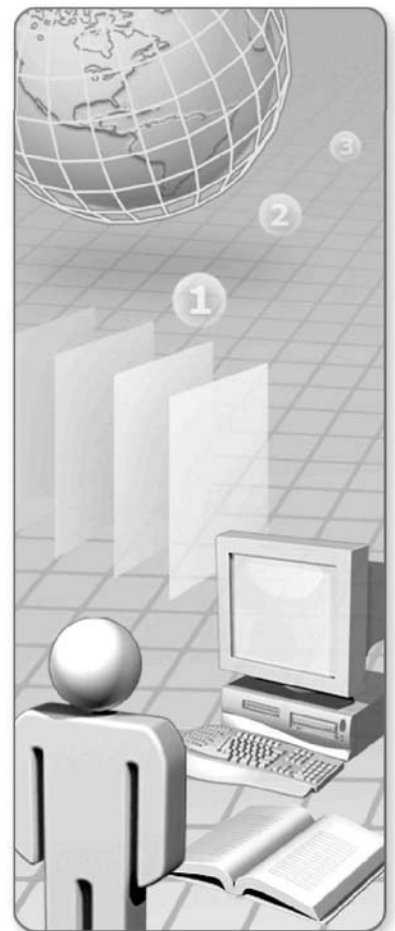
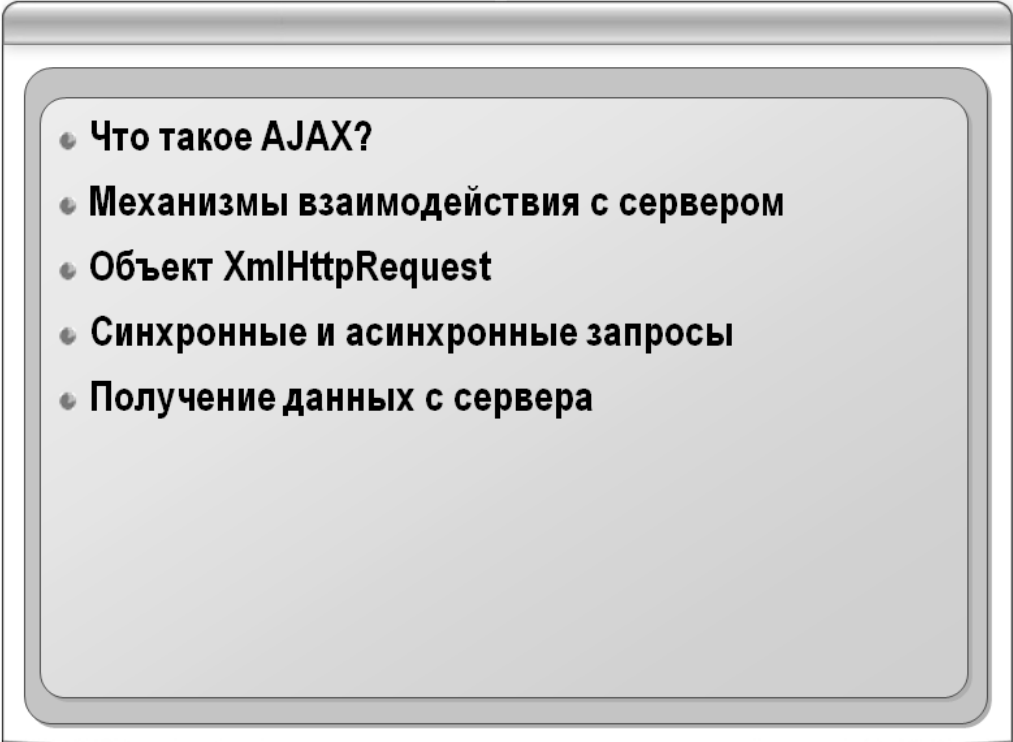


## Модуль 1

# Основы построений AJAX приложений



## Основы построений AJAX приложений

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

## Что такое AJAX?

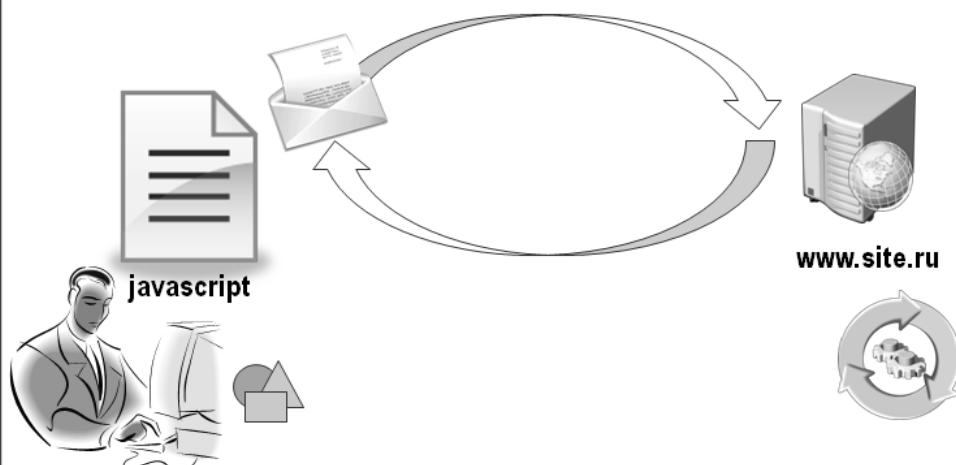
- **Asynchronous JavaScript and XML**  
*«асинхронный JavaScript и XML»*
- **Подход к построению интерактивных Веб-приложений**
- **Обмен данными с сервером без перегрузки страницы в браузере**

## История и AJAX сегодня

- **1998 г. Microsoft Remote Scripting**
- **1999 г. Новая реализация MSXML и объект XMLHttpRequest**
- **2000 г. Outlook Web Access**
- **2005 г. Термин AJAX**  
(статья Джесси Джеймса Гарретта - <http://www.adaptivepath.com/ideas/essays/archives/000385.php>)
- **2005 г. Активное использование технологии компанией Google (Gmail, Google Maps)**
- **2005 г. Термин Web 2.0**  
(статья Тима О'Рейли <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>)

## Механизмы взаимодействия с сервером

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



## Объект XMLHttpRequest (XMLHTTP)

- |                      |                           |
|----------------------|---------------------------|
| • onreadystatechange | • abort()                 |
| • readyState         | • getAllResponseHeaders() |
| • responseText       | • getResponseHeader()     |
| • responseXML        | • open()                  |
| • status             | • send()                  |
| • statusText         | • setRequestHeader()      |

## Создание объекта XMLHttpRequest

- Ранние версии Internet Explorer

```
var req = new  
  ActiveXObject("Microsoft.XMLHTTP");
```

- Internet Explorer 6 и ниже

```
var req = new  
  ActiveXObject("Msxml2.XMLHTTP");
```

- Mozilla Firefox, Opera, IE7

```
var req = new XMLHttpRequest();
```

## Пример универсальной функции

```
function getXmlHttpRequest() {  
  if (window.XMLHttpRequest) {  
    try {  
      return new XMLHttpRequest();  
    } catch (e){} }  
  else if (window.ActiveXObject) {  
    try {  
      return new ActiveXObject('Msxml2.XMLHTTP');  
    } catch (e){}  
    try {  
      return new ActiveXObject('Microsoft.XMLHTTP');  
    }  
    catch (e){} }  
  return null;  
}
```

## Синхронные и асинхронные запросы

- Синхронный запрос ожидает завершения обращения к серверу
- Операции после асинхронного запроса выполняются параллельно с завершением запроса
- Для контроля состояния используется свойство `readyState` и событие `onreadystatechange`

## Выполнение синхронного запроса

```
// Объект XMLHttpRequest
var request = getXmlHttpRequest();
// Запрос на сервер
request.open("GET", url, false);
request.send(null);
```

## Выполнение асинхронного запроса

```
// Объект XMLHttpRequest
var request = getXmlHttpRequest();
// Установка обработчика
request.onreadystatechange = function()
{
    // только при состоянии "complete"
    if (req.readyState == 4)
        ...
}
// Запрос на сервер
request.open("GET", url, false);
request.send(null);
```

