

AJAX

Разработка web-приложений для Web 2.0

Игорь Борисов
<http://igor-borisov.ru>

Темы курса

- Основы AJAX приложения
- AJAX и HTTP
- Передача сложных типов данных. JSON
- Использование XML. XML-RPC
- Использование XML веб-сервисов. SOAP
- Безопасность и эффективность AJAX приложений

Лабораторная работа 0

Подготовка рабочего места

Упражнение 1: Создание виртуального хоста и запуск сервера

- Откройте проводник Windows
- Перейдите в директорию **C:\Пользователи\Общие\OpenServer\domains**
(Внимание! В некоторых ситуациях русскоязычному пути C:\Пользователи\Общие\ соответствует англоязычный путь C:\Users\Public\)
- В этой директории создайте папку **mysite.local**
- Запустите сервер. Для этого нажмите
[Пуск -> Open Server]
(На всякий случай, сама программа находится по пути C:\Пользователи\Общие\OpenServer\OpenServer.exe)
- В правом нижнем углу (рядом с часами) кликните по иконке с красным флагом
- В открывшемся меню выберите первый пункт **Запустить**
- Дождитесь пока цвет иконки с флагом изменится с желтого на зеленый
- Если запуск закончился неудачей - флагок опять стал красным, то кликните по иконке, выберите последний пункт **Выход** и повторите последние 4 пункта

Упражнение 2: Копирование необходимых файлов

- Получите у преподавателя архив с файлами для работы на курсе
- Скопируйте файл в созданную в предыдущем упражнении директорию **C:\Пользователи\Общие\OpenServer\domains\mysite.local**
- Распакуйте файл в **текущую** директорию
- Запустите браузер и в адресной строке наберите: **http://mysite.local/**
- Убедитесь, что сайт работает

Основы AJAX приложения

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

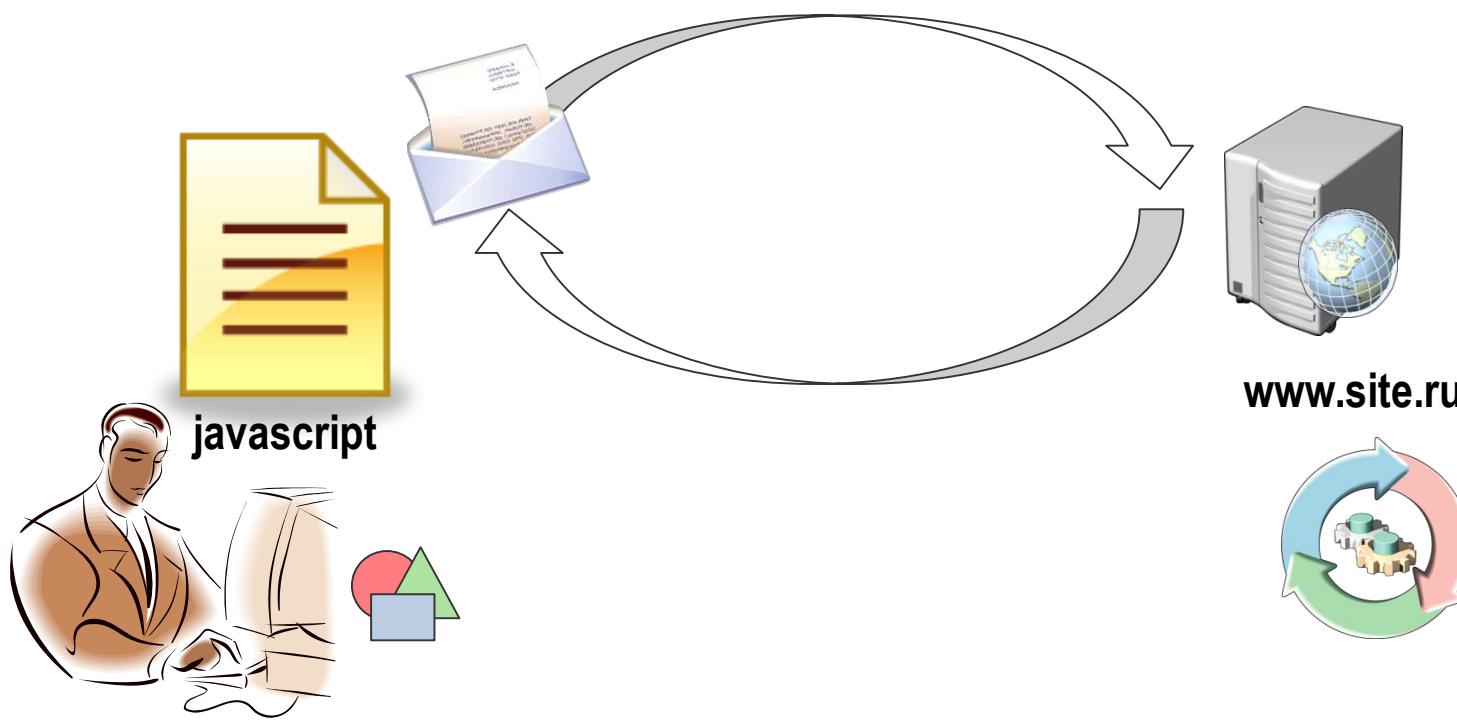
- Что такое AJAX?
- Механизмы взаимодействия с сервером
- Объект XMLHttpRequest
- Синхронные и асинхронные запросы
- Получение данных с сервера

Что такое AJAX?

- Asynchronous JavaScript And XML
 - Асинхронный JavaScript и XML
- Подход к построению интерактивных Веб-приложений
 - Обмен данными с сервером без перегрузки страницы в браузере
 - Использование DHTML для изменения контента на клиенте

Обмен данными с сервером

- Динамическое создание дочерних фреймов
- Динамическое создание элемента <script>
- Динамическое создание элемента
- Использование объекта XMLHttpRequest



Исторический экскурс

- 1996 HTML-элемент IFRAME в Internet Explorer
- 1998 г. Microsoft Remote Scripting
- 1999 г. Новая реализация MSXML и объект XMLHTTP
- 2000 г. Outlook Web Access
- 2005 г. Термин AJAX
 - [статья Джесси Джеймса Гарретта](#)
- 2005 г. Активное использование технологии компанией Google (Gmail, Google Maps)
- 2005 г. Термин Web 2.0
 - [статья Тима О'Рейли](#)

Объект XMLHttpRequest

- Большинство уверено, что это и есть AJAX
- Свойства
 - readyState
 - status
 - responseText
 - ...
- Методы
 - open()
 - send()
 - ...

Создание объекта XMLHttpRequest

- Ранние версии Internet Explorer
 - `new ActiveXObject("Microsoft.XMLHTTP");`
- Internet Explorer 6
 - `new ActiveXObject("Msxml2.XMLHTTP");`
- Все браузеры + IE7 и выше
 - `new XMLHttpRequest();`

Типы АJАХ-запросов

- Синхронный
- Асинхронный
- Контроль состояния
 - readyState
 - onreadystatechange

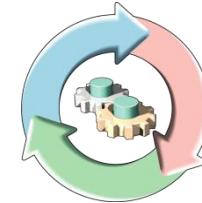
Синхронный запрос



javascript

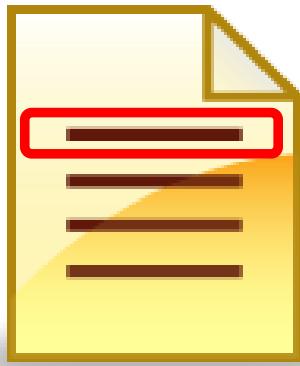


www.site.ru



- `request.open("method", "url", false);`
- `request.send(null);`

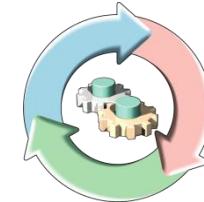
Асинхронный запрос



javascript



www.site.ru



- `request.open("method", "url", true);`
- `request.send(null);`
- `request.onreadystatechange = function(){};`
- `request.readyState === 4;`

Получение данных с сервера

- Свойства
 - responseText
 - responseXML
 - status
 - statusText
- Методы
 - getResponseHeader("header")
 - getAllResponseHeaders()

Лабораторная работа 1

Получение текстовых данных с сервера

Упражнение 1: Создание объекта XMLHttpRequest

- Откройте проводник Windows и перейдите в папку **labs**
- Создайте файл **xmlhttprequest.js**
- Откройте файл **xmlhttprequest.js** в текстовом редакторе
- Напишите кросбраузерную функцию **getXmlHttpRequest()**, которая возвращает объект **XmlHttpRequest**
- Сохраните файл **xmlhttprequest.js**
- Откройте в текстовом редакторе файл **labs\lab-1\index.html** и подключите к нему файл **labs\xmlhttprequest.js**
- Убедитесь работоспособности кода, открывая его в различных браузерах. Для проверки объекта выводите результат работы функции с помощью **alert()**
(Важно! Все пробы и тесты проводите только открывая файл с сервера, то есть, вводя в адресной строке адрес [http://mysite.local/...](http://mysite.local/), а не просто кликая по файлу!)

Упражнение 2: Асинхронный запрос к серверу

- Вернитесь к файлу **labs\lab-1\index.html**
- Найдите в блоке скрипта комментарий: **Задание 2. Выборка книги**
- Опишите функцию **getBookByNumber()**, которая **АСИНХРОННО** запрашивает нужную книгу по номеру и выводит строку с названием этой книги
 - Сформируйте запрос к серверу по адресу:
http://mysite.local/labs/lab-1/getbooktxt.php
 - Передайте этой PHP-странице параметр **num** со значением номера книги, например так:
http://mysite.local/labs/lab-1/getbooktxt.php?num=1
 - Полученный от сервера результат выведите в HTML-элемент **divResult**
- Сохраните файл и проверьте его работу, обращаясь к файлу лабораторной работы по адресу:
http://mysite.local/labs/lab-1/index.html

Упражнение 3: Подключение сценария к элементам страницы

- Вернитесь к файлу **labs\lab-1\index.html**
- Найдите в блоке скрипта комментарий: **Задание 3. Обработчик кнопки**

- Опишите функцию **showBook()**
 - Получите значение из HTML-элемента **txtNum**
 - Вызовите функцию **getBookByNumber()**, чтобы показать информацию о книге
- Проверьте работоспособность вашей страницы в разных браузерах

Выводы

- AJAX – механизм обмена данных
- Объект XMLHttpRequest осуществляет запросы
- Запросы могут быть синхронные и асинхронные
- Данные сервера отображаются на странице

AJAX и HTTP

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Методы передачи данных на сервер
- Передача простых данных методом GET
- Управление кэшированием ответа
- Метод HEAD
- Передача простых данных методом POST
- Получение и разбор комплексных данных

