променени посредством HTML, CSS и JavaScript

Дизайн на уеб портали

- Много често порталите не просъществуват защото при изграждането не е обърнато достатъчно внимание на групата от потребители, за които е предназначен сайта.
- Няколко основни правила при проектирането на успешни уеб сайтове:
    - Фокусиране върху целите на организацията
    - Балансирано съдържание и дизайн
    - Обновяване на информацията
    - Изследване на потребителите

Както във всеки технически проект, за създаването на уеб-сайт се изисква точна формулировка на задачата, разработване на отделните елементи и съединяването им в обща функционална система.

Всеки квалифициран дизайнер трябва да има формулирана техническа задача. Яснотата по поставената задача, позволява адекватна работа между няколко специалиста, изключва недоразуменията между клиента и изпълнителя. В техническата задача по проектиране дизайна на уеб-сайта, най-често се разглеждат следващите въпроси:

Желанията на клиента относно художественото и техническо изпълнение;

Списък на конкурентите, на които сме харесали дизайна;

Гамата от цветове, с която ще бъде разработен сайта, шрифта;

Елементите на фирмения стил, които е желателно да се приложат в разработката;

Структурата на ресурса (количество страници, последователност, преход, връзки);

Наличие на програмни скриптове;

Перспективи за бъдещо развитие и разширяване на ресурса;

Главната страница на сайта е ключов елемент в дизайна, от това колко професионално е изпълнена ще зависи популярността на сайта сред посетителите. Според статистиките, повечето хора оценят сайта по неговата обложка, по неговата главна страница. Разработката на подразделите на сайта се подчиняват на основни правила, сформирани в процеса на създаване на основния раздел и се проектира в общ стил. Допълнителните раздели, такива като форумите, могат да имат свой собствен дизайн, затова най-често се разработват отделно.

Основни възможности на CMS системите

- Идентифициране на различен тип потребители и техните права за администриране и контрол на информацията
- Поддръжка на workflow за задачите, с възможност да бъдат информирани потребителите на следващо ниво
- Да се поддържат версии на документите

При използването на workflow(управление на технологичния процес) публикуването на съдържание става с един клик на мишката като е възможно да публикувате текста веднага (ако имате права за това) или да инициирате работен процес. Ако инициирате работен процес това означава, че изпращате текста за преглед към следващия човек в йерархията за одобрение, който има по-големи права от вас. Различните права за достъп дават възможност да се конфигурира работния процес според нуждите на съответната организация – процесът на одобрение може да е само от една стъпка или да включва последователност от стъпки, които предполагат участие на сътрудници, преводачи и редактори.

Категоризиране на CMS

- **Enterprise CMS**
    - Най-често се използват в интранет
- **Web CMS**
    - Уеб приложение, което улеснява създаването, редактирането и контрола на съдържанието.
    - Употреба на бази данни за съхранение на metadata
- **Mobile CMS**
    - Често са допълнение към по-обширни CMS
    - WebSphere Everyplace Mobile Portal

Уеб система за управление на съдържанието

Уеб системата за управление на съдържанията е пакет или самостоятелно приложение за създаване, управление, съхраняване и използване на съдържанията в уеб страниците. Уеб съдържанието включва текст и вградени графики, снимки, видео, аудио файлове, както и програмен код. Уеб системата може да събира и индексира съдържания, избира и сглобява съдържания по време на работа или да доставя съдържания за определени потребители по определен начин, като например друг език. Този вид системи обикновено позволяват на клиента да контролира HTML-базирани съдържания, файлове документи и уеб хостинг планове, въз основа на системата и дълбочината на нишата, която тя обслужва.

Компонентна система за управление на съдържанието

Компонентната система за управление на съдържанието е специализирана в създаването на документи от съставни части. Като например CCMS, която използва DITA XML позволява на потребителите да сглобяват индивидуални теми в една карта (документ). Тези компоненти могат да се преизползват с друг документ или с друго множество такива. Това гарантира, че съдържанието е наблюдавано и в целия набор документация. Все пак този вид система не е подходяща за големи организации, защото те имат възможност да си поръчат система, отговаряща на техните изисквания.

Корпоративна система за управление на съдържанието

Корпоративната система за управление на съдържанието организира документи, контакти и записи, свързани с процесите на търговската организация. Тази система

структурира корпоративната информация по начин, който е най-подходящ за организацията, като предоставя лесна достъпност до нея от служителите и клиентите и осигурява максимална сигурност на информацията.

Примери за CMS

- Umbraco
    - Отворен код през 2004
    - Базиран на Microsoft ASP.NET
    - През 2008 е имало 50 000 работещи уеб сайта иззградени с Umbraco
- Documentum
- FatWire
- IBM Lotus Web Content Management

WordPress — е CMS блог система с отворен код, който се разпространява под лиценза на GNU General Public License. Написан е на PHP, а за база данни се използва MySQL. Софтуерът има различни сфери на приложение — от блогове до сложни новинарски ресурси и дори интернет-магазини. Вградената система от „теми“ и „плъгини“, и стабилна архитектура, която позволява конструирането на най-различни проекти.

Joomla - Джумла също се разпространява под безплатен лиценз (GNU General Public License) и е написана на PHP и използва MySQL.

Първата версия на Джумла! излиза на 16 септември 2005 г. на базата на Mambo 4.5.2.3, комбинирана с няколко други добавки и поправки.

Drupal – също безплатна система с отворен код. Тя е написана на PHP и се разпространява под GNU General Public License. Над 1,5% от всички уеб сайтове в световен мащаб използват тази система. Те варират от личните блогове на корпоративни, политически и държавни обекти, включително и whitehouse.gov и data.gov.uk.

Избор на CMS

- Фактори за които трябва да се вземат предвид:
    - организация за която се разработва - големина
    - сложност – динамична информация, бази данни
    - тип на страниците – прости или базирани на стандартна подредба на графични елементи

- изисквания за структурата – навигация, връзки между страниците

- Три варианта за комбиниране
  - Порталът е направен изцяло без CMS
    - Осигурява пълен контрол
    - Изисква нов код при промяна
  - Вграждане на код във определена CMS
  - CMS и порталът са отделни
    - Дава свобода на избора

За повече информация:
https://rykovodstvo.wordpress.com/2013/09/05/%D0%B2%D0%B8%D0%B4%D0%BE%D0%B2%D0%B5-cms-%D0%BA%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F-%D0%BD%D0%B0-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8%D1%82%D0%B5-%D0%B7%D0%B0/

Информационна архитектура в SharePoint 2013

- Какво е информационна архитектура
  - Средство за структуриране на информацията в дадена организация
  - Специфична за отделните организации
  - Трябва да е гъвкава и лесна за промяна
- Защо е важна
  - Повишена използваемост и откриваемост
  - Повишена производителност
  - По-добро възприемане от потребителя
  - Сигурност при промени

Информационната архитектура (ИА), наричана още "Информационен дизайн", "Информационно проектиране" или "Инфодизайн" е най-бързо развиващата се част от науката за Интернет. Тя представлява скелета, около който се гради всеки уеб сайт.

Практиката показа, че колкото и стойностна да е информацията, тя може да остане напълно безполезна, ако бъде "погребана" под зле организирани менюта или стои заобиколена от несъответстващ контекст.

Информационната архитектура формира ядрото на един сайт, около което се изграждат всички останали негови компоненти: общата визия на сайта, функционалностите му, навигационните схеми, потребителският интерфейс и т.н.

Накратко, информационната архитектура на един сайт се състои в процеса на прецизно структуриране на информацията, нейното организиране в ясни навигационни схеми, и атрактивното й представяне чрез интуитивни и недвусмисленивизуални елементи.

Крайната цел на информационната архитектура в един сайт е да позволи информацията да бъде лесно достъпна, разбираема и управляема.

SharePoint Site columns

- Site Columns - полетата с данни в структурата на SharePoint
- Някои типове
    - Multiple lines of text – тип за съхранение на дълъг текст от повече от един редове
    - Choice – тип за съхранение на данни с предварително определени възможни стойности
    - Number – тип за съхранение на числа
    - Currency – тип за работа с валутни единици
    - Date and Time – тип за работи с дати
    - Lookup – тип за пренасочване към данни от друго поле
    - Yes/No – тип за съхранение на стойности „да“ или „не“
    - Hyperlink or Picture – тип за съхранение на връзка към определен ресурс
    - Image – тип за съхранение на графично изображение
    - Managed Metadata – тип за съхранение на метаданни

Site columns представляват дефиниции на полетата с данни в структурата на SharePoint. Те определят типа на съхраняваните данни, визуализацията на самите данни в определените за това области, както и вида на полетата във формите за въвеждане на данните. За тип на данните в определена колона може да се избере числов или текстов, дата, валута и т.н. Те са най-малкият градивен елемент, който описва информацията, която съхраняваме в портала.

SharePoint Content Types - преизползваема колекция от метаданни

- Съдържат Site Columns

- Могат да се преизползват от списъци
- Всеки Content Type има идентификатор
- Всеки Content Type наследява друг
    - Наследяването се указва чрез полето Id

Content type е преизползваема колекция от метаданни, за определена категория от елементи или документи. Те дават възможност за определяне на структурата на определена информация. Представляват множество от site collumns.

За повече информация: https://msdn.microsoft.com/en-us/library/office/ms472236(v=office.14).aspx

Видове страници в SharePoint 2013

- Application pages
    - Реална страница, която се намира на файловата система
- Content pages
    - Виртуална страница, която се намира в базата данни
- Ghosting and unghosting

За повече информация: http://sharepoint.stackexchange.com/questions/59798/understanding-ghosting-unghosting-site-pages-and-application-pages

Страница с уеб части

Уеб част (Web Part) е обособена фунционалност в контекста на SharePoint Server, която може да има и визуална част и може да се преизползва в различни страници.

ASPX страниците може да съдържат области с уеб части, които можете да използвате, за да добавяте уеб части или да позволявате на потребителите на сайта да добавят уеб части при използване на браузъра. Новата страница има същия изглед и създава същото усещане като останалата част от сайта и автоматично се асоциира със страницата образец на вашия сайт, включително границите и елементите за навигация. Когато създавате нова страница с уеб части, можете да изберете от няколко шаблона за страници с уеб части, всеки от които има различно оформление на областите на уеб части.

ASPX страница

Ако създадете ASPX страница, трябва да добавите необходимите области на уеб части и елементи на страница, за да работи страницата по очаквания начин. Ако създадете ASPX страницата от страниците на сайта в SharePoint Designer, тя не се прикачва към страница образец и следователно няма да се показва с изгледа и усещането, границите и навигацията на останалата част от сайта.

HTML страница

Създава празна HTML страница във вашия сайт. Създадената от вас HTML страница не може да бъде преглеждана направо от вашия сайт, тъй като тя няма необходимия ASP код, който се изисква от сайтовете на SharePoint. Страницата обаче може да се използва за съхранение на HTML документи на сайта.

Модул 19: Облачна архитектура за услуги с масово приложение

- Специализирани бизнес решения
- Облачни решения
- Характеристики на облачните решения
- Архитектура и инфраструктура на облачните решения
- Infrastructure as a Service (инфррастуктура като услуга, IaaS; използване само на физическите ресурси на облака)
- Platform as a Service (платформа като услуга, PaaS; предоставя на клиентите програмен език като платформа или софтуер)
- Software as a Service (софтуер като услуга, SaaS; облака предоставя цялостно решение, включително използваният софтуер)
- SharePoint Online
- Microsoft Azure

Специализирани бизнес решения

- Доставчици
    - HP

  - IBM
  - SAP
  - Microsoft
- Планирането и изпълнението на проектите е времеемко и скъпо
- Изисква умения и опит
- Скъпа поддръжка
- Съвместимост със съществуващи системи

Всеки бизнес е специфичен. Съществуват редица решения, подходящи за интегриране и работа в различни структури. Често се прибягва и до разработване на специализиран софтуер за конкретен клиент, който не може да бъде използван никъде другаде. Това обаче изисква прецизно планиране и дълго време за изпълнение. При разработката на специфичен продукт се изисква и поддръжка, която допълнително оскъпява и усложнява процеса.

От друга страна миграцията към друга система може да се окаже по-сложна. Наемането на нови кадри изисква допълнително обучение, което би могло да се пропусне ако се използва вече популярна система.

Облачните решения

- Джон Маккарти, 1960 година – „Изчисленията може някой ден да бъдат организирани като предприятие за комунални услуги.
- Облак е метафора за интернет – технологии и услуги, достъпни през интернет
- Облак за бизнеса – всичко е услуга
  - Софтуер
  - Хардуер
  - Съхраняване на информационни масиви

Изчисления в облак (Cloud computing) е предоставянето на компютърни (изчислителни) услуги, а не на продукт (за разграничение между услуга и продукт виж статията за услуга).

Това е термин от областта на информационните технологии, означаващ използването на споделени ресурси, софтуер и информация, като предоставяни на компютри и други устройства по мрежа (чрез Интернет).

Терминът съвместява понятия като софтуер като услуга, инфраструктура като услуга, платформа като услуга и други съвременни технологии, които под формата на онлайн бизнес приложения, достъпни през уеб браузър, задоволяват изчислителни потребности докато съхраняват софтуера и потребителските данни на свои сървъри.

С други думи, понятието се отнася както до софтуерни приложения, предоставяни под формата на уеб услуги, така и до достъпа до хардуерните и системни ресурси на център за данни (data centers), които предлагат тези услуги. Всъщност, комбинацията от достъпа до хардуера и софтуера на центъра е това, което е прието да се нарича „облак“. Смята се, че облакът е метафора за Интернет, тъй като често се изобразява така в диаграмите на компютърни мрежи и като абстракция на сложната инфраструктура, която стои зад него.

Определящи характеристики

- Облачната архитектура създава илюзията за безкрайни изчислителни ресурси, налични при поискване, с което се елиминира потребността да се правят предварителни дългосрочни планове за доставката на такива ресурси.
- Елиминира се високата бариера за навлизане и се дава възможност на компаниите да започват с поръчката на малко хардуерни и системни ресурси и да ги увеличават само когато нараснат потребностите им.
- Облачната архитектура дава възможност да се заплащат само изконсумираните изчислителни ресурси, и то за произволно кратък период от време, докато са били реално използвани (например, процесорно време на час или количество памет на ден).

Видове облачна архитектура

- Частен облак
- Публичен облак
- Хибриден облак

Идентифицирани са четири вида облачна архитектура:

Частен облак: инфраструктурата на облака се притежава или наема от една организация и се използва само и единствено от нея.

Общностен облак: инфраструктурата на облака се споделя от няколко организации и служи за поддържането на специфична общност от потребители, които споделят обща мисия, обща политика, общи изисквания към информационната сигурност и др.

Публичен облак: инфраструктурата на облака се притежава от една организация, която продава „облачни“ услуги на широката аудитория,

Хибриден облак: инфраструктурата на облака е съчетание на два или повече облака (частен, общностен, публичен), които остават разграничими, въпреки че са свързани посредством стандартизирана или собственическа технология.