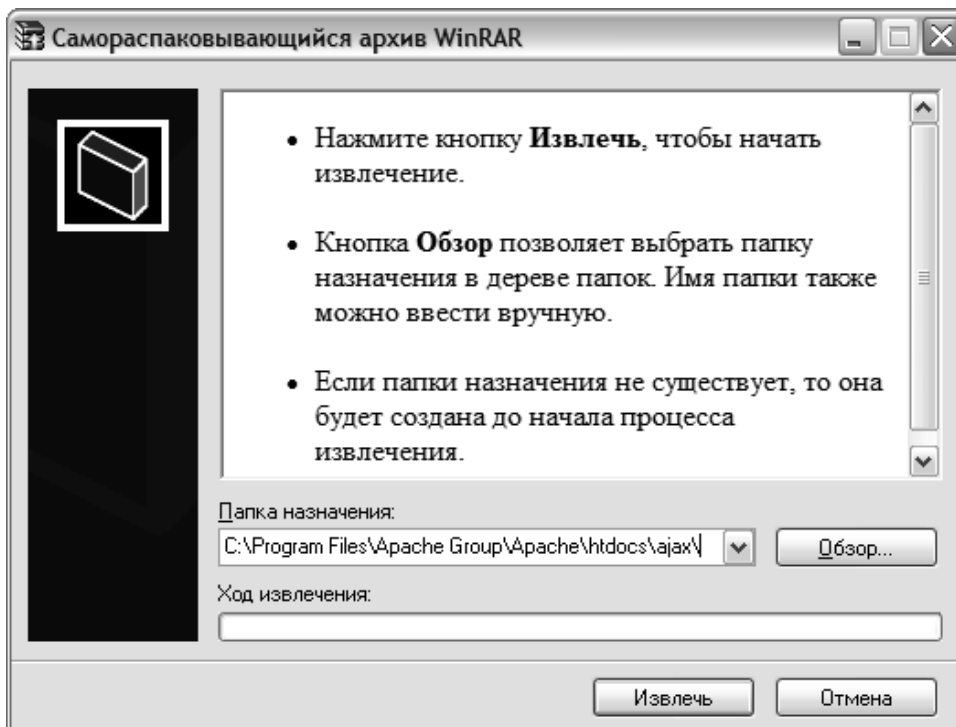
## Получение данных с сервера

- Свойство **responseText** – текст, полученный с сервера
- Свойство **responseXML** – XML DOM, полученный с сервера
- Свойство **status** – HTTP статус ответа сервера
- Метод **getResponseHeader()** – заголовки ответа, переданные сервером

## Лабораторная работа

### Упражнение 0: Подготовка к работе

- ♦ Найдите на диске или получите у тренера файл labsetup.exe и выполните его. Этот файл представляет собой самораскрывающийся архив с файлами лабораторных работ.
- ♦ Запустите этот файл на выполнение
- ♦ Убедитесь, что путь распаковки указывает на папку  
**C:\Program Files\Apache Group\Apache\htdocs\ajax\**



- ♦ Нажмите кнопку **[Извлечь]** и дождитесь завершения операции
- ♦ Откройте с помощью проводника папку  
**C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\lab1**
- ♦ Щелкните по кнопке **[Start]** и далее по ярлыку **Run**
- ♦ Наберите в окне запуска команду **cmd** и нажмите кнопку **[Ok]**
- ♦ В командном окне наберите команду:  
**net start apache**
- ♦ Откройте браузер и наберите в адресной строке адрес  
**http://localhost/ajax/labs/lab1**
- ♦ Убедитесь в работоспособности сервера Apache. Если что-то не получается — спросите у преподавателя



## Упражнение 1: Создание объекта XMLHttpRequest

- ♦ Откройте проводник Windows
- ♦ В папке **lab1**<sup>1</sup> найдите файл **index.html**
- ♦ Щелкните по нему правой кнопкой мышки и выберите пункт «**Edit with Notepad++**»
- ♦ Найдите в блоке скрипта комментарий

```
/*
** Задание 1: Напишите функцию создания объекта XMLHttpRequest
**     function getXmlHttpRequest()
** */
```

- ♦ После этого комментария напишите функцию **getXmlHttpRequest()**, которая вернет объект XMLHttpRequest, работая в любом браузере
- ♦ Сохраните файл и убедитесь в его работоспособности, 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://localhost/ajax/...», а не просто кликая по файлу!**

---

<sup>1</sup> Здесь и далее указываются относительные пути. Все лабораторные работы будут выполняться в папке:  
**C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

### Упражнение 2: Асинхронный запрос к серверу

- ♦ Вернитесь к файлу **index.html** в папке **lab1**
- ♦ Найдите в блоке скрипта комментарий

```
/*  
** Задание 2: Напишите Функцию (функции) запроса к серверу...  
*/
```

- ♦ После этого комментария напишите функцию **getBookByNumber(bookNum)**, которая **АСИНХРОННО** запрашивает нужную книгу по номеру и выводит строку с названием этой книги в элемент **divResult** HTML страницы
- ♦ Эта функция должна сформировать запрос к серверу по адресу **http://localhost/ajax/labs/lab1/getbooktxt.php** и передать этой PHP-странице GET параметр **num** со значением номера книги, например так:

`http://localhost/AJAX/labs/lab1/getbooktxt.php?num=1`

- ♦ Полученный от сервера результат выведите в элемент **<div id="divResult">&nbsp;  </div>** HTML страницы
- ♦ Сохраните файл и проверьте его работу, обращаясь к файлу лабораторной работы по адресу:

`http://localhost/AJAX/labs/lab1/index.html`

### Упражнение 3: Подключение AJAX-сценария к элементам страницы

- ♦ Вернитесь к файлу **index.html** в папке **lab1**
- ♦ Найдите в блоке скрипта комментарий

```
/*
** Задание 3: Напишите код обработчика нажатия на кнопку ...
*/
```

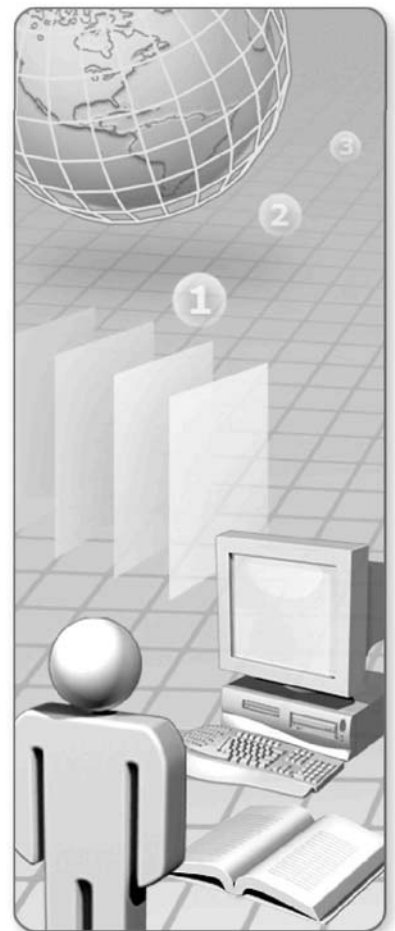
- ♦ После этого комментария напишите функцию **showBook()**, которая обращается к элементу HTML `<input id="txtNum" type="text" />`, считывает введенный номер и вызывает функцию **getBookByNumber(bookNum)**, чтобы показать информацию о книге
- ♦ Проверьте работоспособность вашей страницы в нескольких браузерах

## Выводы

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

## Модуль 2

# Взаимодействие с сервером, передача данных



## **Взаимодействие с сервером, передача данных**

- **Методы передачи данных на сервер**
- **Передача простых данных методом GET**
- **Управление кэшированием ответа**
- **Запросы HEAD**
- **Передача простых данных методом POST**
- **Получение и разбор комплексных данных с сервера**

## **Методы передачи данных на сервер**

- **Взаимодействие с сервером**
- **Методы протокола HTTP (RFC2616)**
- **Статус сервера**
- **Заголовки запроса и ответа**

## Запрос HTTP - терминология

```
GET /AJAX/labs/lab1/index.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2008 14:54:47 GMT
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Last-Modified: Sun, 15 Jun 2008 14:00:06 GMT
Content-Length: 1801
Content-Type: text/html

<html><html>
```

## Заголовки запроса

```
GET /AJAX/labs/lab1/index.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
5.1; ru; rv:1.9) Gecko/2008052906 Firefox/3.0
Accept:
text/html,application/xhtml+xml,application/xml;
q=0.9,*/*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cache-Control: max-age=0
```