Взаимодействие: клиент - сервер

- HyperText Transfer Protocol (RFC 2616)
 - Заголовки запроса и ответа
 - Статусы ответа сервера
 - Методы HTTP
-
- <http://www.ietf.org/rfc/rfc2616.txt>

HTTP: терминология

- **GET /labs/lab-1/index.html HTTP/1.1** ↴
Host: mysite.local ↴
User-Agent: Mozilla/5.0 ... ↴
Accept: */* ↴
↳
- **HTTP/1.1 200 OK** ↴
Server: Apache/1.3.39 (Win32) PHP/5.2.6 ↴
Content-Length: 1801 ↴
Content-Type: text/html ↴
↳
<html>...</html>

Заголовки запроса

- **Host:** mysite.local
- **User-Agent:** Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
- **Referer:** http://mysite.local/labs
- **Accept:** text/html,
application/xhtml+xml, application/xml
- **Accept-Language:** ru,en-us
- **Accept-Charset:** windows-1251,utf-8
- **Keep-Alive:** 300
- **Connection:** keep-alive

Заголовки ответа

- **Date:** Sun, 15 Jun 2013 14:54:47 GMT
- **Server:** Apache/1.3.39 (Win32) PHP/5.2.6
- **Last-Modified:** Sun, 15 Jun 2013
14:00:06 GMT
- **Etag:** "0-709-48552066"
- **Accept-Ranges:** bytes
- **Content-Length:** 1801
- **Content-Type:** text/html
- **Keep-Alive:** timeout=15, max=100
- **Connection:** Keep-Alive

Статусы ответа сервера

- **200** – Успешно
- **301** – Постоянная переадресация
- **302** – Временная переадресация
- **304** – Объект не изменился
- **401** – Требуется авторизация
- **403** – Запрещено
- **404** – Объект не найден
- **405** – Метод не поддерживается
- **500** – Ошибка сервера

Передача данных методом GET

- <form action="server.php" method="get">
 <input type="text" name="user">
 <input type="password" name="pass">
 </form>
- GET server.php?user=root&pass=1234 HTTP/1.1
Host: mysite.local
User-Agent: Mozilla/5.0
Accept: */*

Управление кэшированием

- HTTP/1.x 200 OK

...

Cache-Control: no-store, no-cache, max-age=0

Expires: Sun, 15 Jun 2013 14:54:47 GMT

...

- Заголовок Cache-Control

- <IfModule mod_headers.c>
Header set Cache-Control "no-store, no-cache, max-age=0"
</IfModule>

- Заголовок Expires

- <IfModule mod_expires.c>
ExpiresActive On
ExpiresDefault "now"
</IfModule>

Метод HEAD

- **HEAD /labs/lab-1/index.html HTTP/1.1**
Host: mysite.local
User-Agent: Mozilla/5.0 ...
Accept: */*

■ **HTTP/1.1 200 OK**
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Content-Length: 1801
Content-Type: text/html

Лабораторная работа 2.1

Запрос простых данных методом GET

Упражнение 1: Запрос методом GET

- Откройте в текстовом редакторе файл `labs\lab-2-1\index.html`
- Найдите в блоке скрипта комментарий: **Задание 1. Список категорий**
- Опишите функцию `fillCategories()`, которая асинхронно получает данные из сценария `getcategories.php`
 - Этот сценарий возвращает список строк, разделенных символом конца строки `\n`, содержащих код и название категории
 - Формат строк: «**код:название**», например «1:Web»
 - Сформируйте из этих строк HTML-элементы `options` и добавьте их в HTML-элемент `selCategory`
- Поставьте вызов этой функции в событие `window.onload`, чтобы при загрузки страницы выпадающий список заполнился данными
- Сохраните файл и проверьте работу скрипта

Недостатки методов GET и HEAD

- Ограничена длина строки запроса
- По умолчанию – кэшируются!
- Запоминаются в истории посещений браузера и прокси-сервера
- Ни в коем случае не следует передавать методами GET или HEAD персональные данные или критичные к утечкам данные: логины, пароли, номера кредитной карты, номера телефонов, адреса и т.п.

Передача данных методом POST

- <form action="server.php" method="post">
- POST server.php HTTP/1.1 ↵
Host: mysite.local ↵
User-Agent: Mozilla/5.0 ↵
Accept: */* ↵
Content-Type: application/x-www-form-urlencoded ↵
Referer: http://mysite.local/demo/demo.html ↵
Content-Length: 19 ↵
Cache-Control: no-cache ↵
↵
user=root&pass=1234 ↵
↵

Разбор комплексных данных

- Передача строки с разделителями
 - Вася Пупкин:40:53:user
- Передача массива строк
 - Вася Пупкин:40:53:user ↪
 - Федя Сумкин:35:24:user ↪
 - Вова Морковкин:25:15:admin

Лабораторная работа 2.2

Использование методов POST и HEAD

Упражнение 1: Использование готового кода

- Откройте в текстовом редакторе файл **labs\lab-2-2\index.html**
- Найдите в блоке скрипта комментарий: **Задание 1. Код вставлять сюда**
- Скопируйте javascript-код из **Лабораторной работы 2-1** и вставьте его в файл после комментария
- Сохраните файл и убедитесь в работоспособности кода

Упражнение 2: Запрос методом POST

- Вернитесь к файлу **labs\lab-2-2\index.html**
- Найдите в блоке скрипта комментарий: **Задание 2. Выборка книг**
- Опишите функцию **showBooks()**, которая по выбранному пункту в списке **selCategory** формирует **POST** запрос к сценарию **postbooksbycat.php**
 - В этом запросе информация передается параметром **cat**, в котором число указывает код выбранной категории. Эти данные передавались в список на этапе выполнения **Лабораторной работы 2.1**
 - Сценарий **postbooksbycat.php** возвращает список книг указанной категории в формате:
автор | название | картинка
 - Получите список книг и выведите полученные книги в HTML-элемент **tableBooks**
- Поставьте вызов функции **showBooks()** на событие **onclick** кнопки [Показать]
- Сохраните файл и проверьте работу вашего скрипта

Упражнение 3: Запрос методом HEAD (Дополнительно)

- Вернитесь к файлу **labs\lab-2-2\index.html**
- Найдите в блоке скрипта комментарий: **Задание 3. Показ изображений**
- Напишите сценарий, который при щелчке по ряду таблицы с информацией о книге проверяет наличие файла с изображением на сервере и выводит это изображение в элемент **img**:

```
<div id="divBookInfo">
    <img src="" alt="" />
</div>
```

Изображения книг находятся в папке **/images**:
- Проверку наличия файла необходимо выполнять с помощью метода **HEAD**

- Сохраните файл и проверьте работу вашего скрипта
- Если остается время, допишите функцию проверки наличия файла так, чтобы в атрибут `alt` изображения вписывалась бы информация о размере файла изображения, которую вы можете прочитать из заголовка **Content-Length** при выполнении запроса методом **HEAD**

Выводы

- Простые данные могут быть переданы на сервер методами GET и POST
- Информация об объектах сервера может быть запрошена методом HEAD
- Результаты запросов GET и HEAD кэшируются браузером
- Кэширование управляется заголовками ответа Cache-Control и Expires
- У такого способа передачи данных на сервер есть недостатки

Передача сложных типов данных. JSON

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Недостатки простых текстовых форматов
- Сериализация сложных данных
- Что такое JSON
- Разбор JSON пакета в браузере
- Разбор JSON пакета на сервере (PHP)
- Получение данных с сервера

Сериализация и десериализация

- Сериализация
 - процесс преобразования объекта (структуры) в текстовое или бинарное представление данных
- Десериализация
 - процесс восстановления объекта (структурьи) из текстового представления или бинарных данных
- При сериализации сохраняются только свойства объекта, а не его методы

JSON

- JavaScript Object Notation
- Текстовый формат обмена данными, основанный на JavaScript
- Строится на двух структурах:
 - набор пар имя/значение.
 - пронумерованный набор значений.

Пример JSON-строки

```
■ {  
    "firstName": "Василий",  
    "lastName": "Пупкин",  
    "address":  
    {  
        "streetAddress": "Бакунинская, 71",  
        "city": "Москва",  
    },  
    "phoneNumbers":  
    [  
        "+7 (495) 780-48-48",  
        "+7 (495) 775-31-94"  
    ]  
}
```

Элементы JSON

- Объект
- Массив
- Значение
- Стока

Простая десериализация JSON

- ```
var jsonString =
'{
 "title" : "Книга",
 "author" : "Автор",
 "price" : 150
}';
```
- ```
var book = eval("(" + jsonString + ")");
```
- ```
alert(book.title);
```

# Использование JSON на клиенте

## ■ Проблемы

- Опасность простой десериализации
- Сложность сериализации

## ■ Решение

- <http://www.json.org/js.html>
- JSON.parse(строка)
- JSON.stringify(объект)

# Лабораторная работа 3

## Авторизация пользователя на сайте

### Упражнение 1: Форма авторизации пользователя

- Откройте в текстовом редакторе файл **labs\lab-3\index.html**
- Изучите HTML-код этой страницы:
  - Обратите внимание на форму логина (HTML-элемент **frmLogin**)
  - Эта форма не показывается, так как стилевой разметкой для нее установлено свойство **display:none**
  - Необходимо написать обработчик для кнопки формы [Вход] (найдите эту кнопку в HTML-коде)
- Найдите в блоке скрипта комментарий: **Задание 1. Отображение формы**
- Опишите функцию **showLoginForm()**, которая отображает форму входа на экране
  - Измените у формы значение CSS-свойства **display**
  - Поменяйте цвет фона всей страницы на серый
- Сохраните файл и проверьте работоспособность кода — правильное отображение формы при нажатии на кнопку

### Упражнение 2: Проверка пользователя

- Вернитесь к файлу **labs\lab-3\index.html**
- Обратите внимание на объявленную функцию-конструктор **UserInfo()**
- Найдите в блоке скрипта комментарий: **Задание 2. Проверка пользователя**
- Опишите функцию **validateUser()**
  - Считайте введенные пользователем данные
  - Используя функцию-конструктор **UserInfo()** создайте объект **userInfo** и заполните у него свойства **login** и **password**
  - Произведите JSON-сериализацию объекта **userInfo**
  - В асинхронном режиме передайте методом **POST** полученную строку серверу **user\_auth.php**  
(Не забудьте указать Content-type: text/plain )
  - В случае успеха сервер вернет вам JSON-строку с данными. Десериализуйте эту строку в объект и сохраните его в глобальную переменную **ticket**
  - Прочитайте у объекта **ticket** свойство **valid**