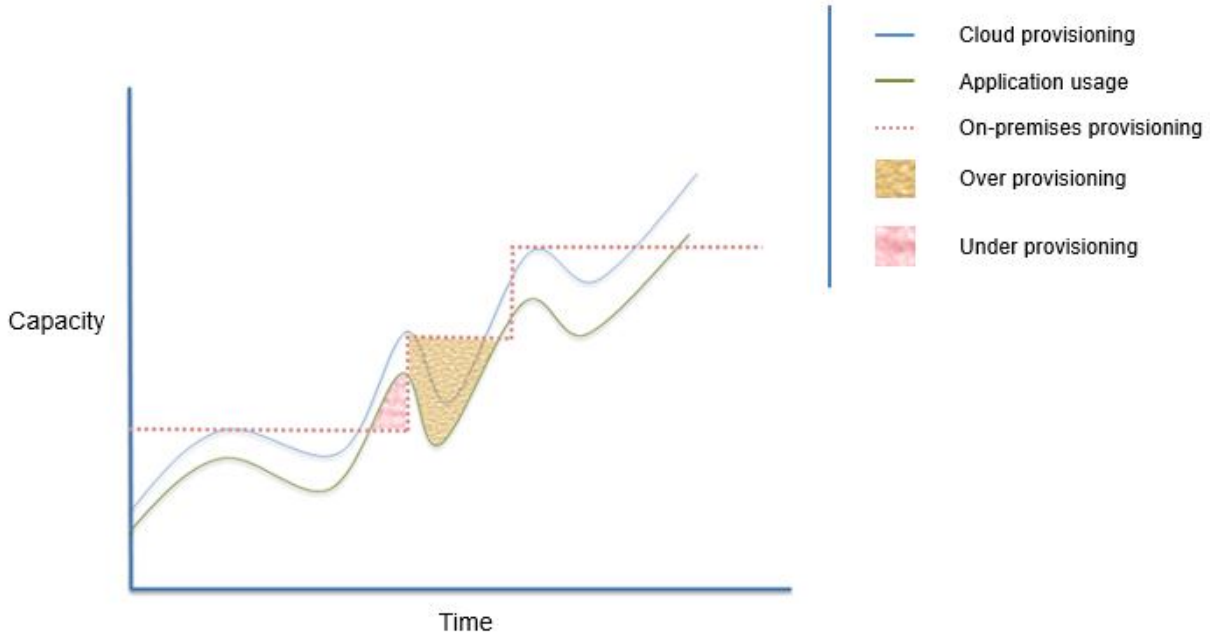
Проблеми при възприемането

Пред потребителите съществуват няколко проблема, свързани с възприемането на технологиите на облачната архитектура:

- Наличност на обслужването
- Сигурност и неприкосновеност на личните данни
- Поддръжка
- Оперативна съвместимост
- Съгласуваност

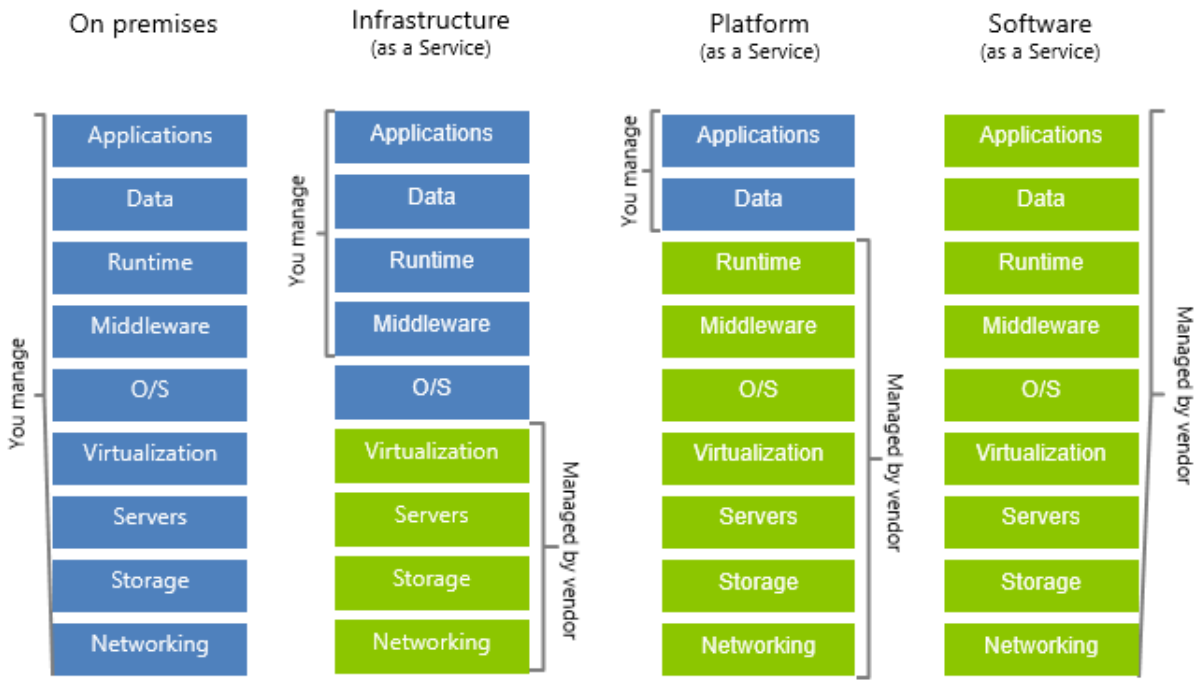
Тези проблеми произлизат от това, че данните, приложенията и изчислителните ресурси вече не са под прекия контрол на потребителите.

Тъй като облачната архитектура не дава на потребителите възможност да разполагат с данните си физически (освен ако ползваната услуга не съдържа опция за запазване на твърдия диск на резервно копие, back-up) това прехвърля отговорността за съхраняването и контрола над данните в ръцете на доставчика. Допълнителни проблеми могат да се породят поради процесите (методи, функции, транзакции), публичността и преносимостта на данните, съвместимостта с конкурентни услуги, непрекъснатостта на обслужването, възстановяването на данни при срив (disaster recovery), договорни клаузи между доставчици и потребители, и др.



Доставка на облачни технологии

- За да се изпълнят изискванията на различни потребители доставчиците на облачни технологии използват различни:
    - Услуги
        - Software as a Service (SaaS)
        - Platform as a Service (PaaS)
        - Infrastructure as a Service (IaaS)



Infrastructure as a Service – Инфраструктура-като-услуга

- Доставчиците осигуряват:
  - изчислителна мощ
  - дисково пространство
  - интернет мрежа
  - оперативна памет
- Клиентите имат контрол да определят сами параметрите на опертивната памет, разход на процесорно време, брой IP адреси, операционна система, инсталиране на софтуерни приложения
- Надеждни Data centers (центрове за съхранение на данните)
- Доставчици
  - Sun Microsystems
  - Dropbox

Доставчиците на облачна инфраструктура като услуга предоставят на клиентите възможност да ползват изчислителна мощ, дисково пространство, интернет мрежа, оперативна памет и други основни технологични ресурси, които правят възможно

внедряването и работата на различни софтуерни програми като операционни системи и приложения. Инфраструктура-като-Услуга предоставя също виртуална среда като услуга, при която клиентите имат контрол да определят сами параметрите на опертивната памет, разход на процесорно време, брой Internet Protocol(IP) адреси, операционна система, инсталиране на софтуерни приложения както и допълнителни мрежови компоненти като защитна стена (firewall), load balancers (разпределители на трафика) и др. Клиентите нямат достъп до базовата инфраструктура на самия облак, а определят единствено параметрите на собствената виртуална машина. Важно условие за доставчиците на IaaS е да ползват услугите на надежден дата център и да предоставят на потребителите богата на информация система за мониторинг на виртуалните машини. Примери за доставчици на IaaS в световен мащаб са Amazon EC2 and S3, Sun Microsystems и Dropbox.

Platform as a Service – платформа-като-услуга

- Доставчиците предоставят на клиентите програмен език като платформа или софтуер като например Java, Python или .Net
- Разработване на решения и използването им чрез Application Programming Interface (приложно-програмен интерфейс, API) или портали
- Клиентите имат контрол над разработените приложения и до известна степен до настройките на средата, в която се хостват приложенията
- Отговорност на доставчика на услугата е да се погрижи за сигурността на средата за разработване
- Примери
    - Google App Engine
    - Force.com
    - Microsoft Azure.

Доставчиците предоставят на клиентите програмен език като платформа или софтуер като например Java, Python или .Net (не само тези, разбира се) за да внедрят създадени собствени или приложения на трета страна в облачната инфраструктура и да ги направят достъпни през интернет с API или уебсайт портали за своите клиенти. Доставчиците на платформена облачна услуга предоставят различни услуги на разработчиците на приложения като виртуална среда за разработка и предварително настроени за тази среда инструменти, стандарти за приложението, съобразени с изискванията на разработчика както и предварително изграден канал за разпространение, който се предоставя на разработчиците на публични приложения. Клиентите имат контрол над разработените приложения и до известна степен до настройките на средата, в която се хостват приложенията. Както и при SaaS клиентите нямат контрол до базовата облачна инфраструктура като мрежа, сървъри, операционна система и дисково пространство. При PaaS отговорност на доставчика на

услугата е да се погрижи за сигурността на средата за разработване, докато отговорност на разработчика е сигурността на самото приложение. Примери за доставчици на PaaS са Google App Engine, Force.com и Microsoft Azure.

Software as a Service – Софтуер-като-услуга

- Доставчикът предоставя на клиентите достъп до лицензирани софтуерни приложения, които са инсталирани на облака
- Модел на заплащане-при-употреба (pay-per-use)
- Не се изсиква да управляват или контролират елементи от инфраструктурата на облака като мрежа, сървър, операционна систем или хранилище за данни
- Употреба
    - текстови редактори
    - аудио-видео софтуер
    - уеб базирани имейл програми

При този модел доставчикът предоставя на клиентите достъп до лицензирани софтуерни приложения, които са инсталирани на облака. Потребителите могат да достъпват тези приложения през интернет чрез уеб браузер като се прилага модел на заплащане-при-употреба (pay-per-use). Потребителите не се изсиква да управляват или контролират елементи от инфраструктурата на облака като мрежа, сървър, операционна систем или хранилище за данни. Понастоящем Софтуер-като-Услуга (SaaS) е перфектно работещ модел за достъп до леки приложения като текстови редактори, аудио-видео софтуер, уеб базирани имейл програми и пр. Когато става въпрос за ползване на тежки ресурни приложения като 3D игри, качеството на услугата може да се понижи заради времето на предварителна подготовка на приложението (buffering time). В общия случай доставчикът на услугата разполага и поддържа приложението, което се наема от клиентите на виртуална машина в облачната технологична среда. SaaS се предлага от известни производители като Zoho Suite, Apple’s MobileMe и Google Docs.

SharePoint Online

- Споделяне
    - Създаване и преглеждане на публикации
    - Информационен канал
    - Работа с микроблогове
    - SharePoint Newsfeed

- Организиране
    - Връзка с OneDrive

SharePoint Online е версия на SharePoint която позволява разработването и модифицирането на сайтове чрез продукт, намиращ се изцяло в интернет. При използването му се минимизират проблемите свързани с поддръжка, тъй като той се обновява и поддържа от екипа на Microsoft. Това е изключително удобно, тъй като при смяна на функционалността Microsoft гарантира че разработените решения ще продължат да функционират. В случаите когато се работи със SharePoint Server обаче, при смяна на версията се налага и миграция на продукта, което често води до проблеми.

Някои нови функционалности са налични преди излизане на версия на SharePoint Server.

За повече информация:

1. https://products.office.com/bg-bg/sharepoint/sharepoint-online-collaboration-software
2. https://support.office.com/bg-bg/article/SharePoint-Online-%D1%81%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%B8-%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D1%8F-8f34ff47-b749-408b-abc0-b605e1f6d498?ui=bg-BG&rs=bg-BG&ad=BG
3. https://support.office.com/bg-bg/article/%D0%A0%D1%8A%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%B7%D0%B0-%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D0%B0%D0%BD%D0%B5-%D0%B2-SharePoint-Online-%D0%B7%D0%B0-Office-365-Small-Business-a496eeeb-f9ce-40e3-aa5d-932cb268badc?ui=bg-BG&rs=bg-BG&ad=BG

Microsoft Azure

- Платформа за облачни изчисления и инфраструктура
- Microsoft Azure осигурява
    - Compute and hosting services(услуги за изчисления и хостване)
    - Storage and databases(хранилище за данни и бази от данни)
    - Building block services (функционалност, потдържаща основните компоненти на облачната структура)
    - Структурирана е така, че непрекъснато да е online
- Поддържа голям набор от технологии и платформи

Microsoft Azure е гъвкава облачна платформа, която осигурява възможност за секунди да вдигнете виртуални машини с Windows или Linux, да внедрите Web сайтове и приложения, да използвате база от данни, да добавите допълнително място за съхранение, колкото ви е необходимо, да внедрите Windows клиентски приложения в облака, които да се достъпват отдалечено от всяко устройство, да разработвате приложения с Visual studio online да вдигнете виртуална мрежа, да осигурите защита на достъпа до вашите данни и приложения с допълнително ниво аутентикация и др.

Модул 20: Употреба на уеб аналитика (web analytics) и контрол в портали

- **Видове web analytics**
- **Контрол на версиите**
- **Recycle bin**
- **Оторизация**
- **Потребителски групи**
- **Аутентикация**

Какво е уеб аналитика

- Уеб аналитиката, най общо казано, е инструмент за измерване на трафика на уеб портала
- Употреба
    - Бизнес проучвания
    - Рекламни кампании
- В какво се състои
    - Измерване на посещенията към портала
    - Индивидуалните посещения към всяка страница
    - Изследване на навигацията на потребител вътре в самият портал
- Порталите често презентират няколко теми на една страница

Уеб аналитиката представлява анализиране на информация за потребителите на даден портал или информация в интернет. Може да бъде използвана като инструмент за бизнес анализи, анализиране на резултати от рекламни кампании или пазарни проучвания. Уеб аналитиката предоставя данни за брой посетители на даден портал или брой прегледи на даден материал, проследяване на поведението на даден потребител в портал на базата на изградената навигация. Тази информация се обработва и сравнява за да се определи колко успешно уеб порталът удовлетворява поставените цели, какво може да се промени за да се постигнат по-добри резултати, за да се определи по-точно таргет групата на уеб сайта или маркетингова кампания и т.н. Проследяването на потребителският трафик е необходимо за да се разбере кои части от портала предоставят необходимата информация на потребителите.

Порталите често представят информация от различна тематика и с различно предназначение на една страница, което поставя специфични предизвикателства пред разработчиците и администраторите относно изводите за употребата на портала.

Видове web analytics

- Целта на уеб аналитиката е архитектите и дизайнерите непрекъснато да усъвършенстват порталите и да се конкретизира по-подробно таргет групата
- Аналитичната информация може да се събере по следните начини:
    - Анализиране на логовете на сървъра – специално създадена система на сървъра
    - Active Page tagging – JavaScript технология, която изпраща информация до сървъра на момента
    - Анализиране на потребителските кликове

В порталните приложения от съществено значение е каква информация за потребителското поведение да се следи и записва. Тя до голяма степен определя каква ще бъде структурата на портала в бъдеще, както и как ще бъдат формирани URL адресите на неговото съдържание. Може да се очаква че познанията за потребителското поведение ще доведат до увеличаване на приходите от дейността, както и да намали разходите за тестване на функционалността.