## Заголовки ответа

```
HTTP/1.x 200 OK
Date: Sun, 15 Jun 2008 14:54:47 GMT
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Cache-Control: no-store, no-cache, max-age=0
Expires: Sun, 15 Jun 2008 14:54:47 GMT
Last-Modified: Sun, 15 Jun 2008 14:00:06 GMT
Etag: "0-709-48552066"
Accept-Ranges: bytes
Content-Length: 1801
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

## Статус сервера

- **2xx – Успешно**
- **301, 302 – Переадресация**
- **304 – Объект не изменялся**
- **401 – Требуется авторизация**
- **403 – Запрещено**
- **404 – Объект не найден**
- **405 – Метод не поддерживается**
- **500 – Ошибка сервера**



## Передача простых данных методом GET

```
<form action="server.php" method="get">
  <input type="text" name="t1" />
  <input type="text" name="t2" />
  <input type="text" name="t3" />
</form>
```

```
GET server.php?t1=vasya&t2=pupkin&t3=megaUser HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*
```

## Управление кэшированием ответа

```
HTTP/1.x 200 OK
Date: Sun, 15 Jun 2008 14:54:47 GMT
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Cache-Control: no-store, no-cache, max-age=0
Expires: Sun, 15 Jun 2008 14:54:47 GMT
Last-Modified: Sun, 15 Jun 2008 14:00:06 GMT
Content-Length: 1801
Content-Type: text/html
```

---

```
# Заголовок Cache-Control
<IfModule mod_headers.c>
  Header append Cache-Control "no-store, no-cache"
</IfModule>

# Заголовок Expires
<IfModule mod_expires.c>
  ExpiresActive On
  ExpiresDefault "now"
</IfModule>
```

## Запрос HEAD

```
HEAD /AJAX/labs/lab1/index.html HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*

HTTP/1.1 200 OK
Date: Sun, 15 Jun 2008 14:54:47 GMT
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Last-Modified: Sun, 15 Jun 2008 14:00:06 GMT
Content-Length: 1801
Content-Type: text/html
```

## Лабораторная работа 2.1

### Упражнение 1: Запрос методом GET

- ♦ С помощью проводника Windows откройте папку **lab2.1**<sup>1</sup>
- ♦ Щелкните правой кнопкой мышки по файлу **index.html** и выберите пункт «**Edit with Notepad++**»
- ♦ Найдите комментарий:

```
/*  
** Задание 1 ...  
*/
```

- ♦ Напишите функцию **fillCategories()**, которая асинхронно получает данные из сценария **getcategories.php**
- ♦ Этот сценарий возвращает список строк, разделенных символом конца строки, содержащих код и название категории
- ♦ Формат строк: «код:название», например «1:Web»
- ♦ Сформируйте из этих строк элементы **options** и добавьте их в HTML элемент **<select id="selCategory"></select>**
- ♦ Поставьте вызов этой функции в событие **window.onload**, чтобы при загрузке страницы выпадающий список заполнился данными:

```
// При завершении загрузки страницы  
window.onload = function()  
{  
    fillCategories();  
}
```

- ♦ Проверьте работу страницы в нескольких браузерах

---

<sup>1</sup> Напоминаем, папка лабораторных работ находится здесь:  
**C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

## Недостатки методов GET и HEAD

- Ограниченная 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 простых данных методом POST

```
POST /server.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost/AJAX/form.html
Content-Length: 30
Pragma: no-cache
Cache-Control: no-cache

cat=2&user=Vasya+Pupkin&age=30

HTTP/1.x 200 OK
Date: Sun, 15 Jun 2008 15:30:34 GMT
Server: Apache/1.3.39 (Win32) PHP/5.2.6
Cache-Control: no-store, no-cache
Expires: Sun, 15 Jun 2008 19:30:34 +0400
Content-Type: text/plain; charset=utf-8
```

## Получение и разбор комплексных данных с сервера

- **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

### Упражнение 1: Запрос методом POST

- ♦ С помощью проводника Windows откройте папку **lab2.2**
- ♦ Щелкните правой кнопкой мышки по файлу **index.html** и выберите пункт «**Edit with Notepad++**»
- ♦ Найдите комментарий:

```
/*
** Задание 2 ...
*/
```
- ♦ Напишите функцию **showBooks()**, которая по выбранной 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 **<table id="tableBooks"></table>**
- ♦ Поставьте вызов функции **showBooks()** в событие **onclick** кнопки **<button onclick="">Показать</button>**
- ♦ Проверьте работу вашего скрипта в различных браузерах

## Упражнение 2: Запрос методом HEAD

- ♦ Вернитесь к файлу **index.html**
- ♦ 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  
**C:\Program Files\Apache Group\Apache\htdocs\ajax\images**
- ♦ Проверку наличия файла необходимо выполнять с помощью метода HEAD
- ♦ Если остается время, допишите функцию проверки наличия файла так, чтобы в атрибут **alt** изображения вписывалась бы информация о размере файла изображения, которую вы сможете прочитать из заголовка **Content-Length** при выполнении HEAD запроса
- ♦ Проверьте работу вашего скрипта в различных браузерах

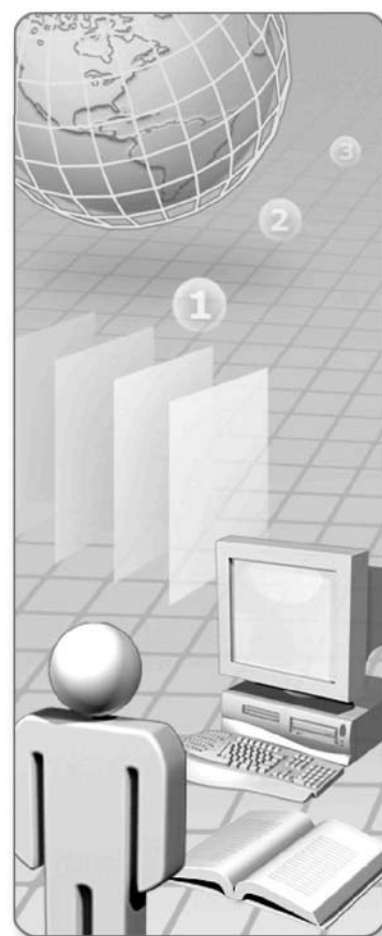
## Выводы

- 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



## Модуль 3

### Передача сложных типов данных. Нотация JSON



## Передача сложных типов данных. Нотация JSON

- Недостатки простых текстовых форматов
- Способы передачи структурированных данных
- Нотация JSON
- Разбор JSON пакета в браузере
- Сериализация и разбор JSON пакета на сервере (PHP)
- Получение данных с сервера

## Недостатки простых текстовых форматов

- Сложность представления данных, особенно в большом количестве
- Сложность отладки и ненаглядность представления
- Сложность обработки большого числа данных
- Отсутствие контроля целостности
- Отсутствие типов данных

## Способы передачи структурированных данных

- JSON 😐
- XML 😊
- Собственные форматы 😞

## Нотация JSON

- JavaScript Object Notation
- Текстовый формат обмена данными, основанный на JavaScript
- Строится на двух структурах:

Набор пар имя/значение. В различных языках это реализовано как **объект**, список с ключом или ассоциативный массив.

Пронумерованный набор значений. Во многих языках это реализовано как **массив**, список или последовательность.

## Пример JSON строки

```
{
  "firstName": "Василий",
  "lastName": "Пупкин",
  "address":
  {
    "streetAddress": "Бакунинская, 71",
    "city": "Москва",
    "postalCode": 105005
  },
  "phoneNumbers":
  [
    "+7 (495) 780-48-48",
    "+7 (495) 775-31-94"
  ]
}
```

## Элементы JSON нотации

- **Объект**
- **Массив**
- **Значение**
- **Строка**

## Объект

- **Объект** — это неупорядоченное множество пар имя/значение, заключенное в фигурные скобки `{}`. Между именем и значением стоит символ `:`, а пары имя/значение разделяются запятыми

```
{  
    "title"    : "Книга",  
    "author"   : "Автор",  
    "price"    : 150  
}
```

## Массив (одномерный)

- **Массив (одномерный)** — это множество значений, имеющих порядковые номера (индексы). Массив заключается в квадратные скобки `[]`. Значения отделяются запятыми.

