Практический курс тестирования программного обеспечения

Урок 7

Test Club 2014

http://www.testclub.com.ua

План занятия:

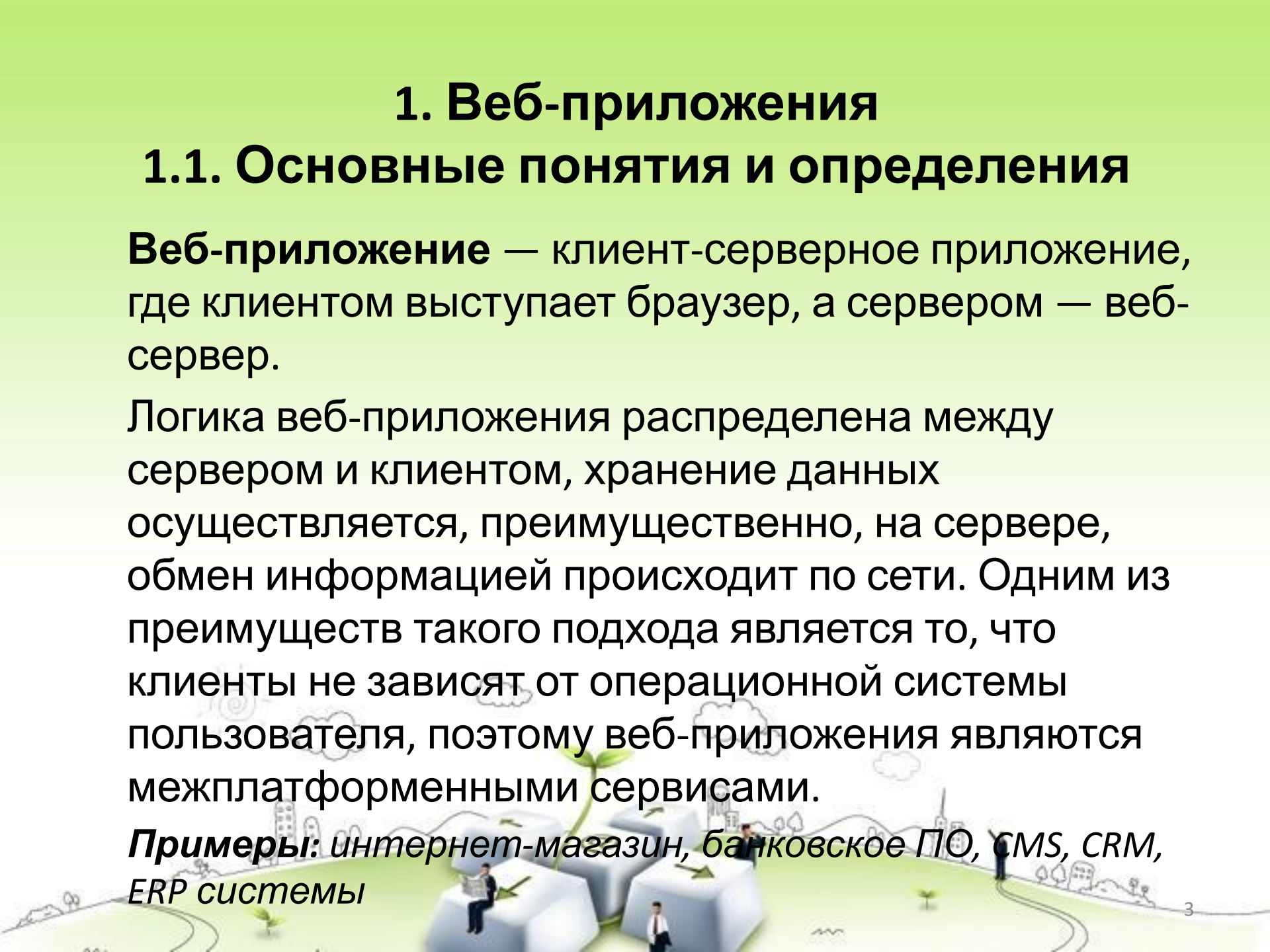
1. Веб-приложения
   1.1. Основные понятия и определения
   1.2. 3-х уровневая архитектура веб-приложения
   1.3. Преимущества веб-приложений
   1.4. Используемые технологии
   1.5. Веб-сервисы
2. Основные функциональности для проверки
   2.1. Functionality testing
   2.2. Usability and Interface testing
   2.3. Compatibility testing
   2.4. Security testing