- Если свойство **valid** — **false**, то покажите пользователю сообщение об ошибке (HTML-элемент **divMessage**)
- Если свойство **valid** — **true**, то
  - спрячьте форму логина (HTML-элемент **frmLogin**)
  - поменяйте цвет фона всей страницы на цвет по-умолчанию
- Поставьте вызов функции **validateUser()** на событие нажатия кнопки формы логина (HTML-элемент **frmLogin**)
- Опишите функцию **hideErrorMessage()**, которая закрывает окно с ошибкой с помощью кнопки [Закрыть]
- Сохраните файл и проверьте работу скрипта. Для входа используйте следующие логины пользователей:  
**vasyap, fedias, vovam, galp, svetao.**  
У всех этих пользователей в качестве пароля выбрано слово **password**

### Упражнение 3: Список пользователей онлайн

- Вернитесь к файлу **labs\lab-3\index.html**
- Найдите в блоке скрипта комментарий: **Задание 3. Список пользователей**
- Опишите функцию **showOnLineUsers()**
  - Получите ссылку на HTML-элемент **UL**, который находится в HTML-элементе **divUsers**
  - Сериализуйте в JSON-строку полученный ранее билет **ticket**
  - В асинхронном режиме запросите сервер **get\_online\_users.php**, передав ему как обычный текст сериализованную строку
  - Получите ответ сервера и десериализуйте его в массив пользователей
  - Удалите все дочерние узлы для списка **UL**, на который вы получили ссылку
  - Проходя по массиву пользователей, добавьте информацию о пользователях в список **UL**
  - Установить таймер на вызов этой же функции в диапазоне 30 - 60 сек...
- Сохраните файл и проверьте работу скрипта в различных браузерах

# Выводы

- JSON – способ представления сложных структурированных данных
- JSON – строка текста, описывающая поименованные свойства, объекты и массивы
- Плюсы: простота, компактность, высокий уровень полезной нагрузки
- JSON строка может очень легко преобразовываться в реальные объекты

# Использование XML. XML-RPC

Игорь Борисов  
<http://igor-borisov.ru>

# Темы модуля

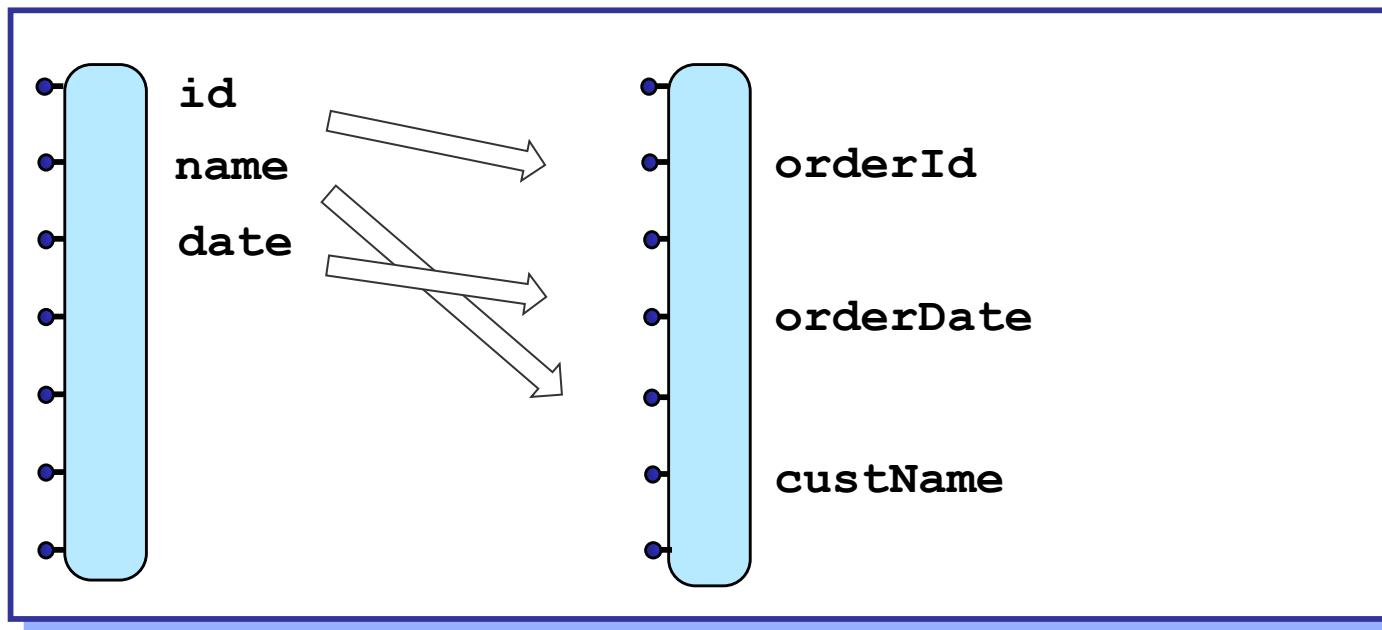
- Проблемы текстовых данных и JSON
- Другие способы передачи структурированных данных
- Обзор XML технологий
- Клиент-ориентированная и сервер-ориентированная архитектура
- Протокол XML-RPC
- Формирование XML-RPC запроса
- Преобразование XML данных

# Проблемы текстовых данных и JSON

- Отсутствие типов данных или их ограниченный набор – Что это?
  - “date” : “01-02-03”
  - “price” : “двести”
- Сложность контроля целостности данных – Как это проверить?
  - “author” : “12345678”
  - “price” : “200 рублей”

# Проблемы текстовых данных и JSON

- Сложность визуализации данных
- Сложность преобразования данных



# Форматы передачи данных

## ■ Текст

```
Василий Пупкин:vasya@mail.ru:46:15
```

## ■ JSON

```
{"firstName": "Василий Пупкин",
"email": "vasya@mail.ru",
"answers": 45,
"raiting": 15}
```

## ■ XML

```
<user name="Василий Пупкин"
 email="vasya@mail.ru"
 answers="45"
 raiting="15" />
```

# XML

- eXtensible Markup Language
  - расширяемый язык разметки
- Текстовый формат, предназначенный для описания, хранения и передачи структурированных данных

```
<?xml version="1.0" e
<quiz>
 <question>
 Who was the forty-second
 president of the U.S.A.?
 </question>
 <answer>
 William Jefferson Clinton
 </answer>
 <!-- Note: We need to add
 more questions later.-->
</quiz>
```

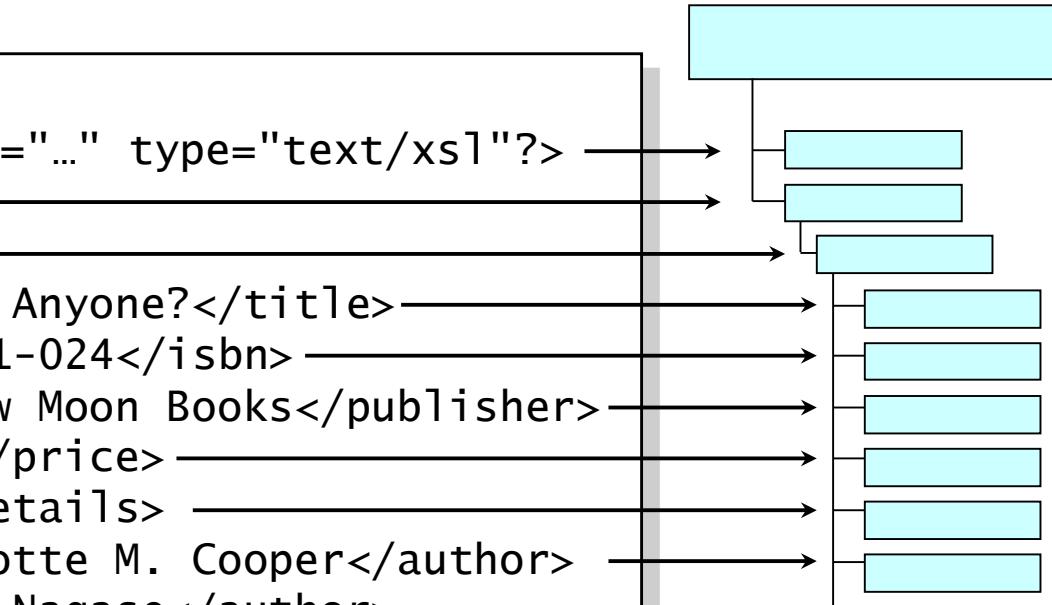
# Обзор XML-технологий

- DOM
  - программное взаимодействие с данными
- XPath
  - описание и выборка элементов
- XSL, XSL-FO, XSLT
  - преобразование XML документов
- XLink, XPointer
  - указатели и ссылки

# DOM и XMLHttpRequest

- `var dom = request.responseXML;`

```
<?xml version="1.0"?>
<?xmlstylesheet href="..." type="text/xsl"?>
<booklist>
 <book>
 <title>Sushi, Anyone?</title>
 <isbn>72-80081-024</isbn>
 <publisher>New Moon Books</publisher>
 <price>14.99</price>
 <details>...</details>
 <author>Charlotte M. Cooper</author>
 <author>Yoshi Nagase</author>
 </book>
 ...
</booklist>
```