```
[  
    "Понедельник",  
    "Вторник",  
    false,  
    280  
]
```

## Значение

- Значение может быть *строкой* в двойных кавычках, или *числом*, или *true*, или *false*, или *null*, или *объектом*, или *массивом*. Эти структуры могут быть вложены друг в друга.

## Строка

- Строка — это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки, с использованием *escape-последовательностей* начинающихся с обратной косой черты (*backslash*). Символы представляются простой строкой.

## Сериализация и десериализация

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

## Простая десериализация JSON

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

## Опасность простой сериализации

```
var jsonString = '{"title" : "Книга", "author" : "Автор", "price" : 150}'; alert("Вас хакнули!"); //';

var book = eval("(" + jsonString + ")");

alert(book.title);
```

## Разбор JSON пакета в браузере

```
<script type="text/javascript" src="json2.js"></script>

var myObject =
    JSON.parse(myJSONtext[, filter]);

var myJSONText =
    JSON.stringify(myObject);
```

<http://www.JSON.org/js.html>



## Лабораторная работа 3.1

### Упражнение 1: Форма авторизации пользователя

- ♦ Откройте проводник Windows и найдите папку lab3<sup>1</sup>
- ♦ Найдите файл **index.html**, щелкните по нему правой кнопкой мышки и выберите «**Edit with Notepad++**»
- ♦ Изучите HTML код этой страницы
- ♦ Обратите внимание на форму

```
<form id="frmLogin" onsubmit="return false" class="block">
```

Эта форма не показывается, так как стилевой разметкой для нее установлено свойство `visibility: hidden`
- ♦ Напишите обработчик кнопки **[Вход]** (найдите эту кнопку в HTML коде), который отображает форму входа на экране, устанавливая для этого объекта значение свойства `visibility: visible`
- ♦ Для этого напишите функцию `showLoginForm()` после комментария в коде

```
/*  
** Задание 1. Напишите сценарий отображения формы frmLogin  
** при нажатии на кнопку "Вход" в блоке divUsers.  
*/
```

- ♦ Добейтесь правильного отображения формы при нажатии на кнопку **[Вход]**

---

<sup>1</sup> Лабораторные работы находятся в папке **C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

### Упражнение 2: Проверка пользователя

- ♦ Обратите внимание на объявленные функции-конструкторы классов **UserInfo()** и **Ticket()**  
Для проверки пользователя вы должны сформировать объект **UserInfo**, заполнив его свойства **login** и **password** и передать этот объект в виде JSON строки на сервер сценарию **user\_auth.php**. Этот сценарий проверит пользователя по базе данных и вернет вам объект **Ticket** (билет пользователя), в котором находится информация **id** – идентификатор и **valid** – свойство, равное **true**, если пользователь правильно ввел данные. Полученный билет ваш скрипт должен сохранить в переменной и в последствии предъявлять его серверу для запроса других данных. Сервер по идентификатору билета сможет узнать и проверить, что ваш пользователь прошел проверку и авторизацию.

- ♦ Найдите в коде следующий комментарий

```
/*  
** Задание 2. Напишите функцию проверки пользователя ...  
*/
```

- ♦ После этого комментария напишите функцию **validateUser()** и поставьте ее вызов на
- ♦ В этой функции считайте введенные пользователем
- ♦ Создайте объект **UserInfo** и заполните у него свойства **login** и **password**
- ♦ Произведите JSON сериализацию этого объекта и передайте методом POST полученную строку серверу **user\_auth.php**. Не забудьте указать **Content-type: text/plain** и **Content-length**, равный длине строки
- ♦ В случае успеха сервер вернет вам строку с данными. Десериализуйте эту строку в объект и прочитайте у него свойство **valid**. Если это свойство **true** – пользователь правильно ввел логин и пароль.
- ♦ Если свойство **valid = false**, то пользователь ошибся. Покажите пользователю сообщение об ошибке. Для этого установите свойство **visibility: visible** для объекта HTML  
`<div id="divMessage" class="block">`
- ♦ Не забудьте написать код закрытия окна с ошибкой с помощью кнопки **[Заккрыть]**
- ♦ Сохраните полученный объект (это объект **Ticket**) в переменную **var ticket**
- ♦ Проверьте работу скрипта. Для входа используйте следующие логины пользователей: **vasyap**, **fedias**, **vovam**, **galp**, **svetao**  
У всех этих пользователей в качестве пароля выбрано слово **password**
- ♦ Проверьте работу скрипта в разных браузерах

## Сериализация и разбор JSON пакета на сервере (PHP)

```
<?php
$val = array("abc" => 12,
             "foo" => "bar",
             "bool0" => false,
             "bool1" => true,
             "arr" => array(1, 2, 3, null, 5),
             "float" => 1.2345
            );
$output = json_encode($val);

echo $output."\n";
?>
```

## Десериализация JSON на сервере (PHP)

```
<?php
$input = '{ "abc": 12, "foo": "bar",
           "bool0": false, "bool1": true, "arr":
           [ 1, 2, 3, null, 5 ], "float":
           1.234500 }';
$val = json_decode($input);

print_r($val);
?>
```

## JSON и .Net Framework

```
Product product = new Product();
product.Name = "Apple";
product.Expiry = new DateTime(2008, 12, 28);
product.Price = 3.99M;
product.Sizes = new string[]
    { "Small", "Medium", "Large" };

string json = JavaScriptConvert.SerializeObject(product);

Product deserializedProduct =
    JavaScriptConvert.DeserializeObject<Product>(json);
```

<http://www.codeplex.com/json>

## Лабораторная работа 3.2

### Упражнение 4: Список пользователей он-лайн

- ♦ Переключитесь на редактор Notepad++ с файлом index.html
- ♦ Если вы его закрыли, то откройте проводник Windows и найдите папку lab3<sup>2</sup>
- ♦ Найдите файл **index.html**, щелкните по нему правой кнопкой мышки и выберите «**Edit with Notepad++**»
- ♦ Найдите в коде следующий комментарий

```
/*  
** Задание 4. Напишите функцию запроса пользователей,  
** которые находятся в режиме online  
*/
```

- ♦ Напишите функцию **showOnLineUsers()**
- ♦ В этой функции получите ссылку на объект **UL**, который находится в объекте `<div id="divUsers" class="block">`
- ♦ Сериализуйте в строку JSON полученный ранее билет **ticket**
- ♦ В асинхронном режиме запросите сервер (**get\_online\_users.php**), передавая ему как обычный текст (Content-type: text/plain) 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**
- ♦ Проверьте работу скрипта в различных браузерах

### Домашнее задание

#### Упражнение 5: Обмен сообщениями между пользователями

Самостоятельно дома напишите сценарий PHP и необходимые AJAX функции для обмена сообщениями между пользователями, находящимся в online.

---

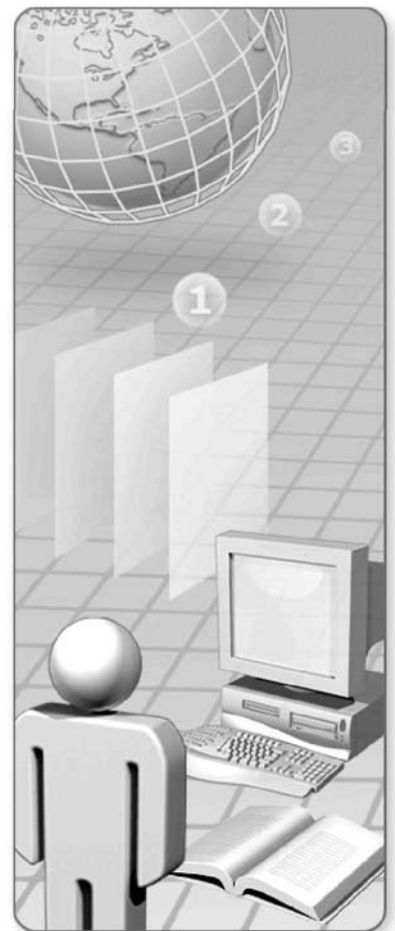
<sup>2</sup> Лабораторные работы находятся в папке **C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

## Выводы

- **JSON – это 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**
- **PHP поддерживает JSON сериализацию встроенными средствами**
- **ASP.Net... А .Net Framework может всё! 😊**

## Модуль 4

### Использование XML XML-RPC



## Использование XML. XML-RPC

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



## Проблемы текстовых данных и JSON пакетов

- Отсутствие типов данных или их ограниченный набор
- Сложность контроля целостности данных
- Сложность визуализации данных
- Сложность преобразования данных

### Отсутствие типов данных

- **Что это?**

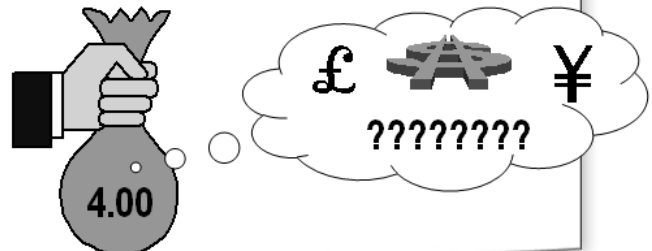
date: "02/07/2007"

quo: 12

price: "Hello, world!"

id: 12-1

size: 56



## Сложность контроля целостности данных

- Как это проверить?

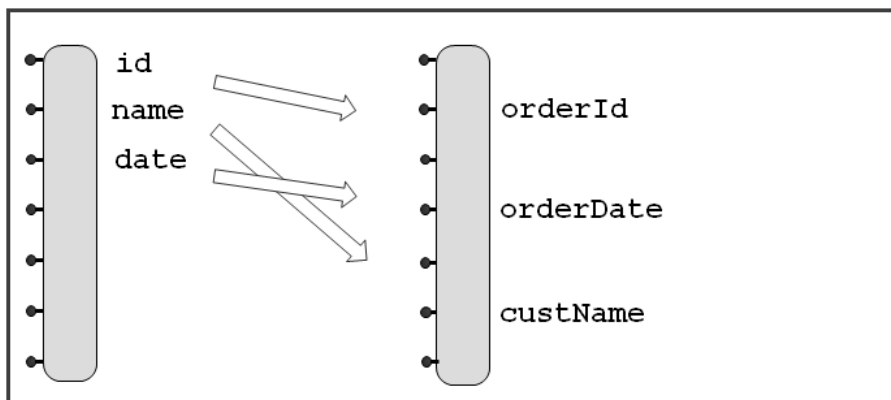
title: "Книга 1"

price: "qwe-12"

autor: "123"

## Сложность визуализации и преобразования данных

- Большие объекты, сериализованные JSON, сложно отобразить, например, в виде HTML
- Еще сложнее преобразовать из одного формата в другой, например перенести часть свойств объекта элемента каталога в заказ



## Другие способы передачи структурированных данных

- XML — *eXtensible Markup Language* — расширяемый язык разметки
- XML — 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>
```