Анализиране на логовете на сървъра – уеб сървъра разбива информацията за поведението и транзакциите на потребителите и я записва във файлове. Типичната сървърна уеб аналитика се базира на метаданни, като лог-файловете се записват на самият сървър

Active Page tagging – това е технология, която се реализира от страна на клиентският софтуер в реално време, базирана на скриптове. Определена информация се събира и се изпраща към сървъра посредством AJAX заявки. Този метод използва XmlHTTPRequest(обект от JavaScript), което може да се окаже проблем при определени настройки за правата на браузъра.

Анализиране на потребителските кликове – обръща се внимание на самите кликове като част от поведението на потребителя. Анализира се броя на кликовете и разпределението им в различните части на портала

Следене на събитията

- AJAX технологията
    - Данните се променят непрекъснато
    - Няма кликове
    - Няма смяна на страницата
    - Трябва да се следят callback методите
- Google Analytics (GATC)
    - Много браузъри има защити и блокират cookies, което е проблем за GATC
    - Cookies са проблем и при мобилните устройства

С развитието на уеб технологиите проследяването на събитията придобива все по-голяма значимост. В традиционната уеб страница всеки клик на препратка или бутон се свързва със заявка към сървъра и повторно зареждане на страницата. При използване на AJAX технологията обаче се опресняват само определени части от съдържанието, без да се презарежда цялата страница. Това може да става автоматично чрез скриптове от страна на клиента. Тогава няма да има регистриране на кликове. Възможност да се проследят събитията дават callback методите, които са основна част от структурата на AJAX.

Google Analytics използва page tagging технологията. Базира се на скрит фрагмент от JavaScript код, който потребителя прибавя при посещението на всяка страница. Целта е да се събере информация, която да се изпрати към сървърите на Google, където тя може да бъде обработена по подходящ начин. GATC зарежда голямо количество данни от Web сървъра на Google и свързва различни променливи с профила на потребителя. Тази технология използва cookies на анонимните потребители за да може да засече дали са посещавали сайта

и преди това и от къде е достигнал потребителя до съдържанието му. Много браузъри предоставят филтриране на приложенията и могат да блокират GATC. От друга страна отделените мрежи могат да замаскират отделните потребители и по този начин да объркат данните, които се получават. GATC също среща трудности с мобилните технологии във връзка с използването на cookies.

Разлики между порталите и обикновените сайтове

- От гледна точка на различните потребители
    - Информацията е различна
    - Дори цели секции от порталите са недостъпни за някои потребители
- От гледна точка на композицията на страниците
- От гледна точка на документите, съхранени в CMS
    - Различни пътища за връзка до един и същи документ
    - Кеширането на тези документи може да се окаже проблем са аналитичната система
- От гледна точка на технологията и за композиране и визуализиране на страниците

Порталите по своята същност са уеб сайтове, в които се добавят данни и възможности от различни измерения. Едно от тези измерения за потребителите. Порталите определено са достъпни за определени потребители – за корпоративните портали има изключително значение към каква група принадлежат потребителите. В публичните портали потребителя трябва да се идентифицира и да предостави информация кой всъщност е той. Това е съществена разлика между портала и уеб сайта, тъй като той може да има коренно различен изглед и възможности при различни потребители.

Друго измерение на възможностите на портала е композицията на страниците. При уеб сайтовете информацията е структурирана в статични HTML страници, докато при порталите тя се определя и визуализира динамично в зависимост от поведението на потребителя.

Третото измерение е свързано с не дотолкова с информацията като визуализация пред потребителя, колкото с документите, които се качват за общо ползване. Много от порталите имат изградена връзка с CMS. Порталът може да предостави препратка към един и същ документ, но в различни категории. Когато потребителя заяви използването на такъв документ възникват две събития – едно за заявка за документа и второ, което указва от къде е бил достъпен ресурса. Това може да създаде проблем когато се използва кеширане на документите, понеже заявката може да не бъде отразена в сървърните логове.

Контрол на версиите в SharePoint

- Конфигуриране на списък или библиотека
- Изискване на check-out(заявка за редактиране)
    - По подразбиране
    - Предпазва от конфликти
    - Останалите потребители могат да променят само след
        - Check-in(предаване на редакциите)
        - Връщане на промените от потребителя или някой със съответните права
        - Лимитиране на броя на главните версии
- Потребителски интерфейс
    - Сравнение
    - Връщане към стара версия

Контролът на версиите в SharePoint представлява функционалност, чрез която може да се проследи историята на промяната на данните, както и кой ги е обработвал. В основните си направления той не се различава съществено от контрола на версиите предоставен от други системи за споделяне на документи. За различните структури от данни контролът на версиите може да бъде настроен по различен начин, което позволява гъвкавост – минимизира се потреблението на ресурси докато бъде осъществен напълно необходимият контрол над информацията.

Особености на контрола на версии

- Check-in и check-out
- Major и minor версии (главни версии и чернови)
- Approval процес (процес на променяне и одобрение на промените)
    - Publish – чрез това действие информацията се публикува за преглед пред анонимните потребители

- Approve – чрез това действие потребителите, които отговарят за съдържанието потвърждават действията, направени от редакторите
- Reject – чрез това действие се отказват промените направени от редакторите на конкретният документ

- RoboHelp – приложение за подпомагане на създаване на персонализирано съдържание. За повече информация: https://en.wikipedia.org/wiki/Adobe_RoboHelp

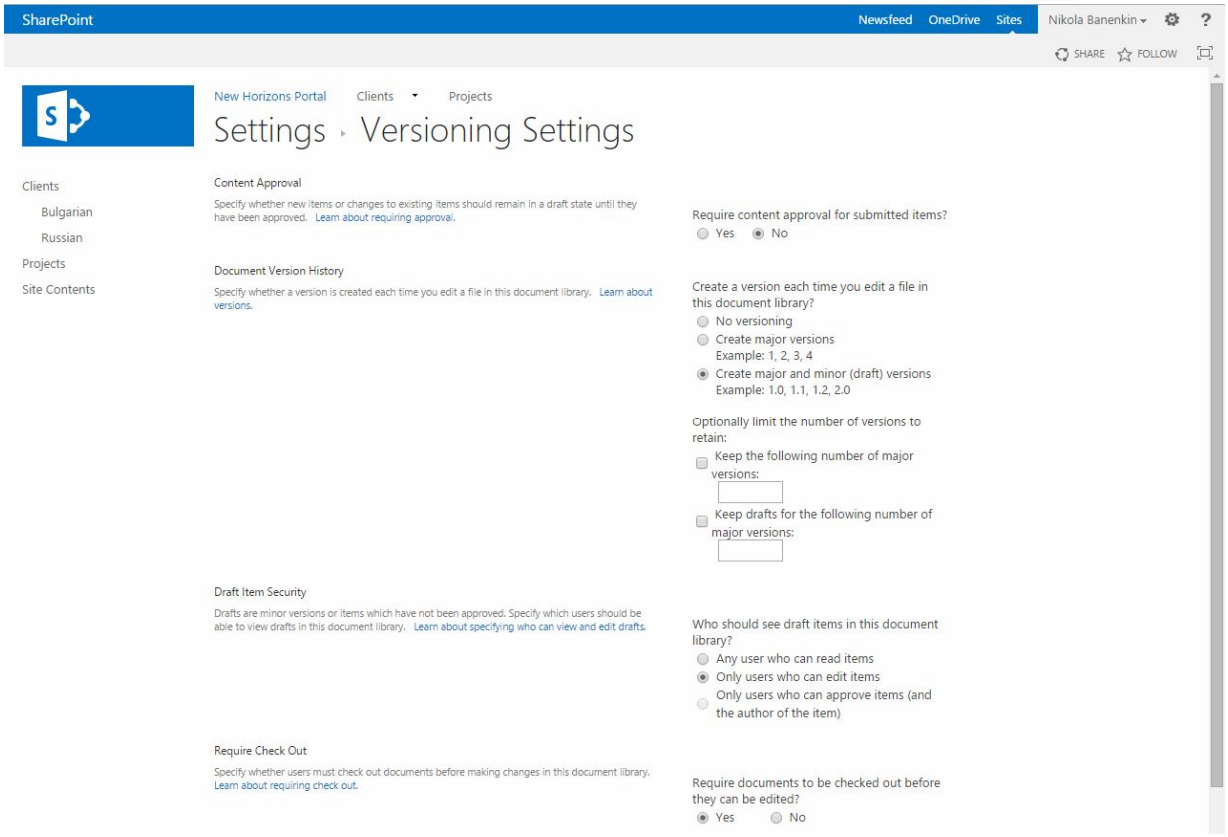
Друга възможност на контрола на версиите е предпазване от възникване на конфликти при работа на различни потребители над един и същ ресурс. За да се промени даден ресурс се изисква check-out на този ресурс от страна на потребителя, който ще го променя. Тогава той се маркира на сървъра и се забранява редакцията му от други потребители. Тази операция се извършва автоматично. Когато промените са приключили, потребителя, който е редактирал трябва да извърши check-in на ресурса, за да го отключи за редакция от другите потребители. Предоставени са и възможности за работа на няколко потребителя в различни секции на един и същ документ.

Потребителският интерфейс, предоставен от SharePoint дава възможност за бързо извършване на check-out, за проследяване на списък с промените в хронологичен ред, както и информация кой е заключил ресурса за промяна в момента. Друга възможност е сравнението на различни версии на ресурса.

Библиотеките и списъците подържат два вида версии – major и minor. Когато се извърши check-in на даден документ, се избира каква да бъде новата версия – major или minor. В случай че се избере major системата прави документа видим за по-голяма аудитория, в зависимост от настройките, които са направени за права на потребителите. Minor версиите (Drafts) се използват за документи, които не са напълно довършени – например автор на статия е приключил работата си, но се очаква да бъде добавен снимков материал към нея.

За да се запазват major и minor версиите на документите се изисква повече място, но е добре да се има предвид, че SharePoint запазва само промените направени в дадена версия, вместо целият документ.

Процеса на работа върху документа трябва да завърши с публикуването му - в някои случаи то трябва да бъде извършено след редакция и т.н. За това има възможност да се изисква content approval от потребители със съответните права. Ако се избере този подход е възможно документът да се редактира множество пъти от различни автори, но на края само един човек да може да го одобри за публикуване.



Recycle Bin

- Преглед
- Изтриване
- Възстановяване

Данните, които се пазят в Recycle Bin остават там докато не бъдат изтрити от там или докато не изтече определеният брой дни, който се настройва през централната администрация.

Recycle Bin е реализирано на две нива – първото е видимо от потребителите на сайта, второто от site collection администратора. Когато даден документ бъде изтрит той попада в Recycle bin. Той може да бъде изтрит и от там, но в такъв случай вместо да бъде загубен се препраща в Site collection recycle bin. От там site collection администратора може да го изтрие напълно или да го възстанови.

http://redmondmag.com/articles/2014/11/06/sharepoint-2013-recycle-bin.aspx

Оторизация

- Метод, който позволява или забранява потребителския достъп въз основа на фактори като принадлежност към група
- Област на оторизацията
    - Site/List/Folder/Item – задаване на права на различни нива от информационната архитектура на SharePoint
    - Security principals – различни обекти на които могат да се задават правомощия
        - User – оторизация на определени потребители
        - Group – задаване на правомощия на базата на определени потребителски групи

В SharePoint 2013 администратора трябва да конфигурира оторизация за да контролира достъпа на потребителите до обектите и съдържание като уеб сайтове, list items, папки или документи. Оторизацията е метод за предоставяне на достъп базиран на фактори като членство в група, притежание на роля или директен достъп.

За да зададе достъп до даден обект, администратора трябва да даде права директно на потребителя, или да добави потребителя в група и да даде необходимите права на групата. Друг вариант е да добави роля на потребителя и да разреши достъпа за съответната роля.

Има сценарии при които в SharePoint фермата не трябва всички сайтове да бъдат достъпни от всички потребители. Това позволява да се скрие конфиденциална информация или информация необходима само на определени отдели. За някои типове сайтове, като публичните фирмени портали, е необходимо да се разреши достъп на всички потребители. От друга страна, интранет порталите трябва да бъдат достъпни само от служители.

Планиране на оторизацията

- Сайт колекциите са областта за администриране
- Site permissions
    - Позволява на потребителите да изпълняват определени действия на ниво сайт
    - Планиране на правомощията на ниво сайт
    - Най-малка привилегия
    - Използва стандартна група
    - Използва нива на достъп
    - Нива на достъп
    - Комбинации
    - Потребителски групи

Нивата на достъп позволяват на потребителите да извършват определени действия в SharePoint сайтовете. Когато се планира оторизацията е важно да се намери балансът между простото администриране и подходящото разбиване на сигурността на малки елементи. Ако се раздават специфични права за всеки потребител ще се увеличи сложността на администриране на портала, а ако правата за достъп бъдат изградени прекалено повърхностно и просто има шанс нивото на сигурност да не е достатъчно високо. Има няколко правила, които трябва да се спазват при планирането на нивата за достъп в портала:

Потребителите трябва да има необходимите права за да изпълняват специфичните за тяхната длъжност задачи.

Потребителите трябва да се добавят към съществуващите SharePoint групи за да имат възможно най-малко правомощия, освен в случаите когато тези правомощния не са достатъчни

Към групата Owners трябва да се добавят само потребители, които ще могат да променят структурата, настройките, изгледа и нивата на достъп до сайта.

Когато е възможно не добавяйте допълнително разработени нива на достъп на потребителите. По-лесно се управляват дефинираните от SharePoint нива на достъп.

Управление на нивата за достъп

- Нива на достъп по подразбиране
    - Дефинирани са на ниво site collection
    - Full Control, Design, Edit, Contribute, Read, Limited Access, View Only
    - Някой права могат да се променят
    - Допълнително настройване на нивата за достъп
    - Ако тези по подразбиране не са напълно приложими
    - Управление на нивата за достъп
    - Добавяне, изтриване, промяна

Нива за достъп са комбинация от индивидуални права на потребителите, които им позволяват да изпълняват множество от задачи. Например Read правото се състои от 11 индивидуални права за List permissions и Site permissions, които включват преглед на елементи, създаване на предупреждения, преглед на страници.