1. Веб-приложения
1.1. Основные понятия и определения

Веб-приложение — клиент-серверное приложение, где клиентом выступает браузер, а сервером — веб-сервер.

Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является то, что клиенты не зависят от операционной системы пользователя, поэтому веб-приложения являются межплатформенными сервисами.

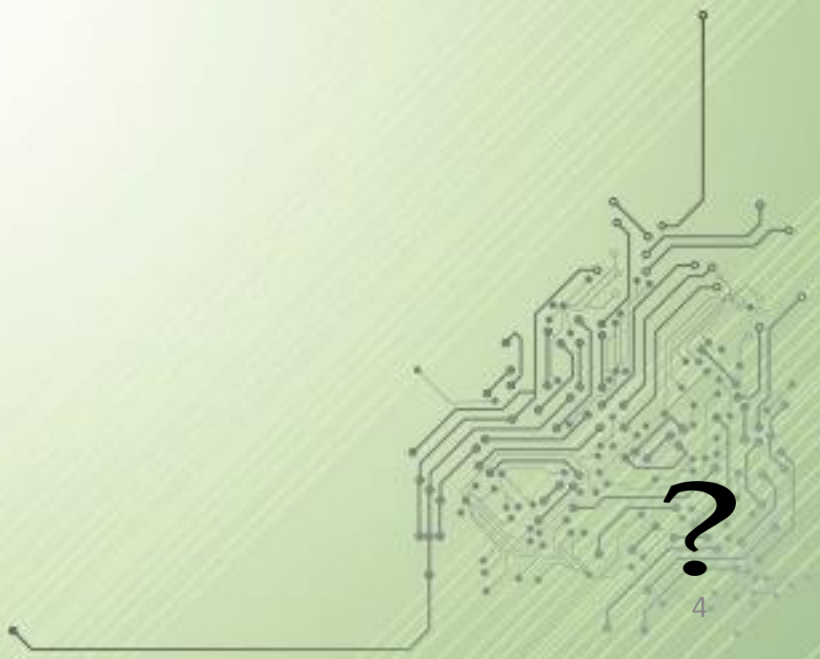
Примеры: *интернет-магазин, банковское ПО, CMS, CRM, ERP системы*



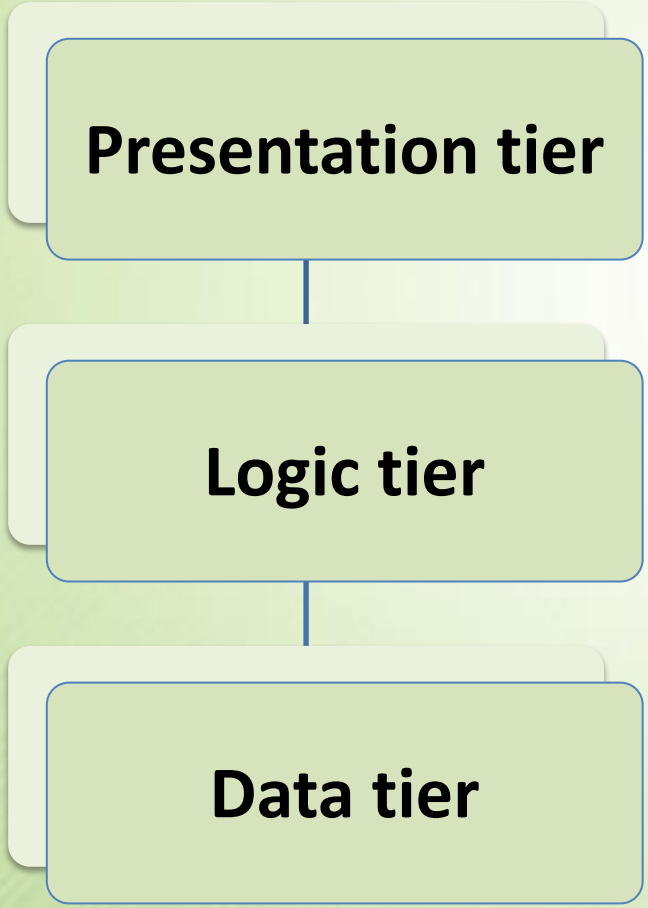
1. Веб-приложения
1.1. Основные понятия и определения