**XML**

## Обзор XML технологий

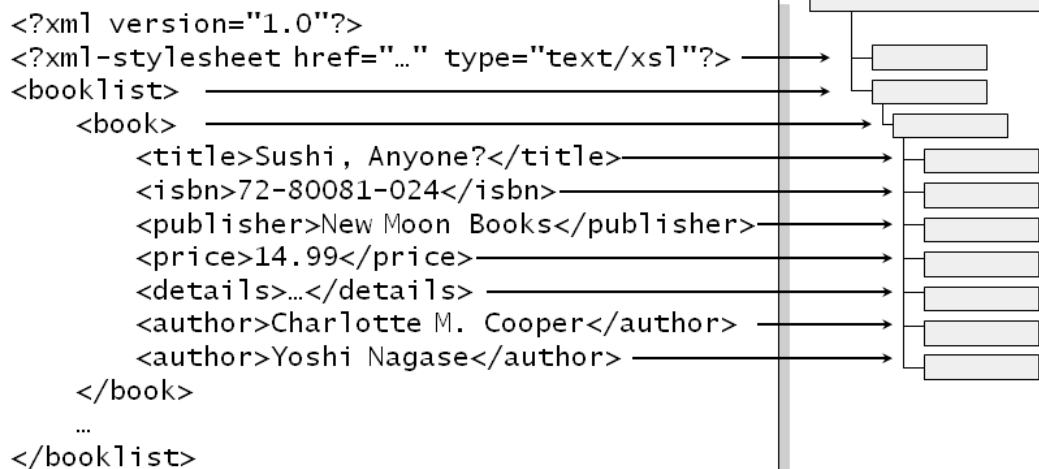
- DOM — программное взаимодействие с данными
- XLink, XPointer — указатели и ссылки
- XPath — описание и выборка элементов
- XSL, XSL-FO, XSLT — преобразование XML документов

## Разбор XML пакета

```
// Объект XMLHttpRequest
var request = getXmlHttpRequest();
// Установка обработчика
request.onreadystatechange = function()
{
    // только при состоянии "complete"
    if (req.readyState == 4)
        var xml = req.responseXML;
    ...
}
```

## DOM представление объектов

- Любой элемент документа рассматривается парсером как объект в памяти

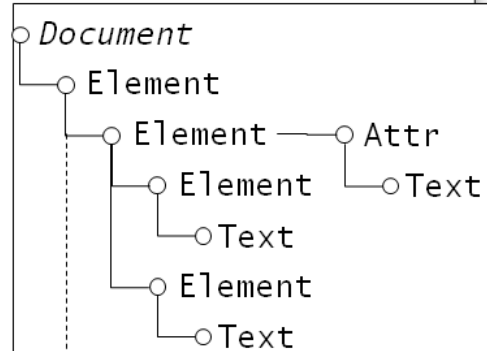


## DOM структура

- **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.



## Преобразование XML документа к строке

- **Internet Explorer**

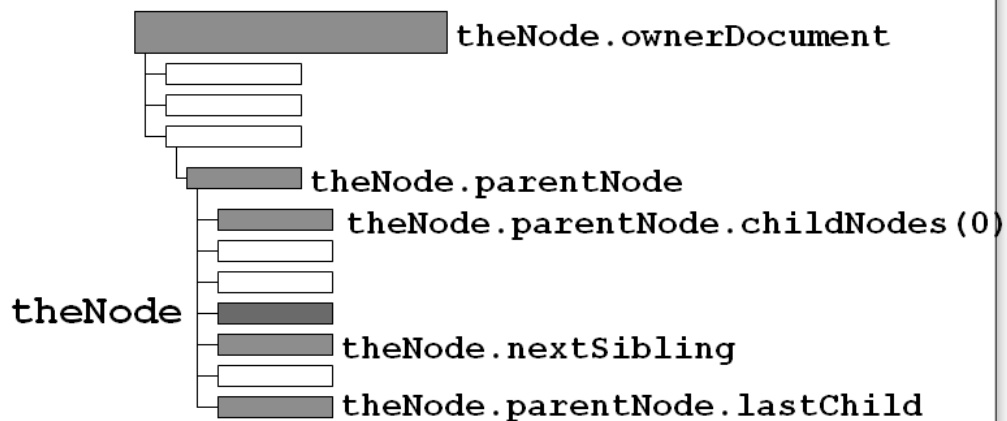
```
var str = dom.xml;
```

- **FireFox**

```
var serializer = new XMLSerializer();
var str = serializer.serializeToString(dom);
```

## Навигация в DOM структуре

- DOM объекты имеют свойства для навигации в дереве



## Основные объекты DOM

- `IXMLDOMNode`
- `IXMLDOMElement`
- `IXMLDOMAttribute`
- `IXMLDOMText`
- `IXMLDOMDocument / DOMDocument`
- другие

## IXMLDOMNode – Node

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

## IXMLDOMElement – Element

- attributes
- childNodes
- firstChild
- lastChild
- nextSibling
- nodeName
- nodeType
- nodeValue
- ownerDocument
- parentNode
- previousSibling
- **tagName**
- appendChild
- cloneNode
- **getAttribute**
- **getAttributeNode**
- **getElementsByTagName**
- hasChildNodes
- insertBefore
- **normalize**
- **removeAttribute**
- **removeAttributeNode**
- removeChild
- replaceChild
- **setAttribute**
- **setAttributeNode**

## IXMLDOMDocument - Document

### IXMLDOMNode+

- documentElement
- ownerDocument
- parentNode
- ...