Нивата за достъп могат да бъдат контролирани на ниво Site collection и включват няколко нива за достъп по подразбиране:

Full control – потребителят има пълен достъп. Това ниво не може да се променя

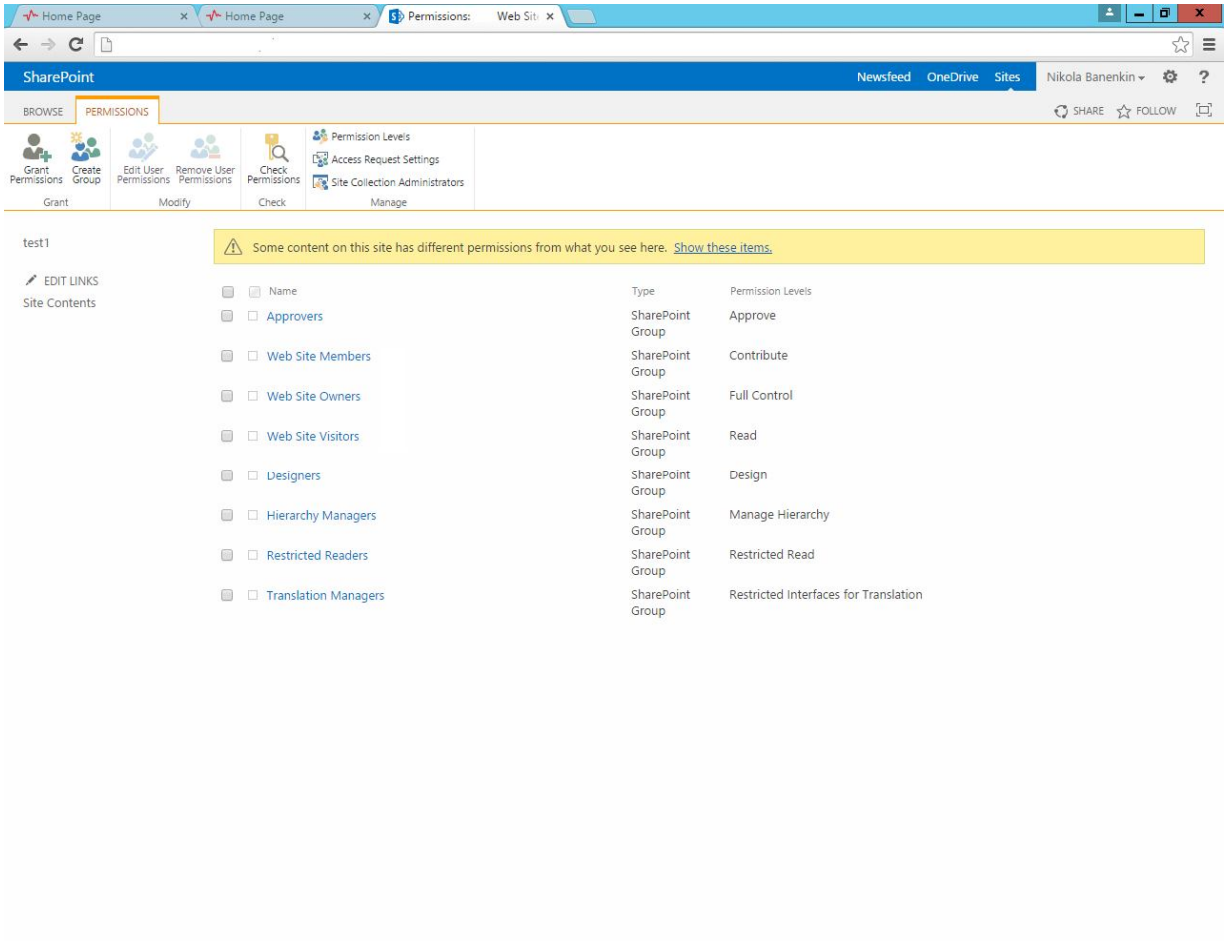
Design – потребителят може да добавя, обновява, изтрива или одобрява съдържание и да променя изгледа на страниците чрез браузъра или SharePoint Designer.

Edit – потребителят може да добавя, редактира или изтрива списъци. Може също така да преглежда, добавя, обновява или изтрива елементи на списъци или документи.

Contribute – потребителят може да преглежда, добавя, обновява или изтрива елементи на списъци или документи.

Read – потребителят може да преглежда страници или елементи на списъци и да изтегля документи

Limited access – потребителят може да преглежда определени списъци, библиотеки от документи, елементи на списъци, папки или документи, за които има специално указан достъп. Това ниво не може да се променя



View only – потребителят може да преглежда страници, елементи на списъци или документи

Управление на потребителските групи

- Групи по подразбиране
    - В зависимост от шаблона на сайта
    - Нива на достъп на групите по подразбиране
    - Owners имат Full Control права
    - Members имат Contribute права
    - Visitors имат Read права

- Управление на групите
- Добавяне на нова група
- Добавяне на потребители към групата
- Даване на права на група

Групите се използват за да направят администрирането на правата по-лесно, понеже те се състоят от множество потребители, на които ще бъдат дадени еднакви правомощия. Едно от най-важните решения, които администратора трябва да направи е как да групира потребителите в портала и какви нива на достъп да им даде.

Дефинираните от SharePoint групи, които може да се използват зависят от това какъв site template се използва при създаването на Site collection

Owners – тази група има ниво на достъп Full control

Members – тази група има ниво на достъп Contribute

Visitors – тази група има ниво на достъп Read

Viewers – тази група има ниво на достъп View only

Аутентикация – процес на идентификазия

- SharePoint 2013 разчита на аутентикация в три направления:
  - Users
  - Apps – допълнителни приложения, които работят в SharePoint среда
  - Server-to-server – Комуникация между сървъри
  - Класическа Windows аутентикация
- Claims-mode
  - Windows Integrated – стандарно използвана от Windows
  - Forms-based – базирта се на информация, попълнена от потребителите във разработена за целта форма
  - SAML token-based – данните за потребителите се форматират в Security Assertion Markup Language (SAML)

Аутентикация е процесът чрез който се установява и проверява самоличността на потребителите, сервизните акаунти или компютрите. SharePoint разчита на аутентикация в три основни направления:

User authentication. Когато потребителя се опита да достъпи до ресурс на SharePoint първо трябва да се разбере неговата самоличност, за да е ясно какви права притежава. SharePoint сървъра установява дали правата на потребителя са достатъчни за достъп до дадения ресурс

App authentication. Когато SharePoint App прави заявка за даден ресурс, трябва да се определят правата както на този app, така и на потребителя, който го е стартирал

Server-to-server authentication. В случай, че друг сървър иска достъп до SharePoint ресурс. Например Exchange server заявява изтегляне на информация за даден потребител. Този процес на аутентикация се базира на предварителна конфигурация на комуникацията между Security Token Service (Услуга за управление на сигурността, базирана на токъни, STS) на SharePoint сървъра и STS на сървъра, който има нужда от достъп.

Claims-based аутентикация

- Claims-based аутентикацията използва следните концепции и термини:
    - Claim
    - Security token
    - Issuing authority
    - Security Token Service (Услуга за управление на сигурността, базирана на токъни, STS)
    - Relying party

Когато се използва claims-based аутентикация SharePoint приложението разчита на външна система, която аутентикира потребителите и предоставя необходимата информация за всеки потребител. Claims-based аутентикацията се свързва със следните термини:

Claim – единична информация за потребителя – име, email адрес или група към която принадлежи.

Security token. Това е сериализирано множество от claims, което е дигитално подписано от Issuing authority

Issuing authority. Това е външна система за управление на идентификацията

Security token service (STS). Това е услуга, която създава и издава security tokens

Relying party. Това е приложението, което разчита на Claims-based аутентикацията за да идентифицира потребителите

Security Token Service Application

- Предоставя STS функционалност
    - Identity Provider STS (IP-STS)
    - Relying party STS (RP-STS)
    - Изпълнява различни функции
    - Прехвърля Windows идентификация в claims-based идентификация
    - Осигурява сигурност на връзката с други платформи
    - Предоставя RP-STS функционалност за Security Assertion Markup Language(SAML) token-based провайдерите

В claims-based аутентикацията STS е уеб услуга, която издава security tokens, за да верифицира самоличността на потребителя.

Identity Provider STS се използва за да създава и управлява claims-based identities.

Relying party STS получава tokens от IP-STS. RP-STS гарантира, че получените tokens са истински и тогава осигурява достъп до локални ресурси

Когато се логвате в уеб приложение с windows акаунт, Security Token Service application конвертира windows акаунта до claims identity.

Когато трябва да се конфигурира връзка между SharePoint ферма и друг сървър, например Office Web Apps Server или Exchange Server, трябва да се конфигурира trust relationship между SharePoint STS и STS на другият сървър.

Windows Claims-mode аутентикация

- Claims-mode аутентикацията може да поддържа различни методи за Windows аутентикация
    - Integrated Windows аутентикация (NTLM or Kerberos)
    - Basic аутентикация
    - Digest аутентикация
    - SharePoint конвертира Windows идентификация в claims-based идентификация
- Към всеки тип аутентификация може да се добави и достъп за анонимни потребители

Това е методът за аутентикация, който се използва по подразбиране когато се създава нов web application.

Когато потребител разглежда web application, който използва Windows claims-mode логин, процесът на аутентикация работи по следният начин:

Планиране на NTLM и Kerberos аутентикация

- NTLM е:
  - Утвърдена
  - Лесна за конфигуриране
- Kerberos е:
  - По-ефективна
  - По-сигурна
  - Ако се използва Kerberos някои service applications изискват употребата на ограничено делегиране

NTLM аутентикацията е създадена преди повече от 15 години и е най-доказаната в продуктите на Microsoft. Когато потребител се логне, потребителското му име се изпраща към domain controller-а но не и паролата. Използва се еднопосочен криптиращ алгоритъм и специален протокол за криптиране и декриптиране.

Kerberos е методът за аутентикация по подразбиране между windows клиенти и сървър и Active Directory domain.

Модул 21: Интегриране на търсене в уеб портал

- Традиционно търсене
- URLs
- Crawling
- Главни аспекти на търсенето
- SharePoint 2013 Search Service
- Crawled и Managed properties

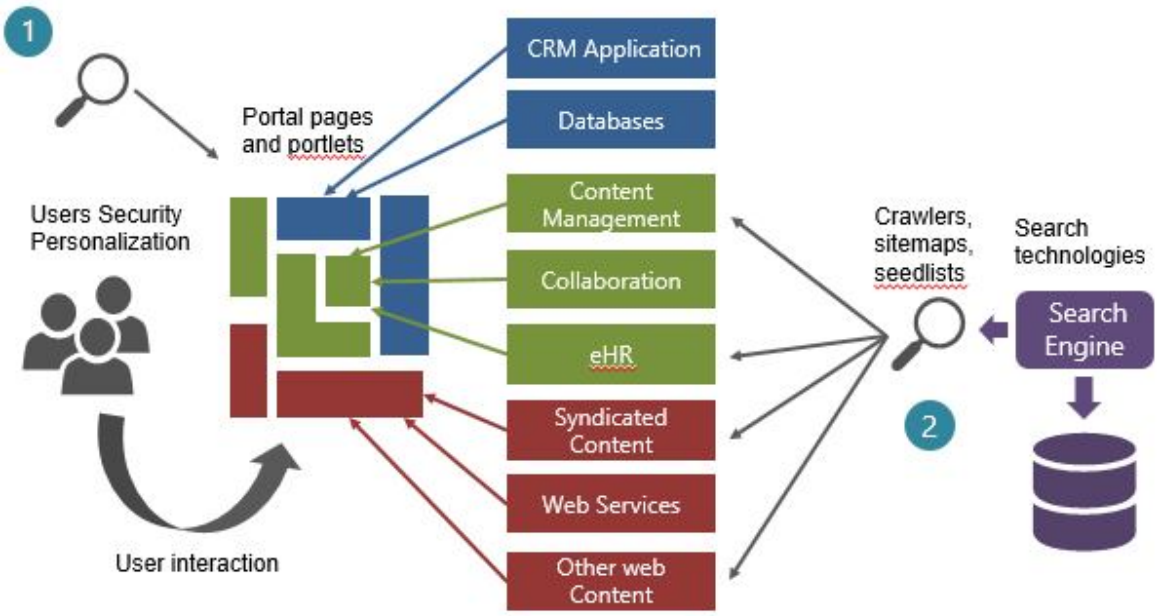
- Search Schema
- Езици за заявки
- REST заявки

Защо да интегрираме търсене

- Необходимост от търсене
  - Доставяне на информация за потребителя
  - По-удобен потребителски интерфейс
- Проблеми при реализацията:
  - Динамичното съдържание
  - Съдържанието, което е генерирано чрез JavaScript
- Необходимост от вътрешна система за портала, която да знае как и от къде да селектира резултатите

В днешно време информацията, която ни предоставя интернет нараства изключително динамично. Докато преди десет години преобладаваха статични уеб сайтове с оскъдна информация, днес съдържанието и услугите се предоставят от портали. С нарастването на количеството информация се поражда и проблема с намирането на най-подходящото съдържание. Именно за това е необходимо уеб порталите да поддържат функционалност за търсене.

Обикновените търсачки дават възможност за намиране на конкретна информация, но когато се използват различни нива на достъп външните сайтове-търсачки имат достъп само до публичната информация



Търсене и URLs

- Google Bot държи на кратки Universal Resource Locators (Уеб адреси, URL), докато информацията в портала може да се намира на всевъзможни места
- URL може да съдържа
    - Към коя страница да се навигира
    - На какъв език да се зареди страницата
    - Допълнителни параметри за показване на страницата
    - Допълнителна навигационна информация (например от коя страница се е стигнало до този линк)
- Много URLs могат да сочат към една и съща страница

При претърсването на интернет пространството често може да се случи така, че търсещата машина да достъпи определен ресурс от различни места и на различен адрес.

При предаване на данните от браузъра на сървъра се използва query string, чрез който може да се зададе например език, на който потребителя държи да се зареди определен ресурс или адрес от който е направен опит за достъп до ресурса или определени филтри. Това става чрез различни параметри. Важно е, когато търсачката обработва данните от търсенето тя да има предвид че намерените данни представляват един и същ ресурс, който се различава само по параметри, зададени при процеса на работа.

От друга страна подобна ситуация може да се получи и чрез съкратените URL адрес (friendly URL). Тогава в зависимост от структурата на портала и същността на представяната

информация, може да съществуват повече от един адреса, на които да бъде разположена абсолютно еднаква информация.

Crawling

- Търсенето в интернет се осъществява от специално разработен софтуер – web crawler(„бот“, „робот“, „паяк“)
- Различните търсачки използват различни алгоритми, които се усъвършенстват в бързина и точност
- Алгоритмите са много сложни, поради наличието на много критерии
    - Брой посещения, брой посещения през търсачка
    - Структура и архитектура на портала
    - Ключови думи
    - Честа обмяна на информацията
    - Дата на създаване на портала
    - Структура на страниците на портала
- Има няколко основни разлики между реален потребител и web crawler
    - Ботовете лесно се разпознават по тяхното ID
    - Реалните потребители първо задават какъв е техният предпочитан език
    - Много от ботовете имат проблеми при работата с JavaScript – могат да изпуснат дори генерирани препратки
- Търсене, интегрирано в портала отделно от публичните сайтове може да помогне да се решат тези проблеми

“Паякът” е софтуерна програма, която търсачките използват, за да намерят какво ново има в постоянно променящата се мрежа.

Използват се много видове паяци, но засега нас ни интересуват само тези, които “пълзят” за намиране на уеб страници. Това донякъде е доста опростена картина, но основно “паякът” започва от един уебсайт, зарежда страниците и следва хипервръзките от всяка страница.

По този начин, теорията казва, че всичко в интернет в крайна сметка ще бъде намерено, тъй като паякът лази от един сайт на друг. Търсачките могат да пускат хиляди уеб-паяци едновременно да пълзят на различни сървъри.

Когато "робота" посети някоя уеб страница, той запомня съдържанието й в база данни. Ако страницата бъде избрана, текстът от нея се зарежда в индекса на търсачката, който представлява огромна база данни от думи и къде те се намират по различните уеб страници.

Така че реално има три стъпки. Започва се с обхождане, след това индексиране, а последната стъпка е когато намерените линковете (адреси на уеб страници, URL) се пращат обратно при паяка, за да бъдат обработени.

Ако паякът (някои от тях проверяват и по-късно дали дадена страница наистина е офлайн) не намери дадена страница, тя в крайна сметка ще бъдат изтрита от индекса. Това е една от причините, поради които е важно да се използват надеждни уеб хостинг услуги.