Веб-сервер — это сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными. Веб-серверы — основа Всемирной паутины.

Примеры: Apache Tomcat, IIS.



1. Веб-приложения
1.2. 3-х уровневая архитектура веб-приложения



The top most level of the application is the user interface. The main functions of the interface is to translate tasks and results to something user can understand.

Coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Here information is stored and retrieved from DB or file system. The information is passed back to the logic tier for processing, and then eventually back to the user.

1. Веб-приложения
1.2. 3-х уровневая архитектура веб-приложения

1. **Клиент** — это интерфейсный (обычно графический) компонент, который представляет первый уровень, собственно приложение для конечного пользователя.
2. **Сервер приложений** располагается на втором уровне. На втором уровне сосредоточена бóльшая часть бизнес-логики.
3. **Сервер базы данных** обеспечивает хранение данных и выносится на третий уровень. Обычно это стандартная реляционная или объектно-ориентированная СУБД.

?

1. Веб-приложения
1.3. Преимущества веб-приложений

1. Расположение приложения в одном месте

Основное преимущество веб-приложений в том, что все логические модели и алгоритмы расположены на сервере, в отличие от обычного ПО, в которых вся программная логика располагается на компьютере каждого пользователя. Так как имеется только одна рабочая копия приложения, его намного проще распространять среди пользователей. Пользователю не нужно тратить ресурсы своего компьютера для выполнения обработки результатов, с которыми работает программа.

Пользователь получает только то, что ему нужно для работы – интерфейс. Все остальное выполняется на сервере.

1. Веб-приложения
1.3. Преимущества веб-приложений

2. Отсутствует необходимость устанавливать программы на компьютер

Все, что нужно пользователю, это запустить браузер и набрать адрес приложения. Работать с веб-приложением можно из любой операционной системы без необходимости что-либо устанавливать на компьютер.

Так как веб-приложение расположено на сервере, то устанавливать его и настраивать нужно только один раз. С точки зрения бизнеса намного выгоднее содержать небольшую команду программистов, работающую в одном месте над одним приложением.

1. Веб-приложения
1.3. Преимущества веб-приложений

3. Нет необходимости переустанавливать приложение на компьютере после выхода новой версии программы

Как только новая версия веб-приложения будет загружена на сервер, все пользователи автоматически получат ее. Им не понадобится ничего переустанавливать на своем компьютере. К тому же, у всех пользователей будет существовать только одна версия программы – самая новая.

1. Веб-приложения
1.4. Используемые технологии

HTTP (Hyper Text Transfer Protocol) — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате HTML, в настоящий момент используется для передачи произвольных данных).

Основа HTTP - технология «клиент-сервер». Предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

1. Веб-приложения
1.4. Используемые технологии

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (англ. Uniform Resource Identifier) в запросе клиента.

Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное.