### IXMLDOMNode+

- createAttribute
- createCDATASection
- createComment
- createDocumentFragment
- createElement
- createEntityReference
- createProcessingInstruction
- createTextNode
- 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";
    }
}
```

## Клиент-ориентированная и сервер-ориентированная архитектура

- **Сервис-ориентированная архитектура (англ. SOA, service-oriented architecture) — подход к разработке программного обеспечения, основанный на использовании сервисов (служб) со стандартизированными интерфейсами**
- **Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные, заменяемые сервисы-приложения**
- **Программные комплексы, разработанные в соответствии с SOA, часто реализуются как набор веб-сервисов, интегрированных при помощи известных стандартных протоколов (SOAP, WSDL, и т. п.)**

## Протокол XML-RPC

- **RPC (от англ. Remote Procedure Call) — технология, позволяющая компьютерным программам вызывать функции или процедуры на удалённых компьютерах**
- **XML-RPC — стандарт/протокол вызова удалённых процедур, основанный на XML**

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

## Типы данных XML-RPC

- **Boolean**
- **Integer**
- **Double**
- **String**
- **Date/time**
- **Base64**
- **Array**
- **Struct**
- **Nil**

## Простые типы данных

- Boolean
 

```
<boolean>1</boolean>
```
- Integer и Double
 

```
<double>-12.53</double>
```
- String
 

```
<string>Здравствуй, Мир!</string>
```
- Date/time
 

```
<dateTime.iso8601>19980717T14:08:55</dateTime.iso8601>
```
- Base64
 

```
<base64>eW91IGNhbid0IHJlYWQgdGhpcyE=</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

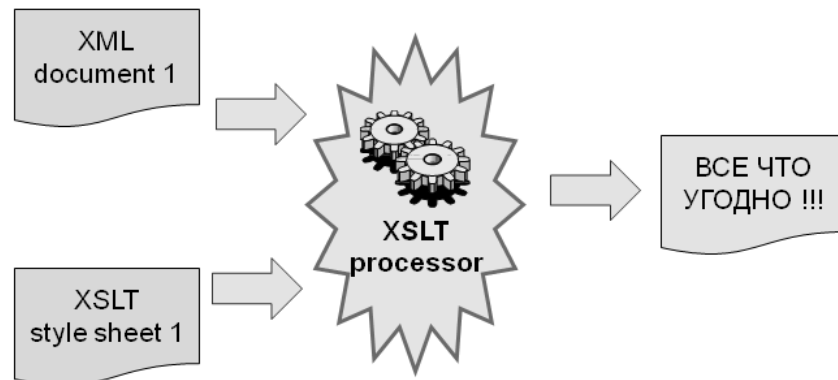
```
• <script type="text/javascript"
  src="js-xml-rpc/xmlrpc.js"></script>
```

```
// формируем сообщение
var msg = new XMLRPCMessage("system.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());
```

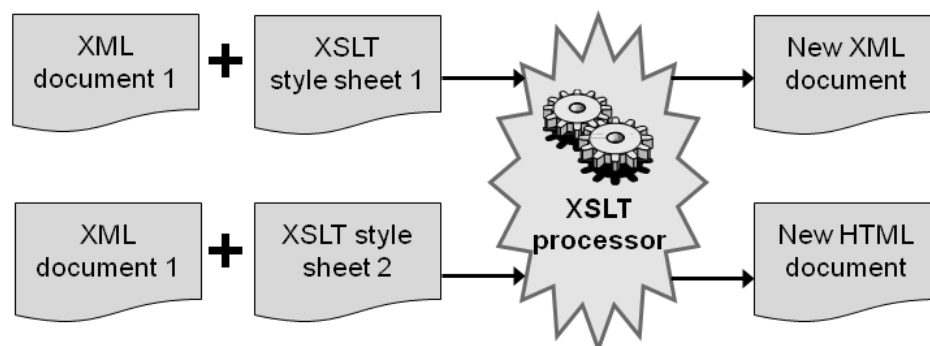
## Преобразование XML данных

- eXtensible Stylesheet Language
- XSL Transformation



## Что такое преобразования XSLT?

- Различные таблицы XSLT формируют различный результат



## 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>
```

## Программное преобразование XSLT - IE

```
// 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);
```

## Программное преобразование XSLT - FF

```
var xslStylesheet;
var xsltProcessor = new XSLTProcessor();

// load the xslt file, example1.xsl
var myXMLHttpRequest = new XMLHttpRequest();
myXMLHttpRequest.open("GET", "example1.xsl", false);
myXMLHttpRequest.send(null);

// get the XML document and import it
xslStylesheet = myXMLHttpRequest.responseXML;
xsltProcessor.importStylesheet(xslStylesheet);

// load the xml file, example1.xml
myXMLHttpRequest = new XMLHttpRequest();
myXMLHttpRequest.open("GET", "example1.xml", false);
myXMLHttpRequest.send(null);

var xmlSource = myXMLHttpRequest.responseXML;
var resultDocument = xsltProcessor.transformToDocument(xmlSource);
```

## Лабораторная работа 4

### Упражнение 1: Получение способов доставки в электронном магазине

- ♦ Откройте проводник Windows и найдите папку lab4<sup>1</sup>
- ♦ Найдите файл **index.html**, щелкните по нему правой кнопкой мышки и выберите «**Edit with Notepad++**»
- ♦ Изучите HTML код этой страницы. Обратите внимание на подключения двух дополнительных файлов:

```
<script type="text/javascript" src="xslt.js"></script>
<script type="text/javascript" src="xmlrpc.js"></script>
```

- ♦ Откройте эти файлы и изучите их программный код
- ♦ Найдите комментарий

```
/*
Задание 1.
...
*/
```

- ♦ Напишите функцию **getDeliveryMethods()** для получения способов доставки товаров электронного магазина. Эта функция должна обратиться к серверу XML-RPC **lab4-server.php**, и вызвать метод **eshop.getDeliveryMethods** (параметров нет). Сервер вернет XML-RPC ответ (примерное содержание можно посмотреть в файле **messages/getDeliveryMethods.xml**). Выведите это сообщение на экран с помощью функции **showXML()**, 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

---

<sup>1</sup> Лабораторные работы находятся в папке C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\



**Упражнение 2:      Отображение способов доставки на экране**

- ♦ Откройте проводник Windows и найдите папку lab4<sup>2</sup>
- ♦ Найдите файл **delivery.xsl**, щелкните по нему правой кнопкой мышки и выберите «**Edit with Notepad++**»
- ♦ Изучите код преобразования XSLT. Обратите внимание, это преобразование формирует элемент **<select>** со вложенными элементами **<option>**. Закройте этот файл, не изменяя его.
- ♦ В файле **index.html** найдите комментарий

```
/*  
Задание 2.  
...  
*/
```

- ♦ Напишите функцию **showDelivery()**. Эта 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**
- ♦ Результат преобразования выведите в объект HTML

```
<div id="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 событие **window.onload** и проверьте работу скрипта в различных браузерах

---

<sup>2</sup> Лабораторные работы находятся в папке **C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

### Упражнение 3: Получение данных о заказе

- ♦ В файле **index.html** найдите комментарий

```
/*  
Задание 3.  
...  
*/
```

- ♦ Напишите функцию **calculateOrder()**, которая сформирует XML-RPC сообщение и вызовет метод **eshop.calculateOrder**, передавая ему следующие параметры:
  - **sum**                    число, сумма заказа
  - **deviveryId**           код способа доставки (значение value списка доставки)
- ♦ Данный метод вернет детализацию расчета общей суммы заказа. Параметры **sum** и **deviveryId** Вы должны получить из элементов **txtOrderSum** (`<input id="txtOrderSum" type="text" value="1000" />`) и сформированным Вами списком **selDelivery** (см. упражнение 2)
- ♦ Передайте сформированную XML-RPC строку методом **POST** (не забываем про правильный Content-type!) серверу **lab4-server.php**
- ♦ Получите и выведите с помощью функции **alert()** результат работы сервера
- ♦ Это сообщение можно преобразовать к строке с помощью функции **showXML()**

**Упражнение 4:      Отображение данных о заказе**

- ♦ В файле **index.html** найдите комментарий

```
/*  
Задание 4.  
...  
*/
```

- ♦ Напишите функцию **showOder()**, которая получает параметр **dom** (ответ XML-RPC сервера), и поставьте ее вызов в функцию следующим образом:

```
// Если нет ошибок, отобразить данные о заказе на экране  
if (!isError(dom)) showOder(dom);
```

- ♦ Эта функция должна, используя заранее загруженное XSLT преобразование (переменная **xslOrder**), произвести преобразование XML данных (полученных с сервера) в переменной **dom** и результат вывести на экран в элемент **divOrder**

```
<div id="divOrder"></div>
```

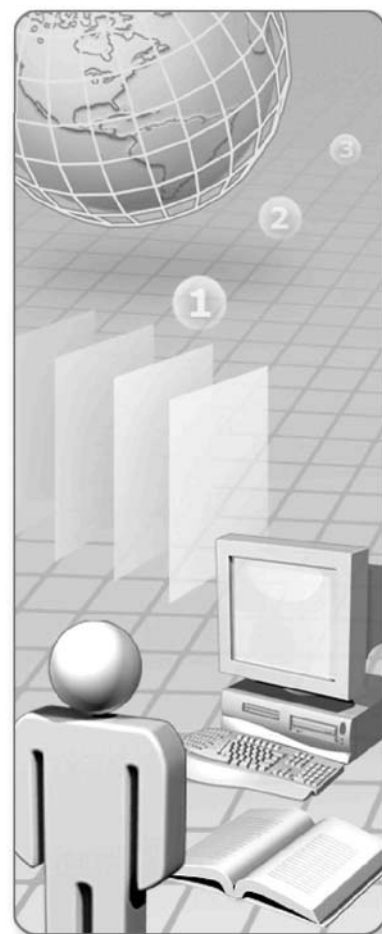
- ♦ Проверьте работу скрипта в различных браузерах