Главни аспекти на търсенето

- Трябва да е интуитивно
    - Резултатите трябва да са възможно най-задоволителни още след първите въведени ключови думи
    - Метаданни
- Потребителите трябва да могат да филтрират резултатите
    - Обхват на търсенето
    - Видове данни – файлови типове
- Разширено търсене – много малък процент от потребителите го използват
    - Разрешаване на въвеждане на по-сложни заявки (булеви изрази)

Когато се реализира търсене в областта на определен портал или в интернет като цяло, трябва да се обърне внимание на няколко важни условия.

Реализираната търсачка трябва да бъде възможно „най-интелигентна", за да може да прецени кои резултати са по-важни за потребителя. Тя трябва да обръща внимание на различни критерии свързани не само със обхожданото съдържание, но и със смисъла на въведеният от потребителя текст.

Задължително е и резултатите да могат да бъдат филтрирани по различни критерии – търсен език, файлови типове и т.н. В някои случаи е необходимо да могат да се задават и по-сложни изрази, като например аритметични операции при извеждане на справка за отчет.

Search процеси

- Архитектурата на SharePoint има шест основни роли

- Администриране
- Crawling - Обхождане
- Content Processing - Обработка на съдържанието
- Analytics Processing - Анализиране на обработената информация
- Query Processing - Анализиране на заявките
- Index - Индексен файл

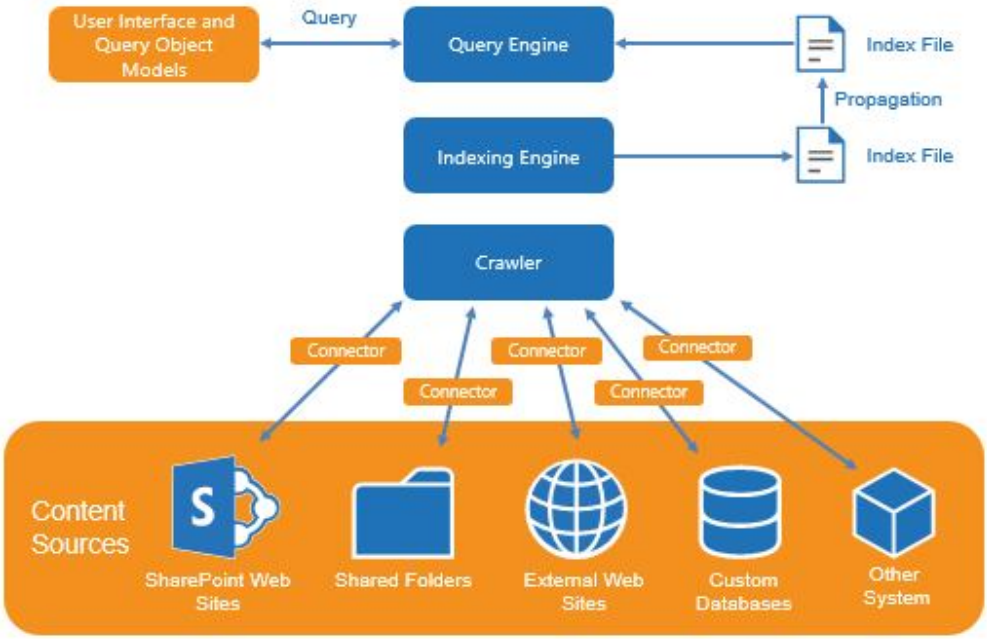
- Главните функционалности се изпълняват от следните процеси
  - NodeRunner.exe
  - HostControllerService.exe
  - Mssmdm.exe

При реализиране на SharePoint търсене от основна важност са няколко процеса. Първоначално трябва да бъде настроен Search Service Application за да се зададат основни параметри на търсенето – кои данни точно трябва да се търсят, допълнителни условия и т.н.

Задължителна процедура е обхождането на самите данни за да се структурират резултатите в Search Index за да са достъпни за по-бързо намиране.

Следва процедура по анализиране на данните спрямо останалата настройка на системата като права за търсене, посещаемост и интерес спрямо принадлежност към аудитории за да може да се определи важността на информацията за различните потребители.

Самите данни се извличат чрез заявки, които се обработват от отделен компонент.



Search Index

- Search Index се съхранява на файловата система
- Разделен е на части за да се подобри производителността:
  - http://www.spdockit.com/blog/sharepoint-2013-search-topologies-explained/#panel-2

Когато хората търсят във вашите сайтове на SharePoint, съдържанието на индекса за търсене определя какво ще намерят. Индексът съдържа информация от всички документи и страници в сайта ви.

Crawled Properties - данни за елемент от източника на информация или част от този елемент

- Примери
  - Колона с метаданни от List
  - Поле createdate от файловата система
  - Заглавието на документ
- Crawl пропъртитата се обработват от crawler-а и процесите за анализ на съдържанието
- Crawl properties са групирани по категории

Managed Properties - отделни crawl properties или съвкупност от такива, които са конфигурирани за специфични за търсенето цели

- В Search Index са включени само managed properties
- Пример
    - Crawl property name не е интуитивно и лесно за работа от потребител
    - Съвкупност от crawled properties от различни източници на данни трябва да бъдат обединени в едно managed property
    - Могат да бъдат използвани като Refiner
    - Могат да бъдат претърсвани лесно

Когато създадете нова колона на сайт в списък на SharePoint, интернет роботът взима името на колоната на сайт като ново crawled property. SharePoint автоматично съпоставя crawled property с ново managed property, което се генерира автоматично. По подразбиране това автоматично генерирано managed property не може да бъде използвано за уточняване.

Ако искате да използвате property като уточнение в клиентската част, трябва ръчно да съпоставите crawled property с managed property, което е зададено като годно за уточняване. За да създадете ново managed property за уточняване в SharePoint Online, трябва да използвате съществуващо, неизползвано crawled property и да го преименувате, като използвате псевдоним. Налични са достатъчно много managed properties за тази цел. Те имат имена като "RefinableString00" и "RefinableDate19."

Ако например сте създали нова колона на сайт, наречена NewColors, и искате потребителите да могат да използват NewColors като опция, когато уточняват резултатите от търсенето. В схемата на търсенето трябва да изберете неизползвано managed property, например RefinableString00, и да го преименувате на "New colors", като използвате псевдоним. След това трябва да съпоставите това ново managed property със съответното crawled property.

Search Schema

- Search Schema на SharePoint някои специфични детайли за конфигурацията на търсенето
    - Result Sources (Съвкупност от резултати)
    - Result Types (Типове резултати)
    - Managed Properties
- Search Schema на SharePoint 2013 е на няколко нива

- Service Application
- Site Collection
- Site

- Позволява промяна в различните нива в зависимост от нуждите

Индексът за търсене се изгражда чрез обхождане на съдържанието на вашия сайт в SharePoint. Интернет робот събира съдържание и метаданни от документите под формата на обходени свойства. Схемата на търсене помага на интернет робота да реши кое съдържание и кои метаданни да избере. Примери за метаданни са авторът и заглавието на документ. Въпреки това, за да могат съдържанието и метаданните от документите да бъдат включени в индекса за търсене, обходените свойства трябва да бъдат съпоставени сконтролирани свойства. Само контролираните свойства се запазват в индекса. Например обходено свойство, свързано с автор, се съпоставя с контролирано свойство, свързано с автор.

Full crawl (пълно обхождане)

- Ако incremental crawl е приключил неуспешно с грешки, за да се обнови индекса
- Ако се инсталира update или service pack
- Нов mapping(свързване) в Search schema
- Промяна на сайтовете и URLs които се търси

Когато промените crawled property или добавите ново, промените ще влязат в сила едва след повторното обхождане на съдържанието. В SharePoint обхождането се извършва автоматично въз основа на определения график за обхождане.

Когато добавите ново свойство към списък или библиотека или когато промените свойствата, използвани в списък или библиотека, съдържанието трябва да бъде обходено отново, за да могат промените ви да бъдат отразени в индекса за търсене. Тъй като сте извършили промените в схемата на търсене, а не в действителния сайт, интернет роботът няма автоматично да индексира отново списъка или библиотеката. За да се уверите, че промените ви са обходени и индексирани, можете да поискате повторно индексиране на списъка или библиотеката. Когато направите това, съдържанието на списъка или библиотеката ще бъде обходено и индексирано повторно, за да можете да започнете да използвате новите контролирани свойства в заявки, правила за заявки и шаблони за показване.

Incremental crawl

- Добавяне на документ в библиотека или елемент в списък

- Промяна на съдържанието на документ
- Изтриване на документ (ще се изтрие и от индекса)
- Промяна на права на елемент
- Properties на документ (columns)

При положение, че единственото, което се е променило в портала е актуалността на търсената информация може да се направи Incremental crawl на съдържанието. Неговото предимство е, че е по-бърз благодарение на това, че не прави промени в индекса и схемата на търсенето. Обикновено автоматичното обхождане на портала се реализира чрез incremental crawl, а full crawl се прави само ако има промяна по структурата на информацията

Езици за заявки в SharePoint

- SharePoint поддържа два езика за заявки
  - Keyword Query Language (Език за заявки по ключови думи, KQL)
  - FAST Query Language (FQL)
- KQL е езикът по подразбиране
  - FQL трябва да се активира ръчно
- Синтаксисът на SQL вече не се подържа
  - Това може да доведе до проблеми при миграцията на приложения написани на стари версии

Search APIs

- Има различни начини да се изпълни заявка в SharePoint
  - През потребителският интерфейс
  - През Web Services (Уеб услуги)
  - Server Object Model
  - Client Side Object Model
  - Representational State Transfer (разпределителна системна рамка, базирана на уеб протоколи и технологии, REST)
- Базират се на широко признати стандарти
  - Web services, HTTP, REST, .NET

След като данните за информацията в портала бъдат събрани и индексирани те трябва да бъдат извлечени по подходящ начин. Необходимо е да могат да бъдат извличани чрез различни технологии, за да могат да се използват на различни места в портала.

SharePoint 2013 е конструиран така, че да предоставя данните от търсенето чрез популярни стандарти за пренасяне на информацията. По този начин тя може да бъде достъпна от приложения, които са коренно различни като структура и технология.

Изпълняване на Web Service заявки

- Web Services трябва да се отбягват в SharePoint 2013
    - Все още се подържат, но по-добре да се използва друго API
- Използват /_vti_bin/spsearch.asmx
- Query и QueryEx методите изискват специален eXtensible Markup Language (разширяем маркиращ език, XML) форматиран параметър

SharePoint web services предоставят методи, които могат да се използват за отдалечена работа с отдалечена инсталация на SharePoint. Разработени са Web Services за аутентикация, достъп до данни от списъци, администриране, подаване на данни на Web Parts и т.н.

Достъпът до тези services става през релативен адрес /_vts_bin/

Web service методите, които се отнасят към SharePoint Search service са имплементирани в search.asmx

За повече информация:

1. https://msdn.microsoft.com/en-us/library/websvcsearch.queryservice_members(v=office.14).aspx

Изпълнение на Representational State Transfer (REST) заявки

- REST е прост протокол, който се базира на HTTP протокола
    - GET, POST. MERGE
- http://server/_api/search/query
    - Резултатите могат да са XML или JSON
    - Базирани са на Accept Header
        - Application/atom+xml
        - Application/json

REST протокола е изключително прост и популярен. Той представлява HTTP заявка към определен адрес, като чрез различни параметри в заявката може да се укаже на сървъра какво трябва да се търси. Чрез параметрите могат да се задава филтриране на резултатите по стойност и тип. Отговорът представлява XML или JSON структура

За повече информация:

1. https://msdn.microsoft.com/en-us/library/office/fp142380.aspx

2. https://msdn.microsoft.com/en-us/library/office/jj163876.aspx?f=255&MSPPError=-2147217396
3. https://sp2013searchtool.codeplex.com/ - графичен инструмент за изготвяне на REST заявки по зададени критерии

Модул 22: Предизвикателства пред съвместимостта на портали с множество мобилни устройства

- Възпроизвеждане на информацията
- Основни правила при определяне на дизайна
- Технологии
- Web Parts
- SharePoint Apps
- Device channels
- Master Pages
- Page Layouts

Въведение

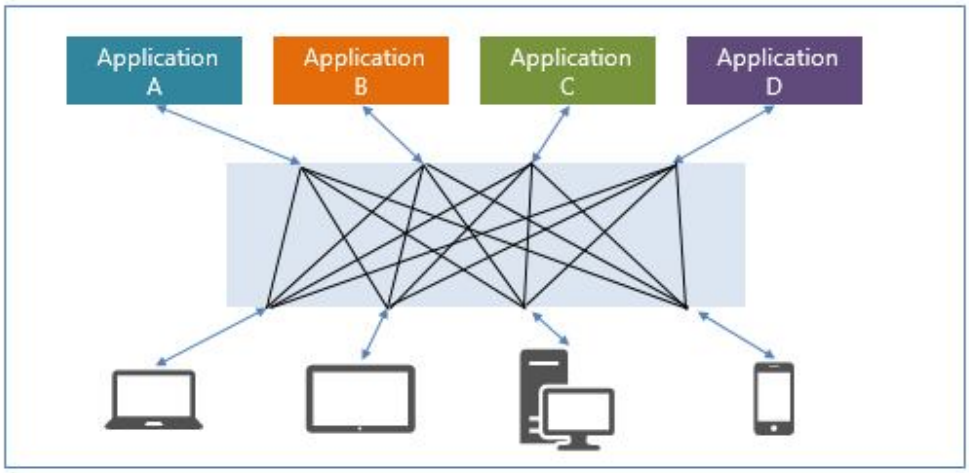
- Съвместимостта на портала с множество устройства - все по-актуален и важен за решаване проблем
- Разлики при устройствата
    - Големина на екрана
    - Формат на изображенията
    - Методи на въвеждане на информацията
        - Клавиатура
        - Екран

Когато потребителите използващи мобилни устройства достъпят сайт през смартфон или таблет, те виждат целия сайт събран и с много малки букви. За да прочетат нещо конкретно им се налага да приближат подобно на зумване на картинка и така те се насочват в определена част от страницата, без да виждат останалата част от нея. Малко вероятно е да се задържат и дори това да стане едва ли ще могат да видят, всичко което подготвено за тях на сайта. Ако има мобилна версия съдържанието на сайта, той се подрежда, така че да се вижда и чете достатъчно добре, независимо колко малко е устройството. Потребителя вижда цялата информация в сайта и лесно може само да скролира за да чете, като най-важните неща са подредени най-горе.

В последно време масово хората използват мобилни устройства като таблети и смартфони за да сърфират във Интернет. 45 процента от хората в България използват мобилни устройства и техния брой се увеличава много бързо. Предвижда се до 2017-та година над 5 милиарда души в света да използват мобилни устройства. Когато вашият уеб сайт е подготвен за мобилни устройства, шанса да спечелите клиенти използващи таблети и смартфони е много голям, тъй като все още малка част от уеб сайтовете са изработени по този начин. Мобилната версия е предимство и за доброто класиране в Google, като наличието на версия за мобилни устройства придава една идея на класирането спрямо истаналите сайтове. Да изключим тези неща чрез мобилната версия сайта изглежда много по-добре когато бива отварян от мобилни устройства.