1. Веб-приложения
1.4. Используемые технологии

HTTP Request:

GET /wiki/страница HTTP/1.1

Host: ru.wikipedia.org

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

1. Веб-приложения
1.4. Используемые технологии

HTTP Response:

HTTP/1.1 200 OK

Date: Wed, 11 Feb 2009 11:20:59 GMT

Server: Apache

X-Powered-By: PHP/5.2.4-2ubuntu5wm1

Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

(далее следует запрошенная страница в HTML)

1. Веб-приложения
1.4. Используемые технологии

Программа для прослушивания траффика – **Fiddler**

http://bit.ly/1jKA1UJ - download Fiddler

http://bit.ly/1nHYzig - information about Fiddler

Response codes:

http://bit.ly/1fiB8bl

OSI Model:

http://bit.ly/1kZ3cbA

1. Веб-приложения
1.4. Используемые технологии

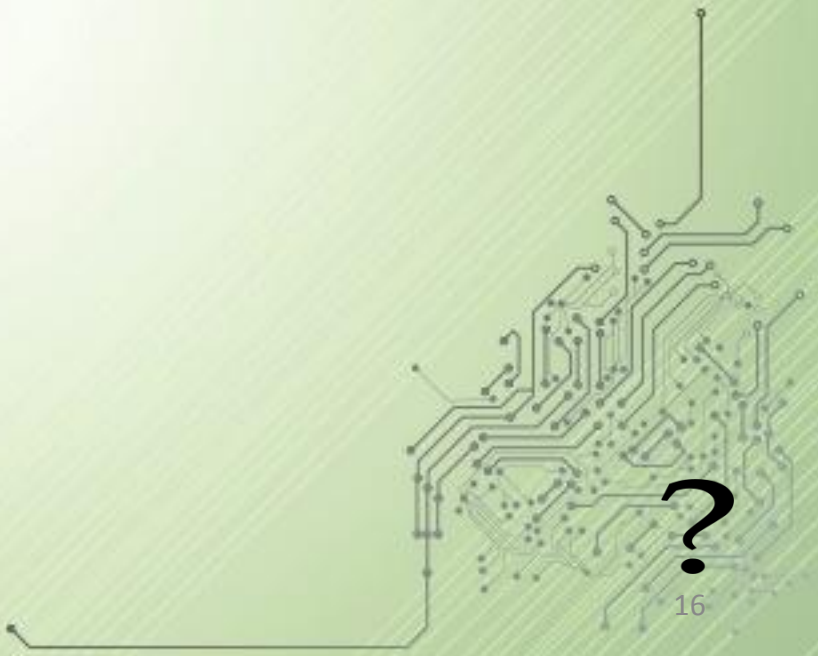
HTML (HyperText Markup Language — «язык разметки гипертекста») — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML).

Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

Основные элементы – открывающие и закрывающие теги http://bit.ly/1jKAhTw

1. Веб-приложения
1.4. Используемые технологии

XML (eXtensible Markup Language — расширяемый язык разметки). Используется для хранения и передачи структурируемых данных.
Больше про XML: http://bit.ly/1qtLQnn



1. Веб-приложения
1.5. Веб-сервисы

Веб-сервисы (Web services)— это реализация абсолютно четких интерфейсов обмена данными между различными приложениями, которые написаны не только на разных языках, но и распределены на разных узлах сети.

Представляют собой надстройку надо протоколом HTTP.

Наиболее распространённые протоколы реализации веб-сервисов:

- SOAP (Simple Object Access Protocol)
- REST (Representational State Transfer)
- XML-RPC (XML Remote Procedure Call)