# Преобразование DOM к строке

- Internet Explorer

- `var xml_str = dom.xml;`

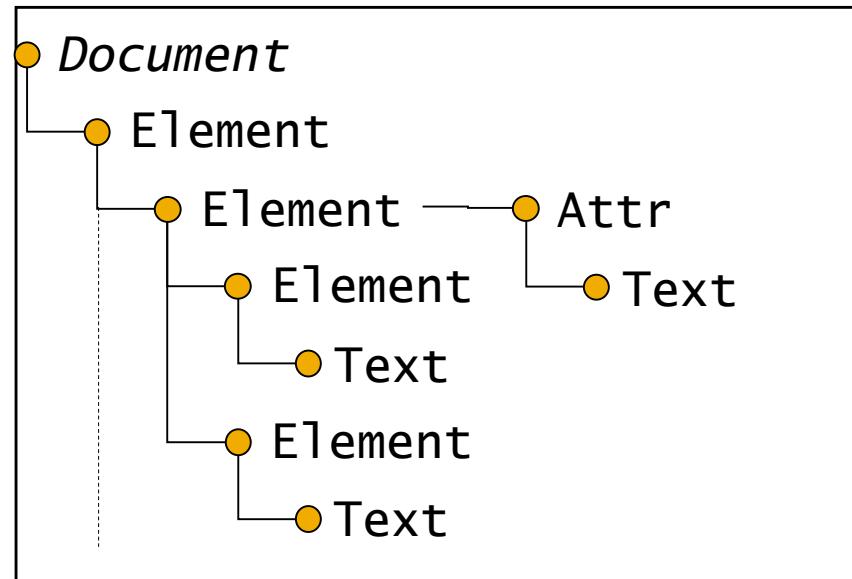
- Другие

- `var s = new XMLSerializer();`

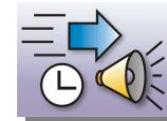
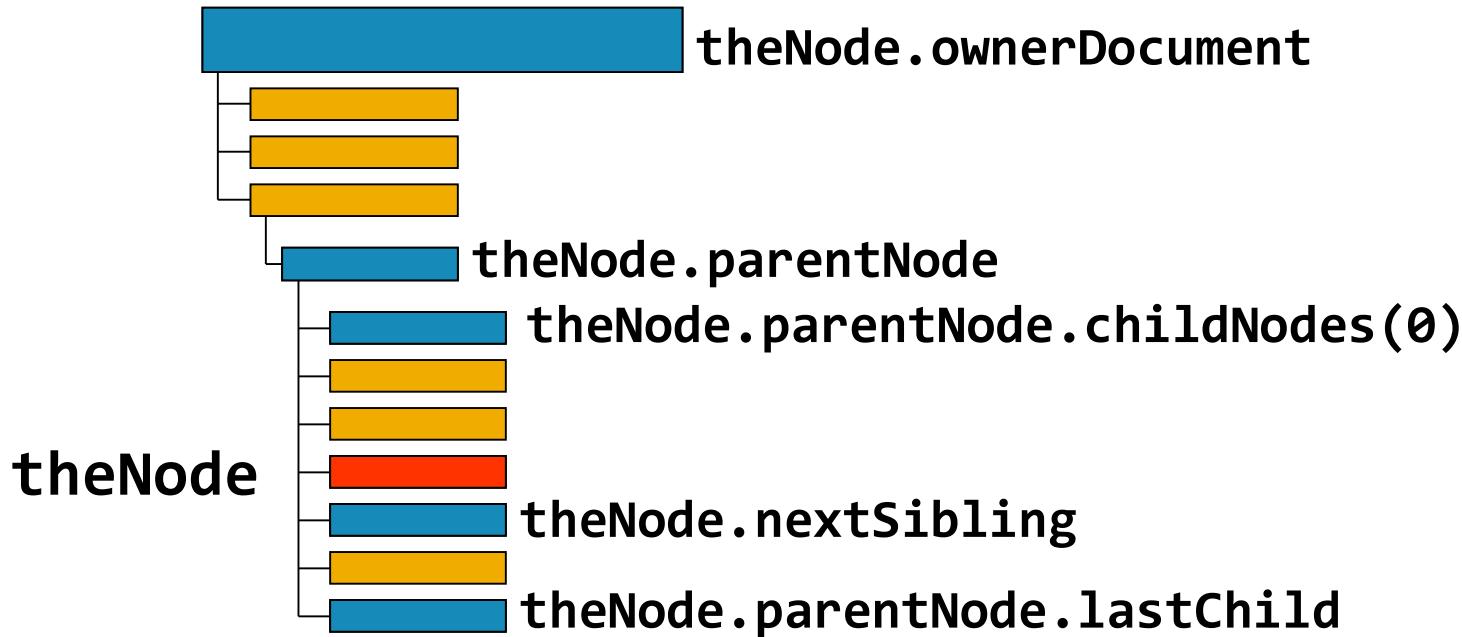
- `var xml_str = s.serializeToString(dom);`

# DOM структура

- Парсер представляет документ как иреархию объектов
- Объекты DOM – это узлы (node) связанные друг с другом
  - Объект Document – основной объект документа
  - Другие объекты представляют элементы, текст, атрибуты, комментарии и т.д.



# Навигация в DOM



# Основные объекты DOM

- IXMLDOMNode
- IXMLDOMElement
- IXMLDOMAttribute
- IXMLDOMText
- IXMLDOMDocument
- другие

# IXMLDOMNode

- Свойства
  - nodeName
  - nodeType
  - nodeValue
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling
  - parentNode
- Методы
  - hasChildNodes
  - appendChild
  - insertBefore
  - replaceChild
  - removeChild
  - cloneNode

# IXMLDOMElement

- Свойства
  - IXMLDOMNode +
  - tagName
- Методы
  - IXMLDOMNode +
  - getAttribute
  - setAttribute
  - getAttributeNode
  - setAttributeNode
  - removeAttribute
  - removeAttributeNode
  - getElementsByTagName

# IXMLDOMDocument

- Свойства
  - IXMLDOMNode +
  - documentElement
- Методы
  - IXMLDOMNode +
  - createElement
  - createTextNode
  - createAttribute
  - createComment
  - createCDATASection
  - getElementsByTagName
  - ...

# Доступ к отдельному элементу

- Корневой элемент
  - `var root = xmlDOM.documentElement;`
- Первый элемент в коллекции
  - `var book = root.childNodes[0];`
- Дочерний элемент
  - `var title = book.childNodes[0];`
- Текстовый узел элемента
  - `alert(title.firstChild.nodeValue);`

# Выборка однотипных элементов

- Выборка всех книг

- `var books = xmlDOM.getElementsByTagName("book");`

- Проход по книгам

- `for (var i = 0; i < books.length; i++){  
 var book = books[i];  
 // Проход по дочерним узлам книги  
 for (var j = 0; j < book.childNodes.length; j++){  
 var node = book.childNodes[j];  
 // Если это не элемент...  
 if (node.nodeType != 1) continue;  
 // Если это title  
 if (node.nodeName == "title")  
 result += node.firstChild.nodeValue + "\n";  
 }  
}`

# Подходы к разработке ПО

- Клиент-ориентированная архитектура
- Сервис-ориентированная архитектура
  - SOA, service-oriented architecture
- Веб-сервисы

# Протокол XML-RPC

- RPC
  - Remote Procedure Call
  - Вызов удалённых процедур
- XML-RPC (
  - Текстовый протокол на базе HTTP
  - RFC-3529

```
<?xml version="1.0"?>
<methodCall>
 <methodName>examples.getStateName</methodName>
 <params>
 <param>
 <value><i4>41</i4></value>
 </param>
 </params>
</methodCall>
```

# Простые типы данных XML-RPC

- Boolean
  - <boolean>1</boolean>
- Integer и Double
  - <i4>237</i4>
  - <double>-12.53</double>
- String
  - <string>Здравствуй, Мир!</string>
- Date/time
  - <dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>
- Base64
  - <base64>eW91IGNhbid0IHJ1YWQgdGhpcyE=</base64>

# Массивы

- Array

```
■ <array>
 <data>
 <value>
 <i4>1404</i4>
 </value>
 <value>
 <string>Что-нибудь здесь</string>
 </value>
 <value>
 <i4>1</i4>
 </value>
 </data>
</array>
```

# Объекты

- Struct

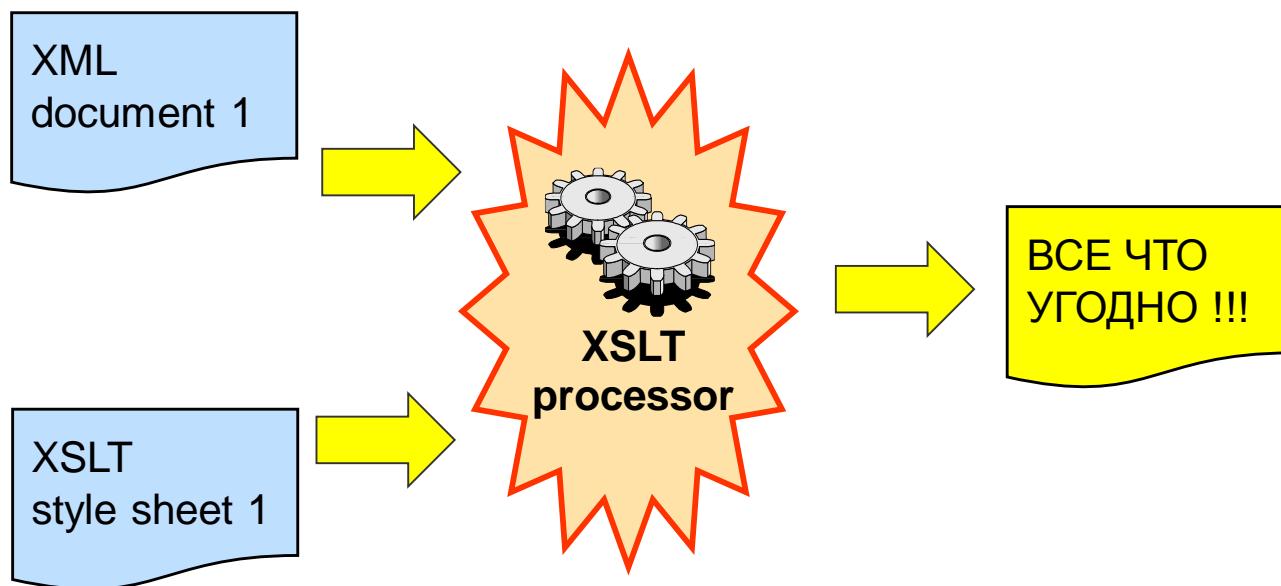
- ```
<struct>
  <member>
    <name>Что-то</name>
    <value><i4>1</i4></value>
  </member>
  <member>
    <name>Ещё что-то</name>
    <value><i4>2</i4></value>
  </member>
</struct>
```

Формирование XML-RPC запроса

- <script src="xmlrpc.js"></script>
- Формируем сообщение
 - var msg = new XMLRPCMessage("myMethod", "utf-8");
msg.addParameter("Строка текста");
msg.addParameter(8);
msg.addParameter(false);
msg.addParameter(a);
msg.addParameter(obj);
msg.addParameter(date);
- Вывод сообщения
 - alert(msg.xml());

Преобразование XML данных

- eXtensible Stylesheet Language
- XSL Transformation



Создание XSL файла

- XSL таблица – XML документ
 - Все элементы принадлежат пространству имен
<http://www.w3.org/1999/XSL/Transform>
 - В XSL таблице могут использоваться любые пространства имен
-
- ```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 ...
 <xsl:template>
 ...
 </xsl:template>
 ...
</xsl:stylesheet>
```

# Шаблоны XSL

- Шаблон (шаблонное правило)
  - правило обработки части XML документа
- ```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="Xpath_выражение">
    тело шаблона
  </xsl:template>

</xsl:stylesheet>
```

XSLT в Internet Explorer

- Загрузка документа

- `var dom = new ActiveXObject("MSXML2.DOMDocument");`

- `dom.async = false;`

- `dom.load("mydata.xml");`

- Загрузка XSLT

- `var xsl = new ActiveXObject("MSXML2.DOMDocument");`

- `xsl.async = false;`

- `xsl.load("my-template.xsl");`

- Преобразование

- `var result = dom.transformNode(xsl);`

XSLT в Fire Fox

- `var xslt = new XSLTProcessor();`
- Загружаем XSL файл с XMLHttpRequest
 - `new, open(), send()`
 - `var xsl = request.responseXML;`
 - `xslt.importStylesheet(xsl);`
- Загружаем XML файл с XMLHttpRequest
 - `new, open(), send()`
 - `var xml = request.responseXML;`
- `var result = xslt.transformToDocument(xml);`