N x M матрица

Когато се разработва съдържание, което би трябвало да бъде видимо през множество устройства, може да се използва логически слой, който да се използва само за интерпретация на информацията в различните случаи. Структурата трябва да позволява на различни приложения да комуникират с този слой, както и той да извежда информацията получена от приложенията на различни устройства.



За какво да се внимава

- Най-важно е да се съобрази потребителският интерфейс с екрана
- Мобилни устройства
    - Въпреки че е нужна по малко информация може да има забавяне заради мрежата на телекомуникационната компания
    - Изисква специално тестване на бързодействието
- Добра практика е отделянето на логиката на приложението от неговият изглед

Една от основните разлики между стационарните компютри и мобилните устройства е големината на екрана. Когато се разработва мобилна версия за уеб ресурс това трябва да бъде първият проблем за решаване, но не и последният. Въпреки бързото развитие на технологиите мобилните устройства все още разполагат с ограничени хардуерни възможности спрямо стационарните компютри. За това е необходимо специално внимание при изработването на скриптове с много изчисления, които могат да натоварят телефона.

Удобство при достъп

- Удобството при достъп до информацията е от съществено значение
    - Мобилни телефони
        - Дължина на URL
        - Навигация
            - Баланс между линкове и дължина на URL
            - Броя на кликовете да е максимум три
            - Допустимо е да има и shortcuts
    - Различен подход при процеса на навигация и при показването на информация
    - Задължително скролиране само в една посока – хоризонтално или вертикално

При разработването на приложението предварително трябва да се обмисли и каква информация ще бъде показана. Може да се наложи промяна на навигационната структура или дължината на адресите на страниците в портала. Навигационното меню може да бъде изнесено в бутон и да се визуализира на определен екран.

Хубаво е да се спазва правило за достъпност на ресурсите – всеки ресурс да се намира на удобно място, дори и да са необходими повече от 3 кликвания за да бъде достигнат.

Използване на ресурсите

- Трябва да се има предвид при разработването на версии за мобилни устройства
    - Използване на прекалено много памет
    - Големината на самото приложение
    - Footprint
- Батерия
    - Ако устройството е мобилно трябва да се избягват сложни изчислителни процеси с JavaScript

Тежките изчисления, реализирани на JavaScript се изпълняват изцяло при клиента. Трябва да се има предвид, че мобилните устройства имат както по-малко памет и изчислителна мощност, така и по-малко налична електроенергия.

Web Parts

- Няма изисквания към инфраструктура на портала при създаването
- Може да извлича или да обменя данни с други Web Parts
- Инсталира се на сървъра чрез .webpart файл
- В някои случаи е препоръчително да бъде заменено с AppPart
- Visual Web Parts
    - Наследяват нормалните Web Parts
    - Дефинират потребителски интерфейс чрез custom control (.ascx файл)

Web parts представляват отделни части с функционалност, които могат да се преизползват в различни страници.

Ако сте собственик на сайт или администратор, можете да персонализирате своята страница с web parts по различни начини, като например да редактирате заглавната лента на страница с web parts, да добавите web parts, да персонализирате изгледите на web parts за списъчен изглед и да промените оформлението на страницата. След като добавите web parts към страницата си, можете да свържете web parts, за да създадете още повече решения по избор за своята страница.

Ако имате програма за уеб проектиране, която е съвместима с Microsoft Windows SharePoint Services, като например Microsoft Office SharePoint Designer 2007, можете да персонализирате допълнително оформлението на страницата си.

SharePoint Apps – приложения, които работят в SharePoint среда

- Отделени от приложенията
- Възможност за деплойване и изолиране на ниво сайт
- Три възможности за хостване:
    - SharePoint hosted
    - Windows Azure hosted
    - Provider hosted

Инсталиране на SharePoint Apps

- Възможности:
    - Потребителите инсталират apps директно
    - Потребителите изискват apps
    - Изискваните apps трябва да се добавят допълнително
- Лицензиране
    - Неограничено
    - На потребител

SharePoint магазинът е публичен пазар, до който можете да получите директен достъп от сайт на SharePoint и да закупите приложения от други разработчици за лично ползване или за нуждите на организацията ви. Съответното приложение за SharePoint е малко, лесно за използване и самостоятелно приложение, което изпълнява определена задача или отговаря на бизнес нужди. Можете да добавяте приложения към вашия сайт, за да го персонализирате с определена функционалност или за показване на информация.

Магазините за приложения на Office и SharePoint са незадължителни услуги, управлявани от фирмата Microsoft Corporation или свързаните с нея лица от някой от офисите на Microsoft по света. Наличните приложения в магазина се предоставят от различни издатели на приложения и се управляват от правилата и условията и декларацията за поверителност на издателя на приложението. Използването на всяко от тези приложения може да доведе до прехвърляне към, съхраняване или обработване на вашите данни във всяка страна, където издателят на приложението, свързаните с него лица или доставчици на услуги имат офис. Наличието на конкретни приложения и методите на плащане зависят от вашия регион и услуга. Можете да прегледате правилата и условията и декларацията за поверителност на издателя на приложението, преди да изтеглите и използвате тези приложения.

За преглед на налични приложения за инсталиране:

1. https://store.office.com/appshome.aspx?legRedir=true&CorrelationId=590b0a00-1c0b-43d8-b05d-fc727ff11595&ui=bg-BG&rs=bg-BG&ad=BG&productgroup=SharePoint

App Permissions

- Изисква се при инсталация
- Изисква се permission и област на действие:
- Възможни permissions
    - Read-only (Само четене)
    - Write (Запис)
    - Manage (Управление)
    - Full Control (Пълен контрол)
- Възможни области
    - Site collection
    - Site
    - List
    - Tenant

При инсталация на приложение се изисква аутентикация. Самите приложения могат да заявят определен достъп до определен ресурс на SharePoint проекта. Когато приложението се инсталира трябва да се прецени дали да му се осигури този достъп.

Когато се задават правомощия на SharePoint apps трябва да се определят точните права, от които приложението има нужда. Тези права могат да бъдат прилагани върху различни области в зависимост от нуждите.

Device Channels (Канали за Устройства)

- Същност на Device Channels
- Конфигуриране
- Асоцииране на Master Pages със Device Channels
- Device Channels панели

Device channels използват браузъра на потребителите за да определят кой master page да бъде зареден към страниците със съдържание. Често device channels се използват за да се

засекат таблети и телефони, за да може информацията да се визуализира подходящо като за устройство с тъч скрийн интерфейс. Тази функционалност е нова за SharePoint 2013

За повече информация:

1. http://social.technet.microsoft.com/wiki/contents/articles/23157.sharepoint-2013-device-channels.aspx - настройване на Device channel

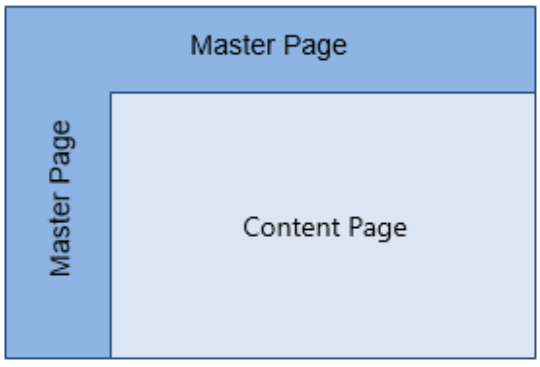
Master pages

- Какво е Master page
- Общ модел на презентиране на страници
    - Лого
    - Навигация
    - Търсене

SharePoint MasterPages осигуряват интерфейс и генерална конструкция нс различните страници в сайта. Общите елементи на страниците – header, footer, навигация и т.н. се разполагат на еднакви места и изглеждат еднакво в повечето страници.

Много страници използват еднаква структура и се различават само по съдържанието си. За това се разработва един master page и множество страници с отделно съдържание, като при визуализирането на всяка от тях се добавя и разработеният master page.

Masterpage може да се определи като контейнер на съдържанието на отделните страници. Потребителският интерфейс или изгледа на уеб сайта се променят през masterpages. Могат да се променят съществуващите master pages или да се разработват нови.



За повече информация:

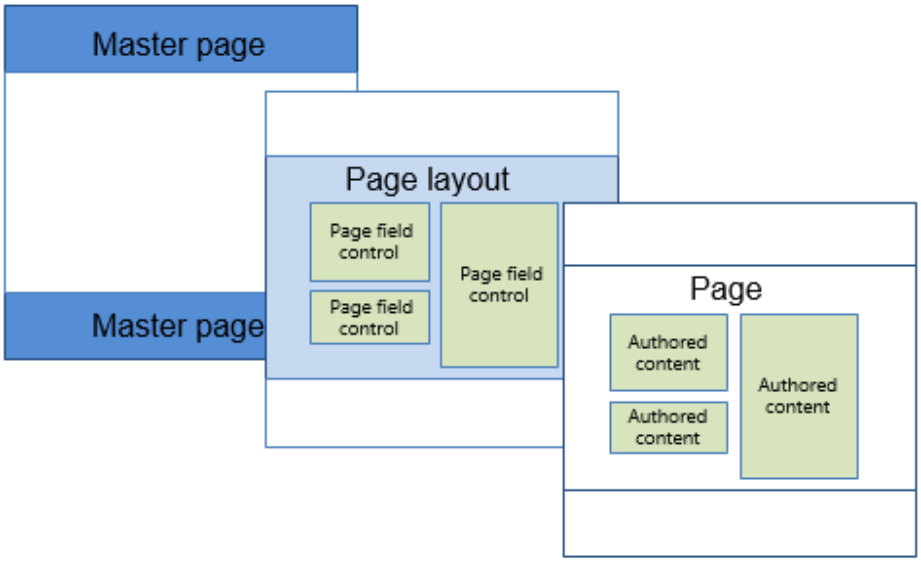
1. https://support.office.com/en-ca/article/Introduction-to-SharePoint-master-pages-dc9c4388-8dce-41b8-abb8-eeda2801b1a7

Page Layouts

- Дефинират изглед на страница
- Могат да изглеждат по различен начин
    - При въвеждане на данни
    - При преглед на данните
- Могат да бъдат създавани от разработчиците или да бъдат използвани готови
    - Catalog Item Image On Left
    - Basic Page
    - Webpart Page

В SharePoint Designer 2010 можете да създадете нова ASPX страница на базата на съществуваща страница образец. Ако изберете този метод за създаване на нова страница на сайта, трябва да изпълните няколко допълнителни стъпки, за да добавите съдържание към страницата. Трябва да намерите основния контейнер на страницата, в който можете да добавяте съдържание. След това трябва да добавите и персонализирате оформлението на областите на web parts, така че да отговарят на нуждите ви.

За да изпълните тези стъпки, разбира се, трябва да имате и необходимите права. Освен това създаването на нова страница на сайта по този начин води до създаване на съхранена в базата данни за съдържание страница, което е възможно да се стремите да избегнете поради проблеми с бързодействието.



Модул 23: Подобряване на подхода към проекти, свързани с Интернет и SOA

- Service Oriented Architecture(Архитектура ориентирана към услуги, SOA)
- SharePoint Service Applications
- Инстанции на Service Applications
- Service Application проксита

Възходът на интернет

- Интернет през 90-те
- Бурното развитие довежда до необходимост от промяна на традиционните методологии
- Отказване от някои Project Management технологии
    - PRINCE2
    - APMP
    - Just Do It подход на разработка на уеб сайтове

При настъпването на 21 век интернет пространството представляваше множество от отделни информативни сайтове, повечето от които разработени без сървърна технология. Порталите не оформяха голяма част от ресурсите в интернет и разработването на уеб продукти се свеждаше до кратко имплментиране и публикуване. В повечето случаи сайтовете бяха разработвани без да се обърне особено внимание на психологията на потребителите и повечето продукти бяха почти еднакви. Развитието на интернет пространството обаче наложи промяна на подхода при създаване на уеб решения.

Какво е SOA?

- Независими услуги, които могат да комуникират една с друга
    - Слабо свързани
    - С висока гранулярност
- Предоставяни и достъпвани чрез Service Bus SOA обещава:
    - По-голяма гъвкавост
    - По-ниски технологични разходи
    - По-добър отговор на изискванията на бизнеса

Ориентираната към услуги архитектура е начин за асемблиране на софтуерни приложения, инженеринг и бизнес процеси чрез свързване на софтуерни услуги. Програмистите разработват тези услуги използвайки традиционни езици като Java, C, C++, C#, Visual Basic, COBOL или PHP.

Сблъсквайки се с възрастта на старите проблеми на гъвкавост и сложност, IT директорите днес са под по-голям натиск от всякога, за да се подобри стратегическата стойност на IT за бизнеса. В най-добрия случай, тези предизвикателства са увеличените разходи, ограничените иновации и повишения риск. В най-лошия случай, те са намалили способността да се отговори на променящите се нужди на бизнеса по съвремеен начин.

Определение – "фундаментален начин на организация на система, предопределен от нейните компоненти, техните взаимоотношения както един към друг, така и към обкръжаващата ги среда, както и принципите, в съответствие с които се осъществява нейното проектиране и развитие".

Основни принципи: Разпределено проектиране. Решения за вътрешните особености на информационните системи се вземат от различни групи хора, имащи собствени организационни, политически и икономически основания. Постоянни промени. Отделните области на архитектурата могат да претърпяват изменения в който и да е момент от времето.

Последователно израстване (усъвършенстване). Локално подобрение на компонентите на архитектурата трябва да доведе до усъвършенстване на архитектурата като цяло – до ръст на сумарната полезност на компонентите от едно ниво, същото важи и за компонентите от по-високите или по-ниските нива.

SOA стандарти

На настоящия етап от своето развитие сервизно ориентираната архитектура използва за описание и организация на взаимодействието базовите стандарти на Web-услугите:

- XML - за предоставяне на данни;

- Web Services Definition Language (WSDL) - за осигуряване на достъп до Web- услугите;

- Universal Description, Discovery, Integration (UDDI) - за създаване на каталог на достъпните Web- услуги;

- Simple Object Access Protocol (Прост протокол за достъп до обекти, SOAP) – за обмяна на данни.

За повече информация:

1. http://nars.bg/wp-content/uploads/file/Nikolay-Manchev-SOA.pdf
2. http://tuj.asenevtsi.com/CNS/SEO%2012.htm

Преимущества на SOA

- Интеграция базирана на стандарти
    - Опростяване и поевтиняване на интеграционния процес
- Преизползваемост и комбиниране на приложения
    - Използване на съществуващи приложения за внедряване на нови бизнес услуги
- Миграция
    - Елегантна миграция на наследени системи у Автоматизация на процесите
- Внедряване на гъвкави процеси за поддържане на променящите се бизнес изисквания
    - Стандартизация
- Улесняване внедряването на стандарти в сливанията и придобиванията
    - Модернизация и съвместяване на IT системите след обединения

Jeremy Westerman формулира следното определение за сервизно ориентираната архитектура: “Това е парадигма, предназначена за проектиране, разработка и управление на дискретни единици логика в изчислителна среда. Прилагането на този подход изисква от програмистите приложенията да се проектират като поредица (набор) от услуги, дори когато преимуществата на такова решение не са очевидни. Програмистите са длъжни да излязат извън границите на своите приложения и да мислят за това как да използват съществуващите услуги или да проучват как техните услуги могат да бъдат използвани от колегите им.”

Едно от най-важните преимущества на архитектурата е нейната надеждност, особено когато става дума за нейната реализация на основата на Web-услуги и спецификации SOAP, които най-често използват протокол HTTP. Първоначално HTTP не е планиран, за да гарантира доставката, поради което не може самостоятелно да осигури изпълнението на критически важни транзакции.