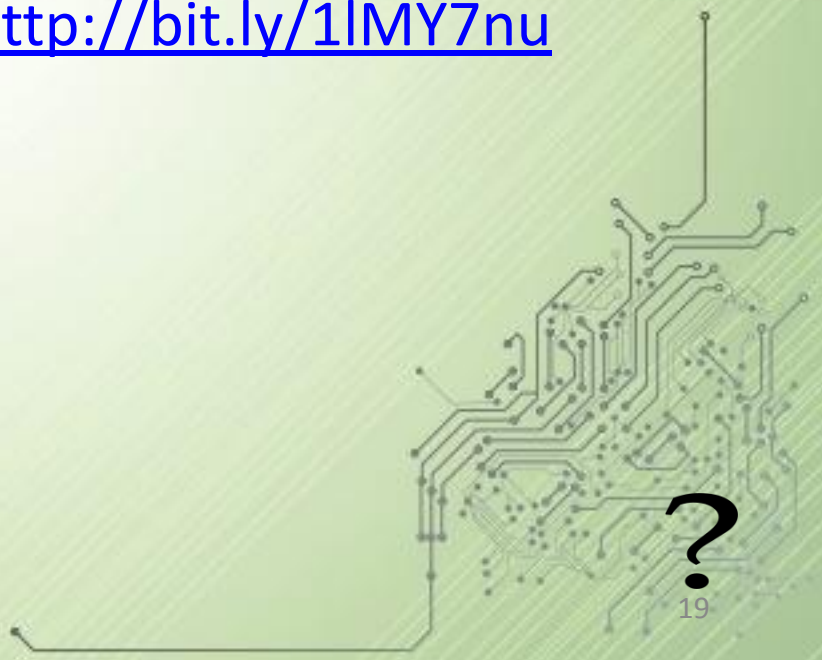
1. Веб-приложения
1.5. Веб-сервисы

Основная задача веб-сервисов - обеспечение межпрограммного взаимодействия. В отличие от традиционного веб-приложения, у веб-сервиса нет пользовательского интерфейса (GUI). Вместо этого у него есть программный интерфейс. То есть веб-сервис предоставляет функции (веб-методы), которые могут быть вызваны удаленно (например, по сети Internet). Веб-сервис не предназначен для обслуживания конечных пользователей. Его задача - предоставление услуг другим приложениям, будь то веб-приложения, приложения с графическим пользовательским интерфейсом или консольные приложения.

1. Веб-приложения
1.5. Веб-сервисы

Пример: Взаимодействие между авиакомпаниями и бюро путешествий. Первые предоставляют через веб-службы полезную информацию, которую вторые используют при поиске оптимальных предложений своим клиентам.

Детально про веб сервисы - http://bit.ly/1lMY7nu



2. Основные функциональности для проверки

Если необходимо протестировать Веб-приложение с нуля (частое практическое задание на собеседованиях), то следует принять во внимание следующую последовательность действий:

- Сделать запрос документов с требованиями
- Протестировать требования (урок 1)
- Сделать декомпозицию на логические модули
- В каждом из модулей проверить применимость видов тестирования

2. Основные функциональности для проверки

Целесообазно выделить такие виды тестирования:

1. Functionality testing
2. Usability and Interface testing
3. Compatibility testing
4. Security testing
5. Создать Traceability Matrix
6. Написать Test Case Headers

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.1. Positive tests for main business scenarios

2.1.2. Database connection

2.1.3. Forms used in the web page: submitting/getting information

2.1.4. All links in web pages – appearance, correct following

2.1.5. Cache / cookie testing

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.1. Positive tests for main business scenarios

Первым делом необходимо выполнить позитивные Smoke tests проверяющие основные сценарии использования приложения.

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.2. Data base connection

Написать CRUD tests, проверять их с двух сторон:

1. Действия на Front End -> Проверка через базу
2. Действия через базу -> Проверка через Front End

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.3 Forms used in the web pages: submitting/getting information

- Проверка валидации полей, кнопок, элементов GUI
- Проверка начальных значений
- Проверка некорректного ввода
- CRUD tests

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.4 All links in web pages – appearance, correct following

- создание чеклиста
- проход
- проверка переходов внутри страницы
- проверка отправки писем администратору
- поиск ссылок, которые никуда не ведут
- поиск битых ссылок

2. Основные функциональности для проверки

2.1. Functionality testing

2.1.5 Cache / cookie testing

Куки (cookie) — фрагмент данных, созданный веб-сервером или веб-страницей и хранимый на компьютере пользователя в виде файла, который веб-клиент (обычно веб-браузер) каждый раз пересылает веб-серверу в HTTP-запросе при попытке открыть страницу соответствующего сайта.