Лабораторная работа 4

Расчет суммы товаров электронного магазина

Упражнение 0: Перед началом работы

- Ознакомьтесь с HTML-кодом файла **labs\lab-4\index.html**
- Обратите внимание на подключение трёх дополнительных файлов: **xslt.js**, **xmlrpc.js** и **xmltools.js**
- Откройте эти файлы и ознакомьтесь с их содержимым

Упражнение 1: Получение способов доставки в электронном магазине

- Откройте в текстовом редакторе файл **labs\lab-4\index.html**
- Найдите в блоке скрипта комментарий: **Задание 1. Получение способов доставки**
- Опишите функцию **getDeliveryMethods()** для получения способов доставки товаров электронного магазина.
 - Обратитесь к серверу XML-RPC **lab-4-server.php** и вызовите метод **eshop.getDeliveryMethods** (без параметров). Сервер вернет XML-RPC ответ (примерное содержание можно посмотреть в файле **messages/getDeliveryMethods.xml**)
 - Создайте переменную **dom** и присвойте ей вернувшийся ответ
 - Для отладки выведите это сообщение на экран с помощью функции **showXML** (она находится в файле **xmltools.js**) , которая просто преобразует XML к строке
- После того, как вы напишите и отладите функцию **getDeliveryMethods()**, поставьте вызов этой функции в событие **window.onload** (просто раскомментируйте нужную строчку)
- Не забудьте про передачу методом **POST** и указание **Content-type: text/xml**

Упражнение 2: Отображение способов доставки на экране

- Откройте файл **labs\lab-4\delivery.xsl** и изучите код преобразования XSLT. Обратите внимание, это преобразование формирует элемент **<select>** со вложенными элементами **<option>**. Закройте этот файл, не изменяя его
- Вернитесь к файлу **labs\lab-4\index.html**
- Найдите в блоке скрипта комментарий: **Задание 2. Отображение способов доставки**
- Опишите функцию **showDelivery()**, которая получает параметр **xmlDOM**
 - Используя функцию преобразования **xsltTransform()** (см. файл **xslt.js**), произведите преобразование с помощью заранее загруженной в переменную **xslDelivery** таблицы преобразования **delivery.xsl**

- Результат преобразования выведите в HTML-элемент **divDelivery**
 (Обратите внимание, в этом объекте уже есть метка <label>, поэтому вывод результата преобразования необходимо сделать так, чтобы строка-результат была бы дописана к существующему HTML-коду)
- Впишите вызов функции **showDelivery()** в тело функции **getDeliveryMethods()**, передав ей в качестве параметра переменную **dom**, предварительно проверив переменную с помощью функции **isError**
- Сохраните файл и проверьте работу скрипта в различных браузерах

Упражнение 3: Получение данных о заказе

- Вернитесь к файлу **labs\lab-4\index.html**
- Найдите в блоке скрипта комментарий: **Задание 3. Получение данных о заказе**
- Опишите функцию **calculateOrder()**, который вернет детализацию расчета общей суммы заказа
 - Сформируйте XML-RPC сообщение и вызовите метод **eshop.calculateOrder**, передавая ему следующие параметры:
 - sum —число, сумма заказа
 - deliveryId — код способа доставки (значение value списка доставки)
 - Параметры **sum** и **deliveryId** вы должны получить из элемента **input txtOrderSum** и сформированным вами списком **selDelivery** (см. упражнение 2)
 - Передайте сформированную XML-RPC строку методом **POST** (не забываем про правильный Content-type!) серверу **lab-4-server.php**
 - Получите и выведите с помощью функции **alert()** результат работы сервера
 - Создайте переменную **dom** и присвойте ей вернувшийся от сервера ответ
 - Для отладки выведите это сообщение на экран с помощью функции **showXML**

Упражнение 4: Отображение данных о заказе

- Вернитесь к файлу **labs\lab-4\index.html**
- Найдите в блоке скрипта комментарий: **Задание 4. Отображение данных о заказе**
- Опишите функцию **showOrder()**, которая получает параметр **xmlDOM** (ответ XML-RPC сервера)
 - Используя заранее загруженный XSL-файл (переменная **xslOrder**), произведите преобразование XML-данных (полученных с сервера) в переменной **xmlDOM**
 - Выведите результат преобразования в HTML-элемент **divOrder**
- Впишите вызов функции **showOrder()** в тело функции **calculateOrder**, передав ей в качестве параметра переменную **dom**, предварительно проверив переменную с

помощью функции **isError**

- Сохраните файл и проверьте работу скрипта в различных браузерах

Выводы

- Простые текстовые форматы имеют серьезные ограничения
- XML – промышленный способ описания и передачи структурированных данных
- Существует множество XML-технологий (XML Schema, XPath, XSLT и др.)
- XML-RPC – простой способ вызова удаленного сервера и передачи ему данных
- XML-RPC позволяет описывать типы

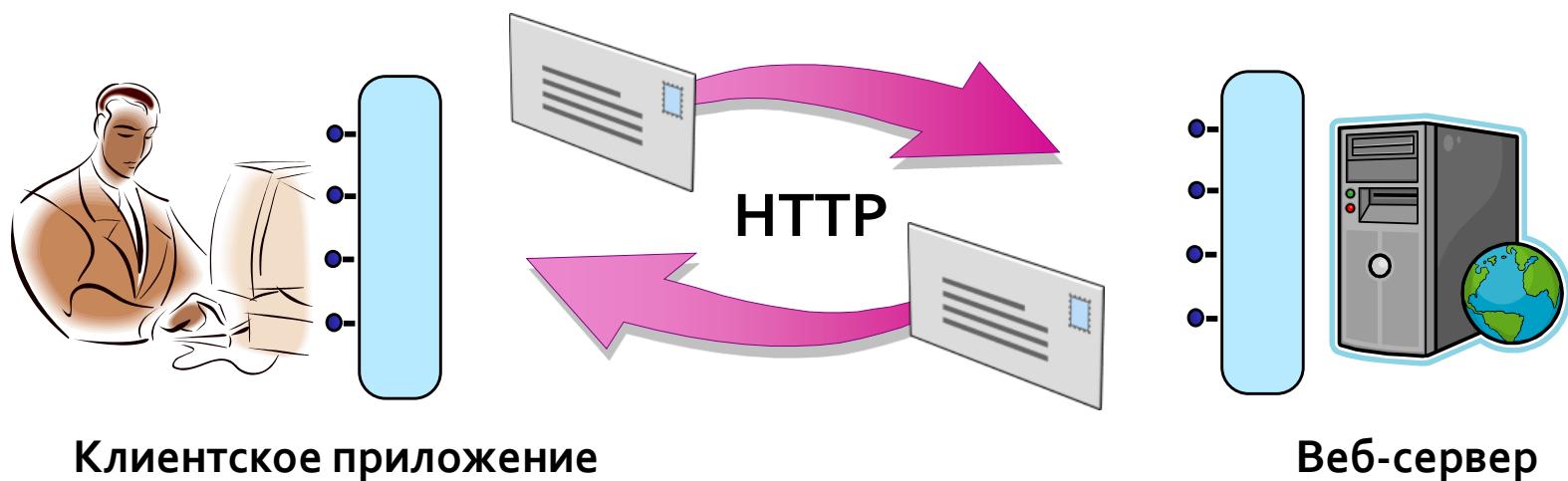
Использование XML веб-сервисов. SOAP

Игорь Борисов
<http://igor-borisov.ru>

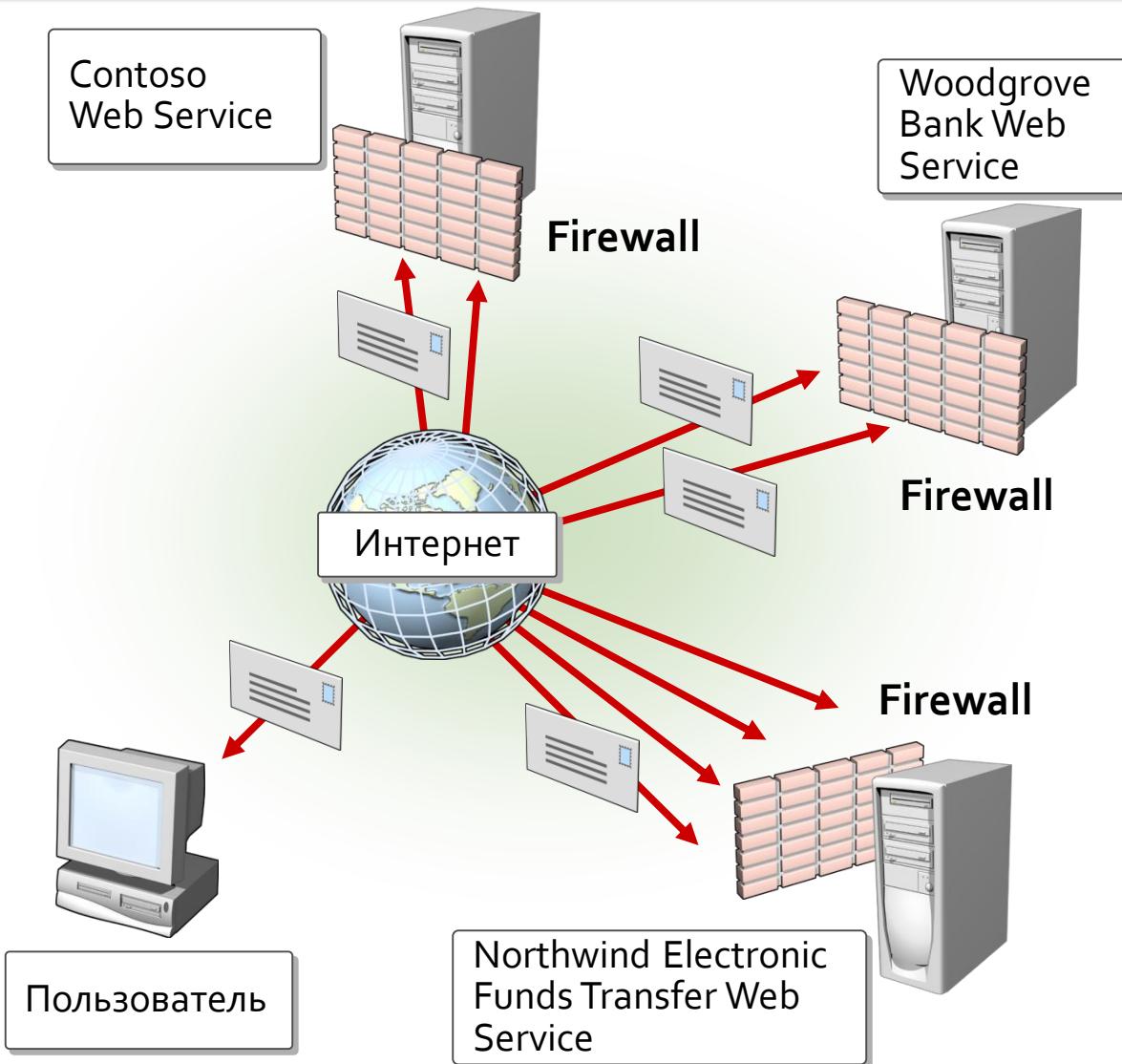
Темы модуля

- XML веб-сервисы
- Проблемы XML-RPC
- XML схемы (обзорно)
- SOAP (обзорно)
- Формирование и разбор SOAP сообщений
- Пример работы с XML веб-сервисами