Уроци и заключения от имплементации на SOA (Service Oriented Architecture)

В една компания може да цари убеждението, че има нужда именно от SOA, но това не означава, че наистина е в състояние да я внедрявате и развивате във вашата корпорация. Двата последни отчета на Центъра за изследване на информационните системи на Масачузетския технологичен институт („Архитектурата на ИТ като стратегия” и „Избор на

ИТ стратегия"), базирани на цяла поредица от проучвания и данни за 456 компании, откроява 4 етапа в развитието на ИТ архитектурата:

- Зоопарк

- Стандартизирани ИТ

- Стандартизирани бизнес процеси

- Трансформируем, „моделен" бизнес

Според проучването, за да се получи пълния обем от предимства, които предоставя SOA, е необходимо бизнес звената, както и самият IT отдел, да са преминали през всички етапи. Не би могло да се прескочи който и да е от тях. В най-добрия случай може да се премине по-бързо.

Според проучването по-голямата част от предприятията в САЩ (до голяма степен това важи и за предприятията в България) се намират между първата и втората фаза на развитие (а както вече стана дума, да се прескачат етапите не изглежда възможно).

SharePoint Service Applications

- Service Applications предоставят различни бизнес функционалности
- Компоненти за управление на service application
- Инстанции и компоненти на service
- Възможности за качване на множество service applications на различни сървъри

Service applications в SharePoint 2013 предоставят на потребителите множество функционалности. Те включват достъп до функционалностите на Word и Excel или сървиси като Manage Metadata Service или Business Connectivity Service. Архитектурата на service applications в SharePoint е така разработена, че архитектите могат да избират да включат в решенията само това, което им е необходимо. От базова гледна точка, service application има следните компоненти:

- Интерфейс който позволява администрирането

- Application pool

- База от данни

- една или повече физически инстанции

Функции на Service Application

- Често използвани service applications
- Нови
    - App Management Service
    - Translation Service
    - Work Management Service
- Отхвърлени
    - Web Analytics Service
    - Office Web Apps Service

Заедно с SharePoint 2013 се предоставят повече от 20 различни service applications. Някои от най-често използваните са:

- Search service – осигурява функционалност за търсене в съдържанието на портала
- Visio service – позволява на потребителите да визуализират диаграми на Visio
- Business connectivity service – осигурява достъп до данни от външни системи
- Managed metadata service – осигурява управлението на метаданни, term sets
- Secure store service – осигурява съхраняването на идентификации в база от данни

За повече информация:

1. http://go.microsoft.com/fwlink/?Linkid=299540

Инстанции на services и Service Application Dependencies

- Множество инстанции на някоя service application могат да се използват за:
    - Производителност
    - Капацитет
    - Service Management
- Service Application Dependencies

На сървърът могат да бъдат създадени множество инстанции на service application. Това дава възможност да се разпределят подобаващо ресурсите за всяка инстанция. Всяка инстанция от своя страна може да обслужва определени сайтове.

Сайтовете облуживани от определени инстанции могат да бъдат разделени на групи в зависимост от изискванията.

Създаване на Service Applications

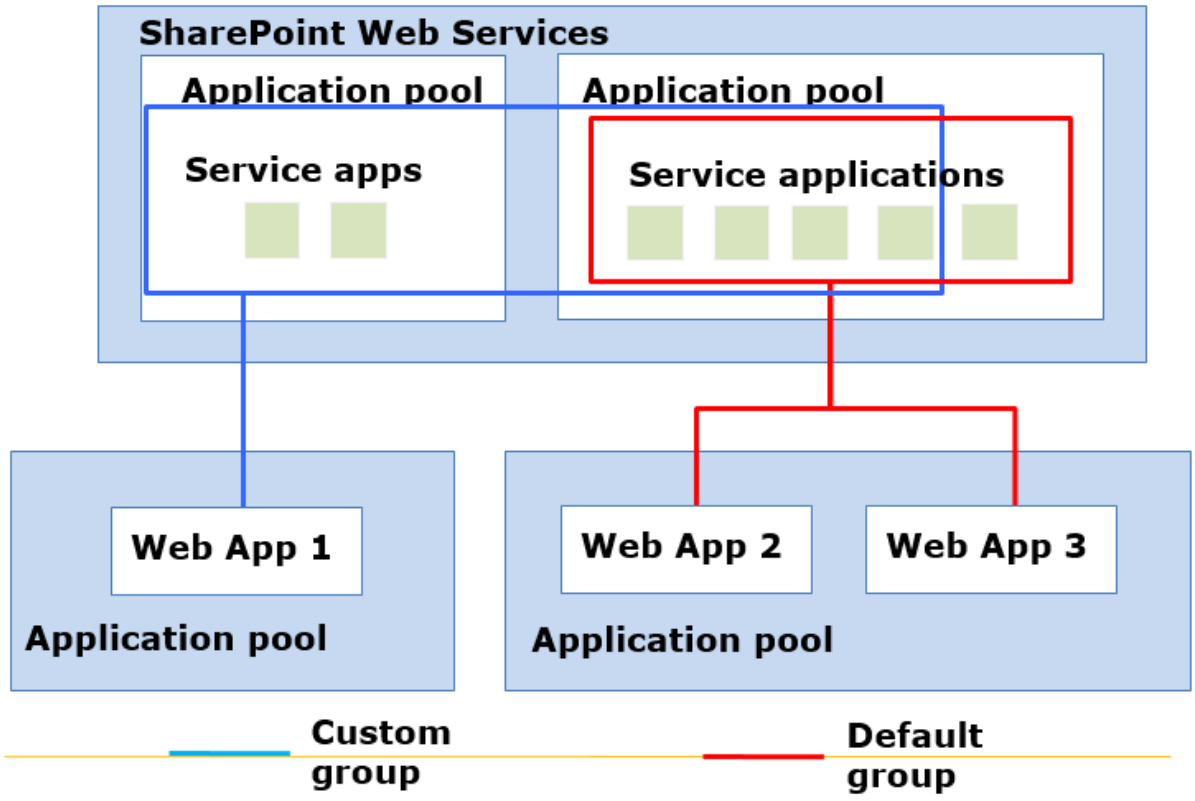
- Service Application може да се създаде и управлява чрез
    - Централната Администрация
    - Windows PowerShell
- Не всички Service Applications имат потребителски интерфейс
    - State service
    - Subscription settings service

Повечето от често използваните Service applications могат да се управляват през централната администрация.

Service Application проксита и прокси групи

- Service application connection (proxy)
    - Връзка между web application и service application
    - Service application е свързана със всички web applications на фермата по подразбиране
- Service application прокси групи
    - Групи по подразбиране
    - Създадени групи
    - Допълнително изолиране на service

Proxy групата определя група от service applications, които дадена web application има достъп. Относително проста е ситуацията, в която всички service applications се намират в една proxy група. Възможно е обаче и да съществуват множество прокси групи, които да се използват от различни web applications.



Стартиране и спиране на инстанции

- Стартиране и спиране на service applications е възможно през:
  - Потребителския интерфейст на централната администрация
    - Само за някои service applications
    - Farm Administrators SharePoint group
  - Windows PowerShell
    - Всички service applications
    - Sequrityadmin във SQLServer
    - Db_owner
    - Administrators група на cmdlet server

Модул 24: Контейнери и конектори като елементи от дизайна на портала

Web Applications

- Логическа структура на SharePoint
- SharePoint 2013 web applications
- Web applications и content databases
- Web applications и IIS уебсайтове
    - IP адрес
    - Порт
    - Host header

Web application представлява IIS уебсайт, който служи за логическо обединение на всички site collections които се създават в даден уебсайт. Следователно web application трябва да се направи, преди да се направят site collections. Когато се създава web application, той се представя чрез IIS уебсайт и конфигурационните данни се свързват с негов собствен или споделен application pool.

Логическа инфраструктура на web applications

- Аутентикация
    - Windows
    - Forms-based (Базирана на форми)
    - SAML token-based
- Методи на аутентикация
    - Claims-based
    - Classic-mode
- Web application и service applications
    - асоциирани чрез service application connection groups(също proxy groups)
    - Proxy groups съдържат application connections

Когато се създава нов web application се създава също и нова content database и се определят методите на аутентикация, които се използват за връзка с базата данни. Един web application може да използва много authentication providers.

Windows authentication – този метод използва съществуващият windows authentication provider, Active Directory Domain Services и Windows domain authentication протоколи

Forms-based authentication – този метод се базира на ASP.NET. Потребителите само попълват данни за себе си, като тези данни се пазят в база от данни.

Classic-mode authentication – базира се на windows security token, който се генерира от NTLM или Kerberos

Claims-based authentication – това е механизмът, който се използва по подразбиране в SharePoint 2013

Web Application зони

- Предоставят различни версии на един и същ IIS уеб сайт на различните потребители, използвайки различни URL-и
- Пет различни зони
    - Default, Intranet, Internet, Custom, Extranet
- Един web application може да бъде представен само веднъж в една зона
- Как се променя зоната

Зоните предоставят различни логически пътища (URLs) за достъп до един и същ web application. Различните зони предоставят възможност да се предостави различен достъп за различните потребители. Чрез зоните се могат да се използват различни методи за аутентикация за един и същ сайт

Всеки web application се създава в Default зоната. Ако възникне нужда се разширява и до другите зони. Една и съща зона не може да се използва два пъти в един и същ web application.

За повече информация:

1. https://technet.microsoft.com/en-us/library/gg276325.aspx

Управление на Content Databases

- Създаване на Content Databases
    - Когато се създава web application
    - New-SPWebApplication
- Добавяне на Content Databases
    - След като web application е създаден
    - New-SPContentDatabase
- Свързване или отвързване (Attaching или detaching) на content database

- Подобно на добавянето
- Mount-SPContentDatabase
- Dismount-SPContentDatabase

Има два варианта за добавяне на content dabases – добавяне на нова база или връзване към съществуваща. Добавяне на content dabases може да се реализира и през централната администрация и чрез Power Shell

За повече информация:

1. http://go.microsoft.com/fwlink/?Linkid=299609

Проектиране и конфигуриране на Managed Paths

- Explicit
  - http://intranet.training.com/
- Wildcard
  - http://intranet.training.com/sites/
- Конфигуриране на Managed Paths
  - Централна Администрация
  - New-SPManagedPath

Managed paths дават възможност на SharePoint да прецени къде могат да бъдат създавани Site Collections и да помогнат на системата да определи пътищата в сайта. SharePoint използва managed paths за да определи дали даден URL трябва да бъде обработен от него или не.

Explicit пътищата са точно въведени пътища, за обработката на които SharePoint поема грижата.

Wildcard пътищата определят началото на самият път, като по този начин на подобен адрес могат да бъдат създавани различнisite collections с различни имена.

За повече информация:

1. https://msdn.microsoft.com/en-us/library/office/bb802766(v=office.12).aspx

Resource Throttling

- Какво е Resource Throttling
- Проектиране и конфигуриране

- HTTP Request Monitoring и Throttling
    - Следи системните ресурси
    - определя рекуестите с нисък приоритет
- List throttling
    - List View Tresshold
    - Object Model Override

Ако някой от уеб сървърите се претовари с HTTP заявки, те се добавят на опашка от заявки. След време може обаче опашката да се препълни и сървърът да не може да отговори на някои от заявките.

Чрез Resource throttling може да се зададе на сървъра определени случаи при които той да отхвърля заявки с нисък приоритет.

Site Collection и Webs

- Site Collection
    - Контейнер за индивидуални сайтове
    - Подържа ниво на сигурност
    - Много от компонентите се разработват на ниво Site Collection
- Webs
    - Контейнер за списъци и библиотеки с документи
    - Контейнер за дъщерни Webs
    - Всеки Site Collection съдържа един основен Web

Колекция от сайтове е група от сайтове на SharePoint, които имат един и същ собственик и използват общи административни настройки, например разрешения. Колекциите от сайтове са йерархични и винаги включват един сайт от най-високо ниво и други сайтове под него.

Екипният сайт и другите сайтове, които създавате под него, са достъпни само за потребители, които каните, като им давате разрешение за сайта. С помощта на екипен сайт членовете на вашата организация, екип или група могат да се свързват помежду си и да си сътрудничат при работа върху документи и други файлове, да публикуват известия, да планират събрания, да поддържат задачи, да проследяват проблеми или елементи на действия, да съхраняват информация в списъци и др. В екипния сайт можете също да създавате различни подсайтове от налични шаблони на сайтове.

В зависимост от размера на вашата организация и обема на съдържанието, което планирате във вашите сайтове, може да искате да създадете подсайтове, за да организирате съдържанието. Сайтовете в колекция от сайтове са подредени в йерархия. Когато създавате сайтове под сайта от най-високо ниво, вие създавате тази йерархия. Можете да създавате подсайтове под вашия екипен сайт, както и да създавате допълнителни подсайтове под тези сайтове.

Има много възможни начини, които можете да изберете, за да организирате подсайтове. Например можете да изберете да създадете подсайтове:

По екипи или отдели

По функционално предназначение

По категории на съдържанието

По проекти

По клиенти

По нива на разрешение или конфиденциалност (ако например имате информация, достъпът до която трябва да бъде ограничен, може да искате да я изолирате в отделен сайт)

Настройване на Site Collections

- Site collections в контекста на сигурността и администрацията
- Site collection owners (Primary и Secondary)
    - Определени от администратора на фермата
    - Получават уведомления по email за събитията свързани с употребата на сайта
- Site collection administrators
    - Добавят се от други администратори или owners
    - Имат същите права като site collection owners

В голяма организация е нещо обичайно средата на SharePoint да включва няколко колекции от сайтове, като всяка има множество сайтове под себе си. В резултат на това е възможно да искате да делегирате управлението на сайтовете или колекциите от сайтове на различни хора в организацията. Моделът на разрешения в SharePoint ви помага, като ви позволява да присвоявате разрешения на различни роли, известни като администратор на колекция от сайтове, собственик на сайт и други роли, за които ще научите по-късно.

Като администратор на колекцията от сайтове вие имате ниво на разрешение Пълен контрол за колекцията от сайтове. Това означава, че можете да добавяте или изтривате сайтове или

да променяте настройките за всеки сайт в колекция от сайтове. Освен това можете да преглеждате, добавяте, изтривате или променяте цялото съдържание в тези сайтове.

Типични отговорности на администратора на колекция от сайтове

Ако сте администратор на колекцията от сайтове за вашия сайт на SharePoint, вашите отговорности могат да включват следното:

Да решавате кой може да има достъп до важно съдържание, съхранено в сайтове на SharePoint (за конфигурирането на разрешения на ниво колекция от сайтове).

Да решавате кои функции да направите достъпни за хората, които ще използват сайтовете във вашата колекция от сайтове.

- ✓ Да предоставяте техническа поддържа за хората, които използват вашата колекция от сайтове.
- ✓ Да избирате резервен администратор за вашата колекция от сайтове.
- ✓ Да създавате нови сайтове.
- ✓ Да преглеждате, изтривате или възстановявате елементи от кошчето на колекцията от сайтове.
- ✓ Да създадете и персонализирате публичния уеб сайт.
- ✓ Да помагате с администриране на определени функции, например:
- ✓ Да включвате или изключвате наличните функции на колекцията от сайтове.
- ✓ Да създавате или персонализирате типове съдържание на сайт.
- ✓ Да задавате регионалните настройки.