На практике обычно используется для аутентификации пользователя; хранения персональных предпочтений и настроек пользователя; отслеживания состояния сессии доступа пользователя; ведения статистики о пользователях.

2. Основные функциональности для проверки

2.1. Functionality testing

2.5 Cache / cookie testing

Кэш (cache) - промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Доступ к данным в кэше идёт быстрее, чем выборка исходных данных из оперативной памяти или загрузки из сети.

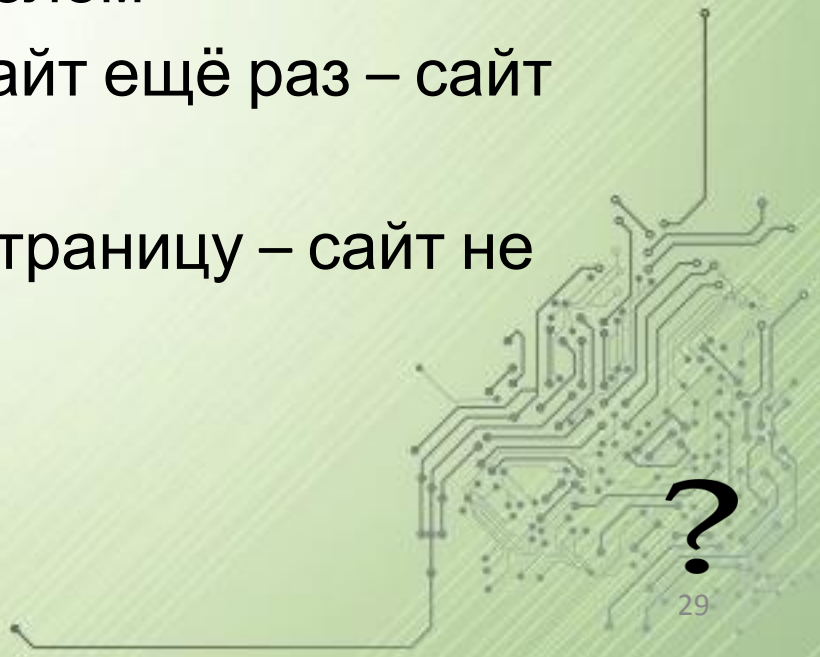
2. Основные функциональности для проверки

2.1. Functionality testing

2.5 Cache / cookie testing

Пример проверки:

1. Открыть сайт http://mail.ru/
2. Создать пользователя
3. Зайти под этим пользователем
4. Закрыть сайт и войти на сайт ещё раз – сайт помнит пользователя
5. Удалить куки и обновить страницу – сайт не помнит пользователя



2. Основные функциональности для проверки

2.2. Usability and Interface testing

Список для проверки:

- Структура страницы
- Цветовая гамма
- Наличие элементов интерфейса пользователя согласно документации
- Наличие и правильность контента
- Насколько удобно сделана навигация
- Насколько быстро можно найти нужное действие