XML веб-сервисы



Использование веб-сервисов



Проблемы XML-RPC

- Нет возможности:
 - проверить правильность XML-RPC сообщения
 - однозначно заранее описать типы и объекты получаемые/передаваемые сервером
 - описывать и проверять типы с произвольными пространствами имен (например, ссылка на другую спецификацию или сообщения)
 - создавать комбинированные сообщения
 - передачи дополнительной информации в сообщении

XML схемы (обзорно)

- Унифицированный способ описания структуры
- <http://www.w3.org/XML/Schema>
- Промышленный стандарт описания XML документа
- Описывает:
 - **словарь** (названия элементов и атрибутов)
 - **модель содержания** (отношения между элементами и атрибутами и их структура)
 - **типы данных**

Пример простой схемы

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="страна" type="Страна"/>
  <xs:complexType name="Страна">
    <xs:sequence>
      <xs:element name="название" type="xs:string"/>
      <xs:element name="население" type="xs:decimal"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

```
<страна>
  <название>Франция</название>
  <население>59.7</население>
</страна>
```

Типы данных в XML схеме (обзор)

- Простые и сложные типы
- Группы

```
<xsd:element name="type"
type="acctTypeCode"/>
<xsd:simpleType name="acctTypeCode">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[A-Z]{2}" />
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:group name="acct">
  <xsd:sequence>
    <xsd:element name="description"
type="xsd:string"/>
    <xsd:element name="number"
type="xsd:string"/>
    <xsd:element name="type"
type="acctTypeCode"/>
    <xsd:element name="balance"
type="xsd:decimal"/>
  </xsd:sequence>
</xsd:group>
```

```
<account status="active">
  <number>1234-5X</number>
  <type>CK</type>
  <balance>5000.00</balance>
</account>
```

```
<xsd:element name="account" type="acct"/>
<xsd:complexType name="acct">
  <xsd:sequence>
    <xsd:element name="description"
type="xsd:string"/>
    <xsd:element name="number"
type="xsd:string"/>
    <xsd:element name="type"
type="xsd:string"/>
    <xsd:element name="balance"
type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="status" type="xsd:string"/>
</xsd:complexType>
```

```
<xsd:complexType name="checkingAcct">
  <xsd:sequence>
    <xsd:group ref="acct" />
  </xsd:sequence>
  <xsd:attribute name="status" type="xsd:string"/>
</xsd:complexType>
```

Сложные типы (обзорно)

■ Композиторы

- sequence
- choice
- all

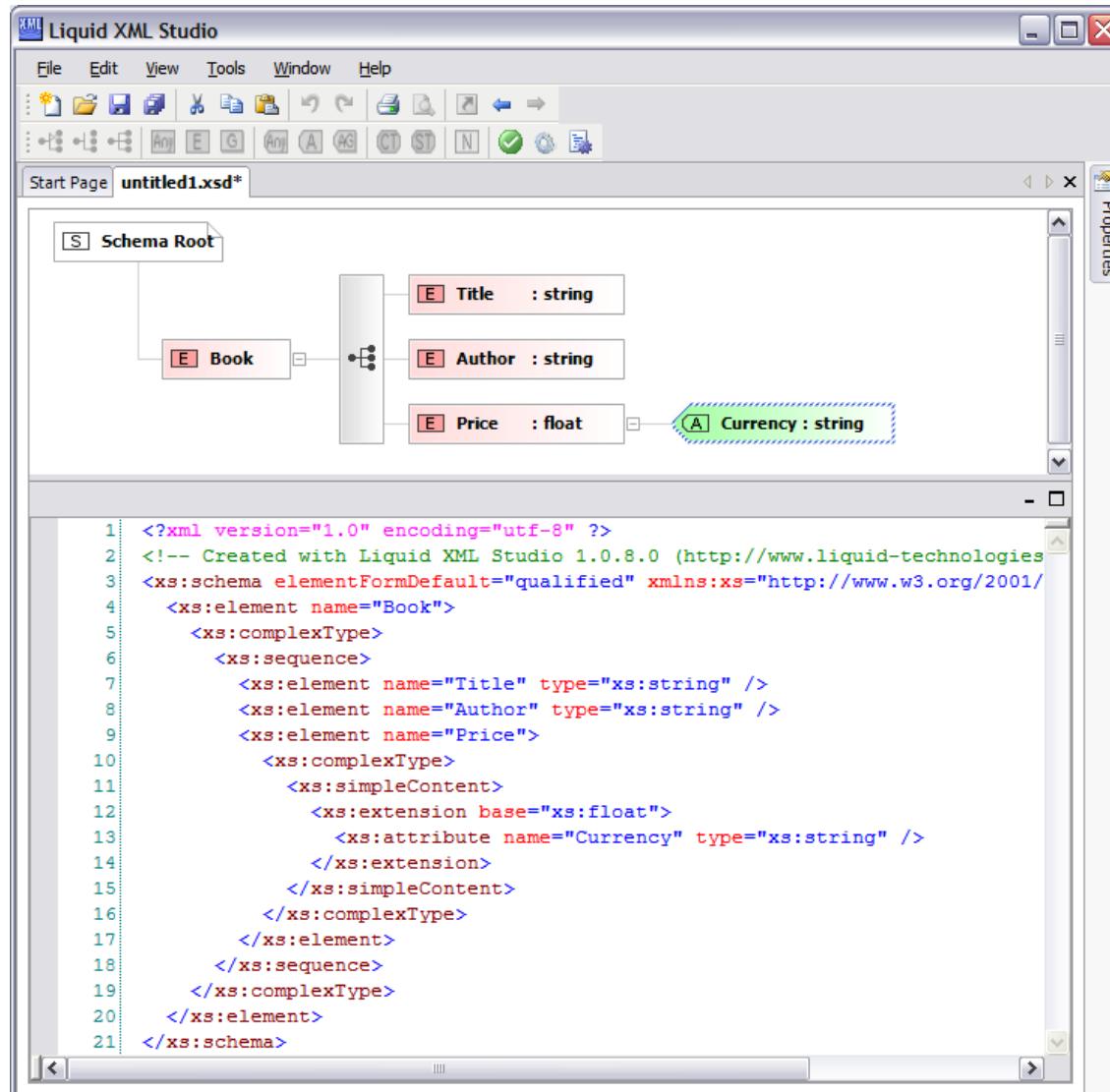
■ Наследование

- restriction
- extension

```
<xsd:complexType name="acct">
  <xsd:sequence>
    <xsd:element name="description"
      type="xsd:string"/>
    <xsd:element name="number" type="xsd:string"/>
    <xsd:element name="type" type="acctTypeCode"/>
    <xsd:element name="balance"
      type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="status" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="savingsAcct">
  <xsd:complexContent>
    <xsd:extension base="acct" >
      <xsd:sequence>
        <xsd:element name="minimumBalance"
          type="xsd:decimal" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Инструменты для XML схем



SOAP

- Simple Object Access Protocol
 - Протокол обмена структурированными сообщениями
- Промышленный стандарт построения распределенных приложений
 - <http://www.w3.org/TR/soap/>
- Может использоваться с любым протоколом прикладного уровня
- Основной протокол реализации XML Web Services (XML Веб-служб)

Основные операции SOAP



Структура SOAP сообщений

- SOAP Envelope
(конверт)
- SOAP Header
(заголовок)
- SOAP Body
(тело)
- SOAP Fault
(ошибка)

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi=...>
    <soap:Header>
        <WoodgroveAuthInfo xmlns="http://tempuri.org/">
            <Username>string</Username>
            <Password>string</Password>
        </WoodgroveAuthInfo>
    </soap:Header>
    <soap:Body>
        <GetAccount xmlns="http://tempuri.org/">
            <acctID>int</acctID>
        </GetAccount>
    </soap:Body>
</soap:Envelope>

</detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

Запрос SOAP 1.1

- POST /DailyInfoWebServ/DailyInfo.asmx HTTP/1.1

Host: www.cbr.ru

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

Content-Length: 90

SOAPAction: http://web.cbr.ru/GetCursOnDate

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <GetCursOnDate xmlns="http://web.cbr.ru/">
            <On_date>dateTime</On_date>
        </GetCursOnDate>
    </soap:Body>
</soap:Envelope>
```