## Выводы

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

## Модуль 5

# Использование XML Web-сервисов Протокол SOAP



## Использование XML Web-сервисов Протокол SOAP

- Проблемы XML-RPC
- Сервис-ориентированная архитектура (SOA)
- SOAP (Обзорно)
- XML Web-сервисы
- Формирование и разбор SOAP сообщений
- Пример работы с XML Web-сервисами

### Проблемы XML-RPC

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

## XML схемы (обзорно)

- <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 xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-
  instance"
  xsi:noNamespaceSchemaLocation=
  "country.xsd">
  <название>Франция</название>
  <население>59.7</население>
</страна>
```

## Соответствие схемы и документа

### ● XML Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.camera.org"
  xmlns="http://www.camera.org"
  xmlns:nikon="http://www.nikon.com"
  xmlns:olympus="http://www.olympus.com"
  xmlns:pentax="http://www.pentax.com"
  elementFormDefault="unqualified">
```

### ● Документ

```
<?xml version="1.0"?>
<my:camera xmlns:my="http://www.camera.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://www.camera.org/Camera.xsd">
```

## SOAP – Simple Object Access Protocol

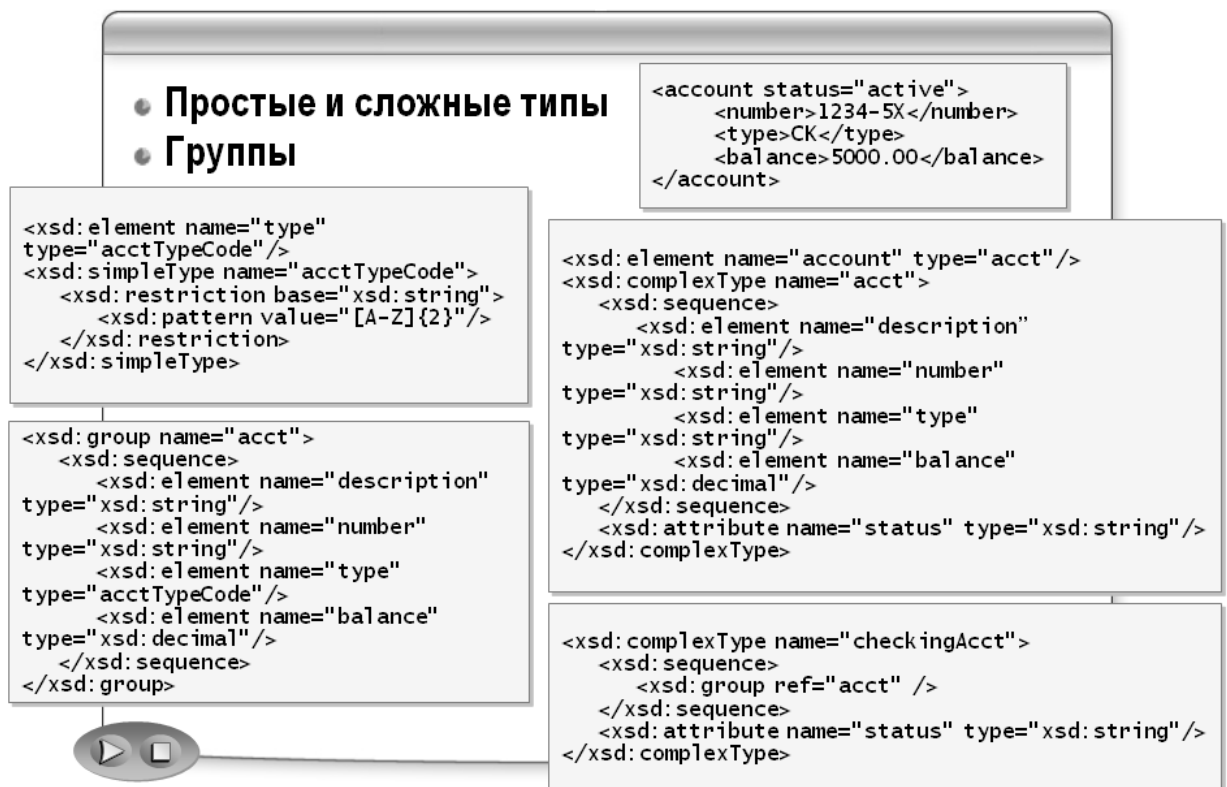
- Протокол обмена структурированными сообщениями
- Промышленный стандарт построения распределенных приложений  
<http://www.w3.org/TR/soap/>
- Является расширением языка XML-RPC
- Может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, MSMQ и др.
- Основной протокол реализации XML Web Services (XML Веб служб)