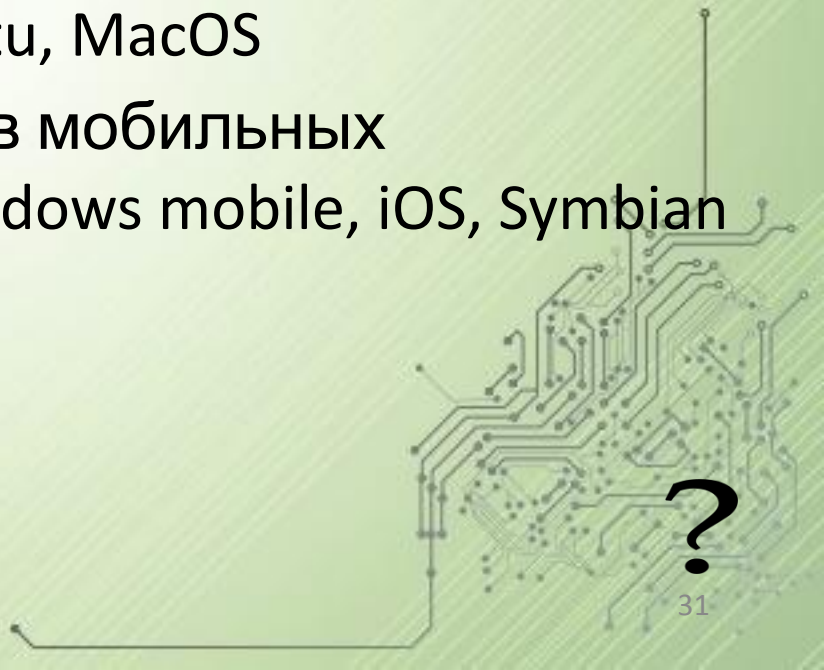
?

2. Основные функциональности для проверки

2.3. Compatibility testing

Список для проверки:

- Browser compatibility – проверка в браузерах IE (версии 8,9,10,11), FF (обычно от 13), Chrome, Safari
- Operating system compatibility – проверка на совместимость операционных систем – Windows XP, Windows 7, Windows 8, Ubuntu, MacOS
- Mobile browsing – проверка в мобильных приложениях – Android, Windows mobile, iOS, Symbian



2. Основные функциональности для проверки

2.4. Security testing

Список для проверки:

2.4.1. Test by pasting internal url directly into browser address bar without login. Internal pages should not open.
Пример: имея аккаунт на gmail –> выйти из своей почты и попробовать ссылку
https://mail.google.com/mail/?shva=1#inbox

2.4.2. Try some invalid inputs in input fields like login username, password, input text boxes. Check the system reaction on all invalid inputs

2. Основные функциональности для проверки

2.4. Security testing

Список для проверки:

2.4.3. If you are logged in using username and password and browsing internal pages then try to change url options directly.

I.e. If you are checking some publisher site statistics with publisher site ID= 123. Try directly changing the url site ID parameter to different site ID which is not related to logged in user. Access should denied for this user to view others stats.
http://job.ukr.net/?event=Login&Id=44994&hashKey=5a229b39af159877aff4bdf4fad91ad5&r=http%3A%2F%2Fjob.ukr.net%2Fpersonal%2Fvacancy%2Fnachinajuwij-testirovwik-testirovwik-junior-regular-senior-test-engineer-1186542%2F

2. Основные функциональности для проверки

2.4. Security testing

Список для проверки:

2.4.4 Web directories or files should not be accessible directly unless given download permission - отсутствие возможности зайти и посмотреть файлы сайта без соответствующих прав

2.4.5. Check that proper message is displayed when user switches from non-secure protocol to secure (http:// to https://) and vice versa – проверка, что для передачи данных кредитной карты используется правильный (секьюрный) протокол – и что эта передача не осуществляется, если протокол не секьюрный – изменение вручную

2. Основные функциональности для проверки

2.4. Security testing

Список для проверки:

2.4.5. Test the CAPTCHA for automates scripts logins – проверка, работает ли Каптча

2.4.6. All transactions, error messages, security breach attempts should get logged in log files somewhere on web server – проверка лог файлов на предмет того, что транзакции все записываются

Домашнее задание

1. Ознакомиться с отчетом о тестировании(доступен по http://bit.ly/1nI1IyS) сайта http://epicentrik.info/. Попробовать воспроизвести найденные дефекты.
2. Используя за основу отчет для сайта http://epicentrik.info/, провести цикл тестирования для сайта Portmone.com (http://bit.ly/1rY0sPO). Составить отчет о тестировании по аналогии с примером.

Также необходимо закоммитить созданный отчет в свою папку (“User*”) и отправить его по email.