Ответ SOAP 1.1

- HTTP/1.1 200 OK

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

Content-Length: 1024

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
    <GetCursOnDateResponse xmlns="http://web.cbr.ru/">
        <GetCursOnDateResult>
            <xsd:schema>schema</xsd:schema>
            ...xml...
        </GetCursOnDateResult>
    </GetCursOnDateResponse>
</soap:Body>
</soap:Envelope>
```

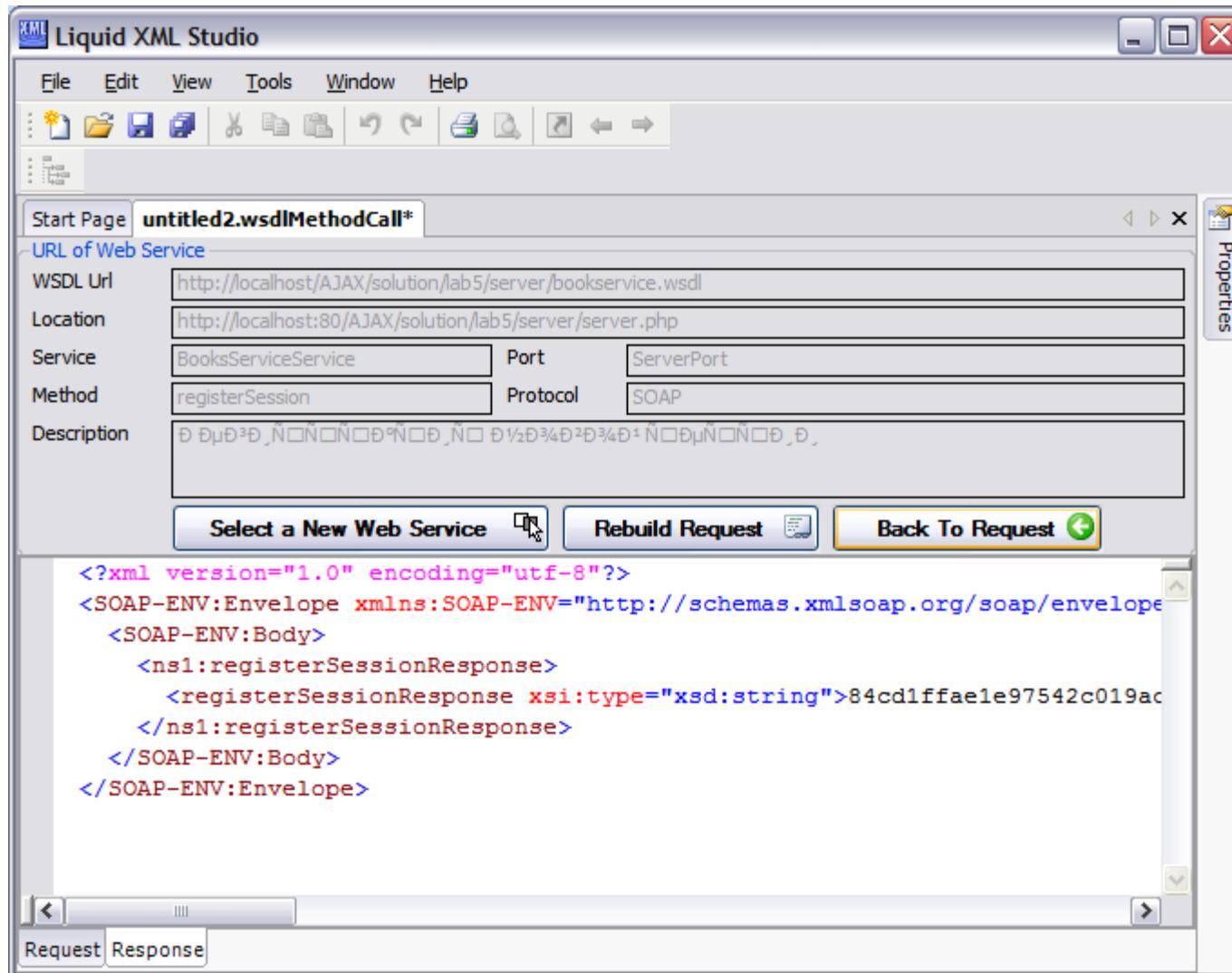
WSDL

- Web Services Description Language
 - Декларация пространств имен
 - Схема данных
 - Типы сообщений
 - Привязка сообщений к методам
 - Декларация методов
 - Описание сервиса и способов взаимодействия

Основные элементы WSDL

- service
 - port
 - address
- binding
 - operation
- portType
 - operation
- message
 - part

Пример работы со службой



Лабораторная работа 5

Сценарий выбора и добавления книг в корзину

Упражнение 0: Перед началом работы

- Ознакомьтесь с HTML-кодом файла `labs\lab-5\index.html`
- Обратите внимание на подключение двух дополнительных файлов: `xslt.js` и `xmltools.js`
- Ознакомьтесь с HTML-кодом файла `labs\lab-5\xmltools.js`

Упражнение 1: Регистрация новой сессии пользователя

- Откройте в текстовом редакторе файл `labs\lab-5\index.html`
- Найдите в блоке скрипта комментарий: **Задание 1. Регистрация новой сессии**
- Опишите функцию `registerSession()`, которая регистрирует новую сессию пользователя, вызывая SOAP-метод `registerSession`
URI этого метода — `urn:SampleServer2-registerSession`
 - Используя функцию `loadXML()` загрузите сообщение `server/messages/registerSession-request.xml`
 - Выполните асинхронный запрос к серверу (`server/server.php`) SOAP-метода `registerSession` (см. URI) с помощью функции `sendMessage()`
(пример такого вызова можно посмотреть в уже написанной функции `showBooks()`)
- Опишите функцию `registerSessionCallback()`, которая будет использоваться в асинхронном вызове сервера
 - Получите SOAP-сообщение ответа сервера (`req.responseXML`) и прочитайте в нем элемент `registerSessionResponse`
(см. файл `registerSession-response.xml`)
 - В этом элементе находится ID новой сессии пользователя. Сохраните его в глобальную переменную `sessionId`
 - Для отладки выведите ID новой сессии пользователя в HTML-элемент `sessionId`
- Поставьте вызов функции `registerSession()` в событие `window.onload` и проверьте работу в разных браузерах

Упражнение 2: Добавление книг в корзину

- Вернитесь к файлу `labs\lab-5\index.html`
- Найдите в блоке скрипта комментарий: **Задание 2. Добавление книг в корзину**
- Опишите код функции `addToBasket()`, которая добавляет книгу в корзину, вызывая SOAP-метод `addToBasket`.
URI этого метода — `urn:SampleServer2-addToBasket`.

Функция должна получить параметр **xmlDOM**, и, вызывая функцию преобразования **xsltTransform()** (см. файл xslt.js), произвести преобразование с помощью заранее загруженной в переменную **xslDelivery** таблицы преобразования **delivery.xsl**

- Загрузите сообщение **addToBasket-request.xml**, используя функцию **loadXML()**
- В этом сообщении не забудьте установить параметры **sessionId** и **bookId**, где **sessionId** — сохраненный в переменной идентификатор сессии пользователя, а **bookId** — идентификатор книги, получаемый функцией, как параметр.
Для установки параметров используйте функцию **setParameter**
(Пример можно посмотреть в уже написанной функции **showBooks()**)
- Выполните асинхронный вызов сервера (**server/server.php**), передавая построенное SOAP сообщение
- Напишите функцию **addToBasketCallback()**, которая будет использоваться в асинхронном вызове сервера
 - В этой функции сохраните ответ сервера (**req.responseXML**) в локальной переменной
 - Найдите и проверьте элемент **addToBasketResult**. Если он равен **true** — книга добавлена в корзину
 - Выведите пользователю сообщение об этом
(На следующем этапе лабораторной работы вы отобразите содержание корзины)
- Проверьте работу вашего сценария. Для этого просто в браузере посмотрите книги в любой категории и щелкните по любой книге

Упражнение 3: Отображение корзины на экране

- Вернитесь к файлу **labs\lab-5\index.html**
- Найдите в блоке скрипта комментарий: Задание 3. Отображение корзины
- Опишите код функции **showBasket()**, которая добавляет книгу в корзину, вызывая SOAP-метод **getBasket**
URI этого метода — **urn:SampleServer2-getBasket**
 - Загрузите сообщение **getBasket-request.xml**, используя функцию **loadXML**
 - В этом сообщении не забудьте установить параметр **sessionId**. Для установки параметров используйте функцию **setParameter**
(Пример можно посмотреть в уже написанной функции **showBooks()**)
 - Выполните асинхронный вызов сервера (**server/server.php**), передавая построенное SOAP сообщение
- Напишите функцию **showBasketCallback()**, которая будет использоваться в асинхронном вызове сервера
- В этой функции сохраните ответ сервера (**req.responseXML**) в локальной переменной

- Загрузите таблицу преобразования **server/xslt/getBasket.xsl** в локальную переменную с помощью функции **loadXML()**
- Выполните XSLT-преобразование с помощью функции **xsltTransform()**, используя данные SOAP сообщения и загруженную таблицу преобразования
- Результат преобразования выведите в HTML-элемент **basketPlaceholder**
- Проверьте работу вашего сценария. Для этого просто в браузере посмотрите книги в любой категории и щелкните по любой книге

Упражнение 4: Очистка корзины

- Вернитесь к файлу **labs\lab-5\index.html**
- Найдите в блоке скрипта комментарий: Задание 4. Очистка корзины
- Опишите код функции **emptyBasket()**, которая удаляет книги из корзины, вызывая SOAP-метод **emptyBasket**.
URI этого метода — **urn:SampleServer2-emptyBasket**
 - Загрузите сообщение **emptyBasket-request.xml**, используя функцию **loadXML()**
 - В этом сообщении не забудьте установить параметр **sessionId**. Для установки параметров используйте функцию **setParameter**
 - Выполните асинхронный вызов сервера (**server/server.php**), передавая построенное SOAP сообщение
- Опишите функцию **emptyBasketCallback()**, которая будет использоваться в асинхронном вызове сервера
 - Сохраните ответ сервера (**req.responseXML**) в локальной переменной
 - В этом результате (см. сообщение **emptyBasket-response.xml**) прочитайте элемент **emptyBasketResult**. Если он равен **true** — корзина очищена
 - Выведите пользователю сообщение об этом, и очистите HTML-элемент **basketPlaceholder**
- Проверьте работу вашего сценария. Для этого просто в браузере посмотрите книги в любой категории и щелкните по любой книге несколько раз. Очистите корзину нажатием на кнопку [Очистить]

Выводы

- XML-RPC имеет определенные недостатки
- SOAP — более совершенная версия XML-RPC
- На базе SOAP строятся XML веб-сервисы
- XML веб-сервисы — промышленный способ организации и построения распределенных приложений
- XML-схемы используются для описания данных
- WSDL — описание XML Web-сервисов
- SOAP клиент может быть любым

Безопасность и эффективность АJАХ приложений

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Вопросы безопасности AJAX приложений
- Аутентификация и авторизация пользователя
- Проблемы юзабилити AJAX приложений
- Производительность AJAX приложений
- Обзор решений AJAX
- Подведение итогов

Безопасность АЈАХ приложений

- Аутентификация пользователя
- Авторизация пользователя
- Защита трафика

Аутентификация и авторизация