SharePoint Навигация

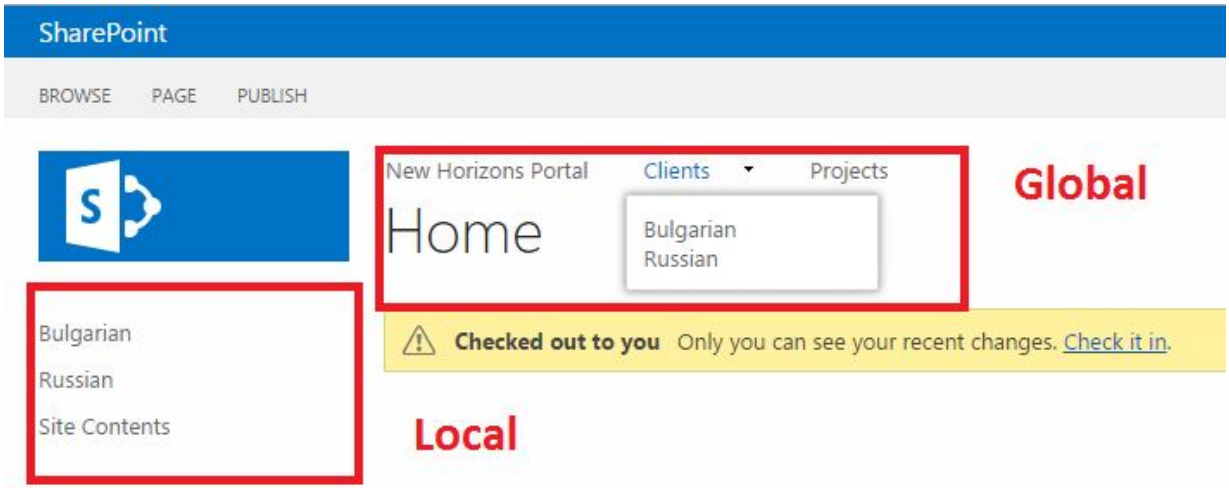
- Автоматична навигация, базираща се на архитектурата на Site collection
- Възможност за навигация базирана на Terms
- Поддръжка на Friendly URL
- Навигация на различни нива
    - Subsites

Трасирана навигация на съдържание Този елемент за навигация предоставя набор от хипервръзки, които разрешават на потребителите на сайта бързо да се придвижат към йерархията на сайтовете в колекцията със сайтове. Когато преминавате към йерархията на сайтовете, на страницата, до която сте се придвижили, се показва трасираната навигация на съдържанието. Тя не се показва, когато сте на началната страница на сайта.

Например ако отидете до списъка "задачи" на вашия сайт, отворете папката, наречена "Папка със задачи 1" и след това щракнете върху елемента, наречен "Задача 1", трасираната навигация на съдържанието показва следното Име на сайт > Задачи > Папка на задача 1 > Задача 1. Тъй като трасираната навигация е проектирана така, че да ви помогне да се

придвижите бързо до йерархията, "Задача 1" няма да е хипервръзка, защото вече виждате този компонент, но всички останали имена в трасирането са с хипервръзки. Може да щракнете върху всяка връзка в трасираната навигация, за да отидете до тази част на сайта.

Глобална трасирана навигация Този елемент за навигация предоставя хипервръзки, които може да използвате за връзка към различни сайтове в колекцията със сайтове. Винаги глобалната трасирана навигация е видима и се появява в горния ъгъл на страницата над името на вашия сайт. Първата връзка в глобалната трасирана навигация е връзка към сайта от високо ниво в колекцията със сайтове. Връзките към сайтовете, които наследяват горната си лента за връзки от родителския сайт, не се показват в глобалната трасирана навигация.



За повече информация:

1. https://support.office.com/bg-bg/article/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B8%D1%80%D0%B0%D0%BD%D0%B5-%D0%BD%D0%B0-%D0%BD%D0%B0%D0%B2%D0%B8%D0%B3%D0%B0%D1%86%D0%B8%D1%8F%D1%82%D0%B0-%D0%BD%D0%B0-%D0%B2%D0%B0%D1%88%D0%B8%D1%8F-%D0%B5%D0%BA%D0%B8%D0%BF%D0%B5%D0%BD-%D1%81%D0%B0%D0%B9%D1%82-3cd61ae7-a9ed-4e1e-bf6d-4655f0bf25ca?CorrelationId=ea3e2f40-fd12-4bbe-94eb-ad2b17f2f8d6&ui=bg-BG&rs=bg-BG&ad=BG

Site Templates (Шаблони на сайтове)

- Мощно средство за преизползване на различни по структура сайтове
- Персонализираните SharePoint сайтове могат да бъдат директно деплойвани

- Персонализираните SharePoint сайтове са лесно преносими в различни SharePoint среди
- Могат да бъдат допълнително променяни

Шаблоните за сайтове на SharePoint са предварително вградени дефиниции, разработени за определена бизнес нужда. Можете да използвате тези шаблони в готов вид, за да създадете собствен сайт на SharePoint, и после да персонализирате сайта по ваше желание.

Освен тези шаблони за сайтове по подразбиране, можете да създадете собствени шаблони за сайт въз основа на сайт, който сте създали и персонализирали. Шаблонът за сайт е начин да пакетирате функциите и персонализациите на сайта, за да можете да ги добавите в галерията с решения. Персонализираните шаблони за сайт често се използват за разполагане на решения в други сайтове или за осигуряване на съгласуваност на сайта в рамките на вашата организация. Може например да имате стандартни правила за управление на проектите и затова да изисквате за всички нови проекти да се използва шаблон за персонализиран сайт за проект.

Персонализираният шаблон за сайт е мощна функция, която ви позволява да създавате решения и после да споделяте решенията с ваши колеги, останалите в организацията или външни организации.

Модул 25: Подобряване на потребителското преживяване от портала

Потребителски профили в SharePoint

- Потребителските профили позволяват на SharePoint да поддържа и записва информация за потребителите
- Информацията за потребителите се използва за:
    - Индексиране на резултатите от търсенията
    - Подобряване на резултатите от търсенията
    - Да подбира съдържание за определена аудитория
    - Дава възможност на потребителите да търсят и да се свързват определени хора от организацията
    - Потребителските профили могат да се доразвиват:
    - Да се създават нови настройки в съответствие с нуждите на организацията
    - Да се свързва Active Directory с други бази данни с потребителски профили

- User Profile Service Application
- Осигурява мениджмънт и конфигурация на:
    - Потребителските профили
    - Импорт на потребителски профили от Active Directory
    - Audiences
    - My Site хост настройки
    - My Site website настройки
    - Social тагове и бележки
    - Права за персонализация на потребителите

Потребителският профил е колекция от потребителски свойства и правилата и настройките, свързани с всяко едно от тези свойства, които описват отделния потребител. Потребителските профили са важни, защото помагат на хората да намират съдържание, което ги свързва с други хора, да научават повече един за друг и да си сътрудничат чрез социални функции.

Свойствата на потребителските профили по подразбиране се въвеждат в SharePoint. Но администратор на SharePoint може да подсили възможностите на SharePoint, като добави свойства на потребителски профили, определи правила за потребителите и създаде аудитории. Функции като Информационен канал, Сайтове и Търсене на хора разчитат на потребителски профили, за да предоставят богати и персонализирани възможности за работа за хората в организацията ви.

Доброто управление на потребителските профили може да подобри комуникацията и работния процес в организацията ви. По същия начин използването на внимателно планирани правила за управление на тези профили ще гарантира, че съответната информация е видима за подходящите членове на организацията.

Планиране на импортиране на потребителски профили

- Планиране на компонентите на потребителският профил:
    - Изисквания
    - Метод на импортиране:
        - Импортиране
        - Синхронизация
    - Права

- Кои обекти да се синхронизират
- Свързване на настройките
- График на синхронизацията

Създаване на User Profile Service Application за импорт от Active Directory

1. Създаване на сервизен акаунт в AD DS
    - Даване на Replicating Directory Changes права
2. Създаване на web application за My Sites
3. Стартиране на инстанцията на User Profile Service
4. Създаване на User Profile Service Application
5. Настройване да се импортира от Active Directory
6. Създаване на връзка за импортиране
    - Указване на сервизният акаунт
7. Стартиране на синхринизацията

SharePoint получава информация за профилите по време на редовна планирана еднопосочна синхронизация, която може да се осъществява най-малко веднъж на всеки 24 часа.

Организацията ви може да избере да използва инструмента Office 365 Directory Sync (DirSync), за да попълни потребителска информация от локален указател Active Directory. DirSync поддържа външна еднократна идентификация. За повече информация относно справочната услуга на Office 365 прочетете Синхронизация на Active Directory в Office 365.

Конфигуриране на настройките на потребителските профили в SharePoint

- Типове данни
    - Числови
    - Да/не
    - Дата
    - Електронен адрес
    - HTML
    - SharePoint потребители
    - Текст

- URLs

Основните свойства на профила, като собственото и фамилното име, телефонния номер и длъжността на даден потребител, също се синхронизират. Ако има допълнителни свойства, които искате да добавите в потребителските профили, за да подобрите функциите за търсене и сътрудничество в SharePoint, то администраторът на SharePoint може да създаде тези свойства на потребителския профил.

Audiences (Аудитории)

- Употреба
    - Navigation links – препратки в навигацията
    - Web Parts
    - List content
    - Конфигуриране
- Поддръжка
    - AD DS group membership
    - AD DS manager attribute
    - SharePoint Online profile properties

Аудиториите представляват групиране на потребители, което можете да използвате, за да насочвате целево съдържание към хората във вашата организация. Групирането се определя от членството в групи за разпространение на Exchange, членството в групи на SharePoint или правила, конфигурирани от администратора на SharePoint с помощта на центъра за администриране на SharePoint. Администраторите на SharePoint могат да добавят, редактират и изтриват аудитории чрез центъра за администриране на SharePoint.

Аудиториите са дефинирани и се съдържат в сервизни приложения за потребителски профил.

Всяка аудитория трябва да бъде съставена, преди съдържанието да може да бъде насочено към нея. Съставянето идентифицира членството в аудитория чрез обхождане на последно получените данни от системата за управление на идентичност. В SharePoint Online съставянето на аудитория е конфигурирано предварително и се извършва на регулярни предварително определени интервали от справочния указател.

Таксономия в SharePoint

- Система от класификации

- В SharePoint таксономията се асоциира с метаданните

Таксономията е официална система за класификация. Тя групира думите, етикетите и термините, които описват даден обект, и след това подрежда групите йерархично.

Хората създават таксономии за почти всеки вид информация – от биологичните системи до организационните структури. Например биолозите класифицират живите организми в четири основни групи: животни, растения, гъби и микроби. Всяка от тези основни групи има множество подразделения. Заедно, цялата система представлява класификация.

Организациите създават огромен брой таксономии. Например сметкоплан – за управление на счетоводните системи, организационни диаграми и класификации на длъжностите – за управление на служителите, продуктови каталози и др. Всички тези класификации представляват йерархично структурирана информация – официални системи за класификация, които помагат на хората да работят с информацията.

Term Sets(Множества от търмове)

- Какво са term sets?
    - Списък от данни, които могат да бъдат асоциирани с дадена информация
- Term set терминология
    - Terms
    - Term set
    - Term set owner
    - Term set group
    - Term set group manager
    - Contributor
- Term set функционалност

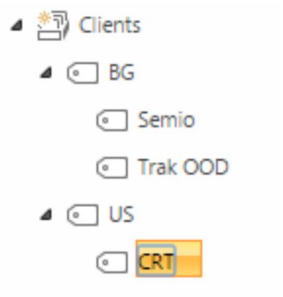
Наборът от изрази е група от свързани изрази.

Наборите от изрази може да имат различен обхват, в зависимост от това къде ги създавате.

Локалните набори от изрази са създадени в контекста на колекция от сайтове и са налични за използване (и видими) само за потребителите на тази колекция от сайтове. Когато например създавате набор от изрази за колона с метаданни в списък или библиотека, този набор от изрази е локален. Той е достъпен само в колекцията от сайтове, която съдържа този списък или библиотека. Например една библиотека с мултимедия може да има колона с метаданни, която показва вида мултимедия (диаграма, снимка, снимка на екран, видеоклип и т. н.). Списъкът с позволени изрази се отнася само за тази библиотека и е достъпен за използване само в нея.

Глобалните набори от изрази са налични за използване във всички сайтове, които са абонирани за определено сервизно приложение за управлявани метаданни. Например една организация може да създаде набор от изрази, който съдържа имената на нейните бизнес единици – "Човешки ресурси", "Маркетинг", "Информационни технологии" и т. н.

В допълнение можете да конфигурирате набор изрази като затворени или отворени. В затворен набор от изрази потребителите не могат да добавят нови условия, освен ако имат подходящи разрешения. В отворен набора потребителите могат да добавят нови условия в една колона, която е нанесена към набора от изрази.



SharePoint Managed Navigation (Управляема навигация на SharePoint)

- Елементи
    - Navigation Term Set
    - Target pages (Целеви страници)
    - Friendly URLs
    - Global Navigation и Current Navigation

На всяко ниво в йерархията можете да конфигурирате определени свойства за група, набор от изрази или израз, като използвате екрана "Свойства" в инструмента за управление на хранилище за изрази. Ако например конфигурирате набор от изрази, можете да посочите информация, като име, описание, собственик, контакт и заинтересувани лица, в екрана в раздел Общи. Можете също да зададете дали наборът от изрази да е отворен или затворен за нови подавания от потребители. Или можете да изберете раздел Планирани приложения и да зададете дали наборът от изрази да бъде достъпен за поставяне на етикети или навигация на сайта.

За повече информация:

1. https://technet.microsoft.com/en-us/library/dn194311.aspx

Използвана литература:

1. http://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D1%81%D1%8A%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B5%D1%82%D0%BE

2. http://bg.wikipedia.org/wiki/Joomla!

3. https://support.office.com/bg-bg/article/%D0%9A%D0%B0%D0%BA%D0%B2%D0%BE-%D0%B5-SharePoint-97b915e6-651b-43b2-827d-fb25777f446f?ui=bg-BG&rs=bg-BG&ad=BG

4. http://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%BE_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%81%D1%82%D0%B2%D0%BE

5. http://tuj.asenevtsi.com/InfResurs/InfResurs19.htm

6. http://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D1%81%D1%8A%D0%B4%D1%8A%D1%80%D0%B6%D0%B0%D0%BD%D0%B8%D0%B5%D1%82%D0%BE

7. https://support.office.com/bg-bg/article/%D0%A1%D1%8A%D0%B7%D0%B4%D0%B0%D0%B2%D0%B0%D0%BD%D0%B5-%D0%BD%D0%B0-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B8-%D0%B2-%D1%81%D0%B0%D0%B9%D1%82-%D0%BD%D0%B0-SharePoint-0ee31678-ee55-40cd-9f48-ea8700035dbd?ui=bg-BG&rs=bg-BG&ad=BG

8. http://bg.wikipedia.org/wiki/%D0%98%D0%B7%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2_%D0%BE%D0%B1%D0%BB%D0%B0%D0%BA

9. https://support.office.com/bg-bg/

10. https://bg.wikipedia.org/wiki/HTTP