- Аутентификация
 - Authentication
 - подтверждение подлинности субъекта
- Авторизация
 - Authorization
 - подтверждение прав субъекта на доступ к защищаемым объектам.
 - обычно проходит после аутентификации

Аутентификация средствами HTTP

- Basic аутентификация (RFC 2617)
- Digest аутентификация (RFC 2617)
- HTTPS (RFC 2618)
- Kerberos (RFC 4120)
- X.509 (RFC 5280)
- OpenID (<http://openid.net/>)
- OAuth (<http://oauth.net/>)
- Windows Live ID (<http://msdn.microsoft.com/en-us/library/bb404787.aspx>)

Базовая аутентификация

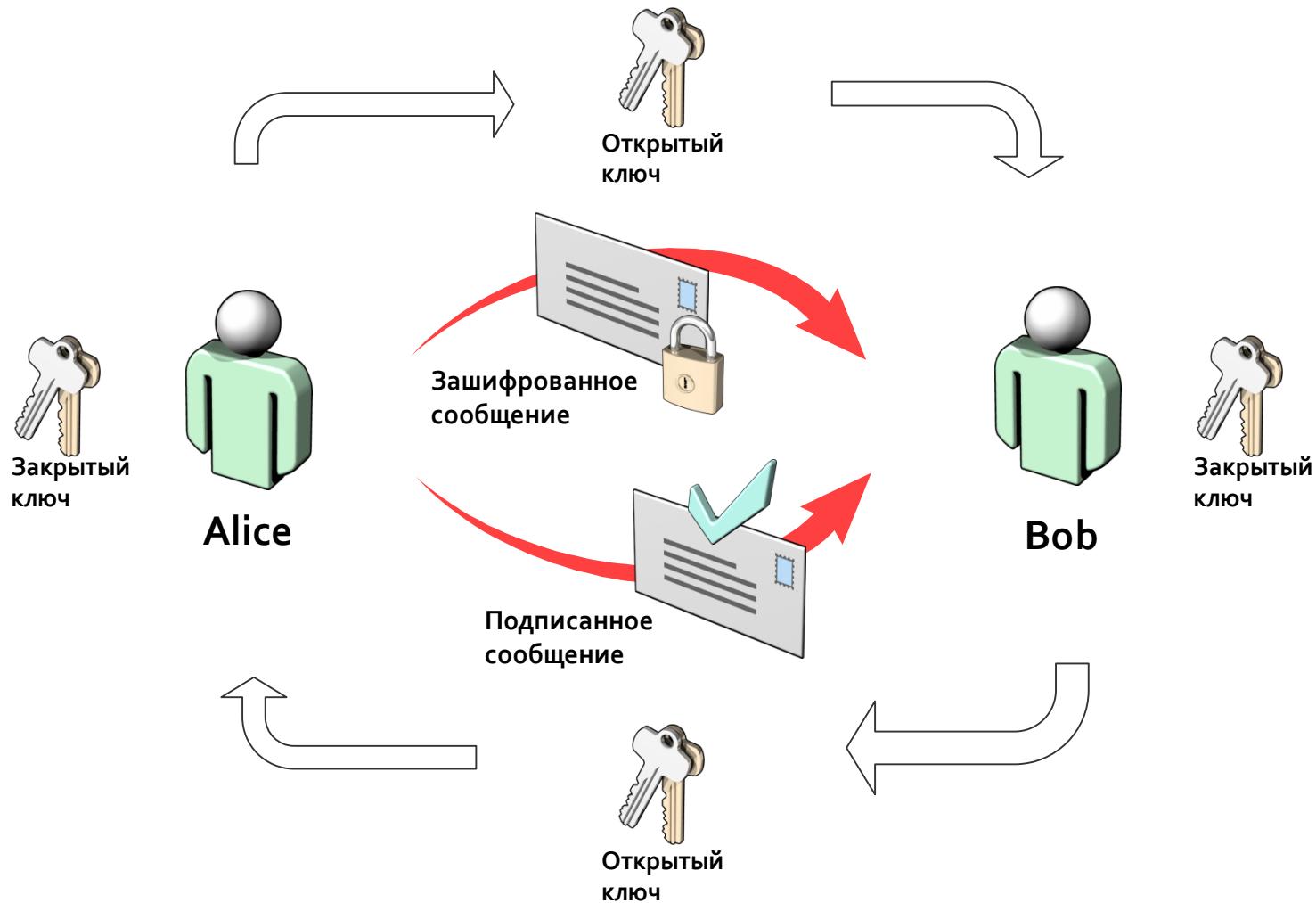
- GET /AJAX/demo/module6/base_auth/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*
- HTTP/1.x **401** Authorization Required
WWW-Authenticate: Basic realm="Private zone"
Content-Type: text/html
- GET /AJAX/demo/module6/base_auth/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*
Authorization: Basic dmFzeWE6cGFzc3dvcmQ=

vasya:password

Хеширование

- MD5 (RFC 1321)
- SHA1 (RFC 3174)
- "Hello, world"
 - bc6e6f16b8a077ef5fbc8d59dob931b9
- "password"
 - 5f4dcc3b5aa765d61d8327deb882cf99
- "!!!"
 - d41d8cd98foob204e9800998ecf8427e

Несимметричная криптография



Защита на прикладном уровне

- Правило № 1:
 - Никогда, ни при каких обстоятельствах, не следует «изобретать» или использовать свои собственные (кустарные, «секретные») алгоритмы шифрования и аутентификации!
- Безопасное хранение и безопасная передача пароля (RFC 2898)
 - хэш(...(salt + хэш(salt + password))

Лабораторная работа 6.1

Аутентификация пользователя

Упражнение 1: Запрос «соли» хеширования с сервера

- Откройте в текстовом редакторе файл **labs\lab-6-1\index.html**
- Найдите в блоке скрипта комментарий: **Задание 1. Запрос «соли» хеширования с сервера**
- Допишите функцию **loginUser()**
 - Создайте новый объект **User** и установите у него свойство **email** из переменной **email**
 - Передайте асинхронным вызовом этот объект на сервер, сериализовав его в JSON-строку
(адрес сервера — глобальная переменная **server**)
 - Получите данные от сервера и десериализуйте его в объект — это объект **User**
 - Вызовите функцию **sendPassword()**, передав ей полученный объект от сервера
- В целях отладки выведите с помощью **alert()** ответ сервера
- Запишите, какие свойства у полученного объекта установлены
- Обратите внимание на значения объектов энтропии
- Для проверки работы используйте следующие данные:
 - e-mail пользователей: vasyap@mail.ru, fedyas@mail.ru, masha@mail.ru
 - у всех пользователей пароль: password

Упражнение 2: Хеширование пароля

- Вернитесь к файлу **labs\lab-6-1\index.html**
- Найдите в блоке скрипта комментарий: **Задание 2. Хеширование пароля**
- Допишите код функции **sendPassword()**
 - Проведите хеширование введенного пароля (свойство **user.password**) с помощью функции **getSaltedHash()**
 - Передайте этой функции «соль» **user.dbEntropy.salt** и число итераций **user.dbEntropy.iterationCount**, и сохраните результат в свойство **user.password**
 - Проведите второе хеширование, передавая «соль» **user.transferEntropy.salt** и число итераций **user.transferEntropy.iterationCount**, и сохраните результат в свойство **user.password**

- Сериализуйте объект user в JSON-строку и асинхронно передайте его на сервер (адрес сервера — глобальная переменная server)
 - Получите данные от сервера и десериализуйте JSON строку в объект
 - Вызовите функцию **showUserData()**, передав полученный объект в качестве параметра
- В целях отладки выведите с помощью alert() ответ сервера
 - Запишите, какие свойства у полученного объекта установлены
 - Обратите внимание на значения свойства **user.name**

Упражнение 3: Вывод данных

- Вернитесь к файлу **labs\lab-6-1\index.html**
- Найдите в блоке скрипта комментарий: **Задание 3. Вывод данных**
- В зависимости от значения свойства **user.name** покажите пользователю результат проверки пароля:
 - Если это свойство пусто — пользователь ошибся
 - Если нет — то в этом свойстве записано правильное имя пользователя
- Покажите сообщение пользователю в HTML-элементе **divResult**
- Проверьте работу скрипта в различных браузерах

Юзабилити AJAX приложений

- Проблемы:
 - Состояние приложения никак не соответствует URL
 - Как правило, нет возможности отмены действия или возврата на шаг назад
 - Очень сложно сохранить или передать состояние приложения
 - При отключенном JavaScript AJAX не работает
 - Серьезные проблемы при использовании мобильных браузеров

Производительность AJAX приложений

- AJAX приложения, как правило, увеличивают нагрузку на сервер
- Необходимо тщательно планировать структуру AJAX приложения
- Если контент можно загрузить статично — загружайте его статично!
- Избегайте AJAX загрузки изображений
- Обязательно: обратная связь с пользователем!
 - AJAX приложение должно сообщать пользователю о процессе своей работы

Реализация обратной связи

- Страйтесь всегда показывать пользователю состояние приложения
- Реализуйте поясняющие сообщения «Идет загрузка данных» «Обработка...» и т.п.
- Управляйте доступностью элементов HTML (например, устанавливайте `disabled` для кнопок) при длительных операциях
- Реализуйте возможность отмены длительной операции
- Реализуйте возможность получения минимума данных при отключенном JavaScript

Лабораторная работа 6.2

Загрузка больших объемов информации (если останется время)

Упражнение 1: Загрузка главы книги

- Откройте в текстовом редакторе файл **labs\lab-6-2\index.html**
- Найдите в блоке скрипта комментарий: **Задание 1. Загрузка главы книги**
- Допишите функцию **getChapter()**
 - Покажите сообщение пользователю о загрузке данных. Для этого установите у HTML-элемента **divMessageLoad** значение свойства **display** в **block**
 - Сформируйте и выполните асинхронный GET запрос к серверу (глобальная переменная **serverXml**), передавая ему параметр **no** с номером текущей главы (аргумент функции)
 - Получите XML данные и вызовите функцию **showChapter()**, передавая ей параметры: полученный DOM документ и номер текущей главы

Упражнение 2: Показ главы книги

- Вернитесь к файлу **labs\lab-6-2\index.html**
- Найдите в блоке скрипта комментарий: **Задание 2. Показ главы книги**
- Допишите функцию **showChapter()**
 - На основании переменной **currentChapter** сформируйте ссылки «Вперед» и «Назад», загружающие следующую и предыдущую главу книги
 - Для загрузки используйте функцию **getChapter()**
 - Выведите сформированные ссылки в HTML-элемент **divChapters**
 - Произведите преобразование полученной главы (переменная **xmlDOM**) с помощью загруженного XSL файла (глобальная переменная **fb2html**)
 - Преобразование выполните с помощью функции **xsltTransform()**
 - Результат преобразования выведите в HTML-объект **divResult**
 - Погасите сообщение пользователю о загрузке данных, устанавливая значение свойства **display** в **none** для HTML-элемента **divMessageLoad**
- Проверьте работу скрипта в различных браузерах

Обзор решений AJAX

- ASP.NET AJAX
- AJAX.OOP
- xajax
- JsHttpRequest

- jQuery
- Extjs
- Dojo
- MooTools
- Prototype

Выводы

- Безопасность приложения – важная часть разработки AJAX приложений
 - *Secure functions ≠ Secure Application*
- Необходимо защищать приложение на всех этапах разработки
- Аутентификация и авторизация пользователя – важная часть защиты
- Юзабилити и доступность AJAX решений – первостепенная задача разработчика
- Существует множество готовых решений AJAX