## Описание SOAP документов и сообщений

- XML схемы
- Типы данных
- Объекты и сложные типы

## Типы данных в XML-схеме (обзорно)



## Сложные типы (обзорно)

### • Контейнеры

xsd:sequence

xsd:choice

xsd:all

### • Наследование

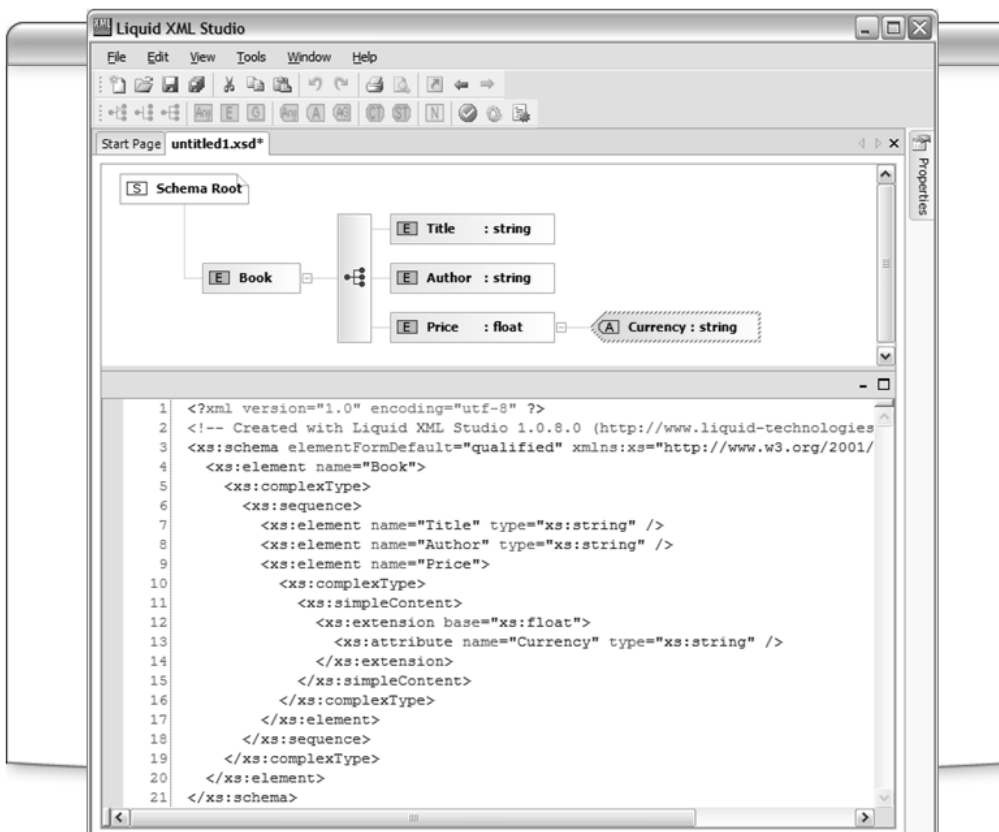
restriction

extension

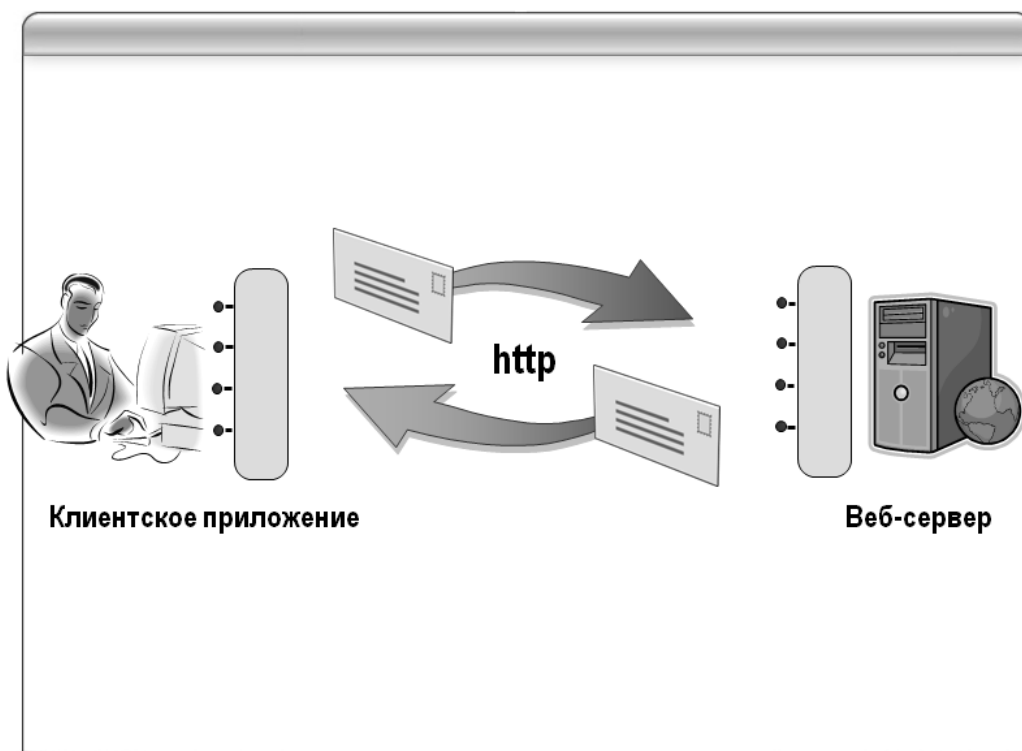
```
<xsd:complexType name="acct">
  <xsd:sequence>
    <xsd:element name="description"
      type="xsd:string"/>
    <xsd:element name="number" type="xsd:string"/>
    <xsd:element name="type" type="acctTypeCode"/>
    <xsd:element name="balance"
      type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute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

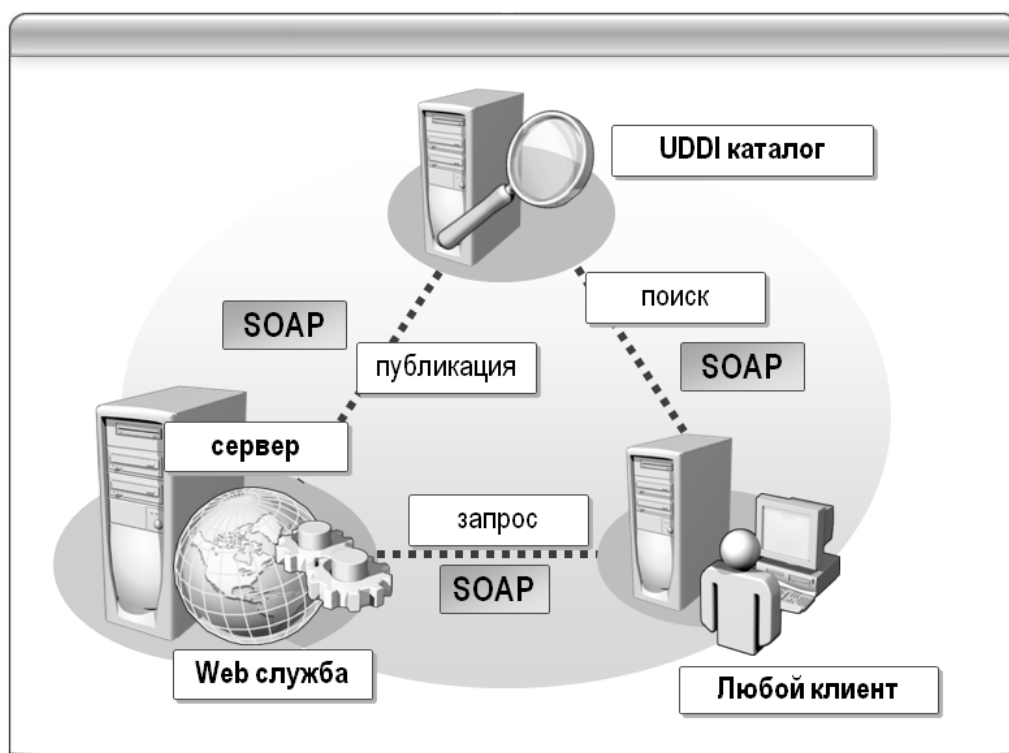
## Инструменты для создания XML схем



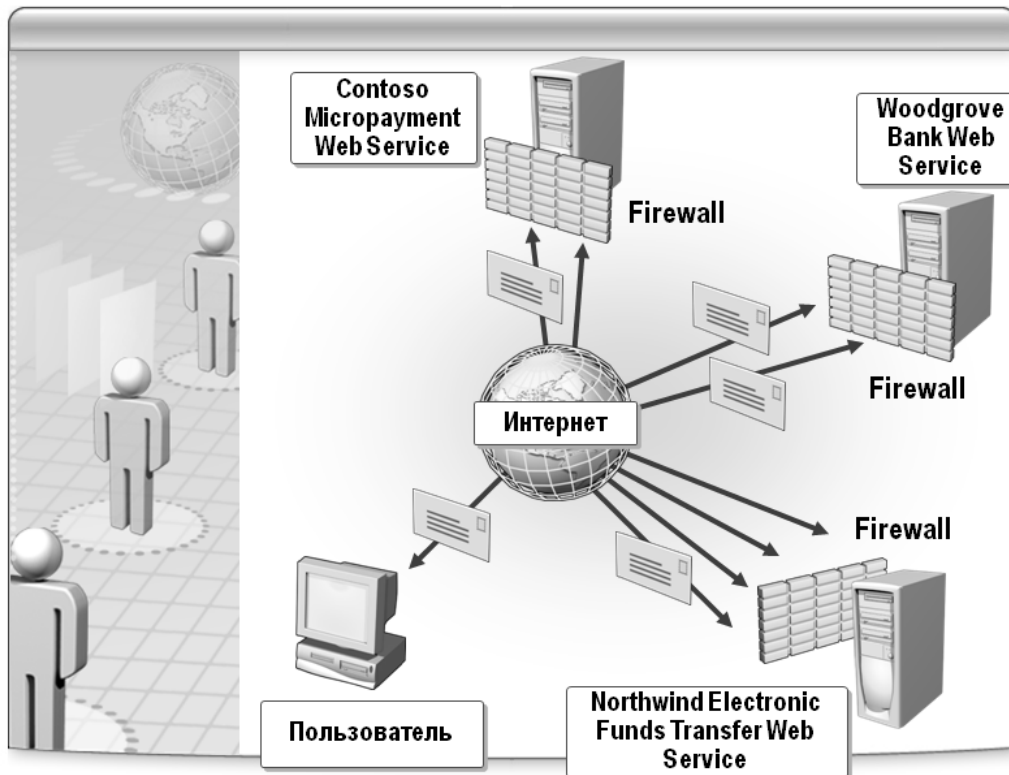
## XML Web сервисы



## Основные операции SOAP



## Пример использования XML Web сервисов



## 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="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <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>
```

## Пример запроса SOAP 1.1

```
POST /DailyInfoWebServ/DailyInfo.asmx HTTP/1.1
Host: www.cbr.ru
Content-Type: text/xml; charset=utf-8
Content-Length: length
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: length

<?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

## WSDL сообщения

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SampleServer1"
  targetNamespace="urn:SampleServer1"
  xmlns:typens="urn:SampleServer1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Сообщения метода getTime -->
  <message name="getTime"/>
  <message name="getTimeResponse">
    <part name="getTimeResult" type="xsd:string"/>
  </message>

  <!-- Сообщения метода sayHello -->
  <message name="sayHello">
    <part name="userName" type="xsd:string"/>
  </message>
  <message name="sayHelloResponse">
    <part name="sayHelloResult" type="xsd:string"/>
  </message>
```

## WSDL привязка

```

<!-- Привязка сообщений к методам -->
<portType name="ServerPortType">
  <operation name="getTime">
    <input message="typens:getTime"/>
    <output message="typens:getTimeResponse"/>
  </operation>
  <operation name="sayHello">
    <input message="typens:sayHello"/>
    <output message="typens:sayHelloResponse"/>
  </operation>
</portType>

```

## WSDL формат методов

```

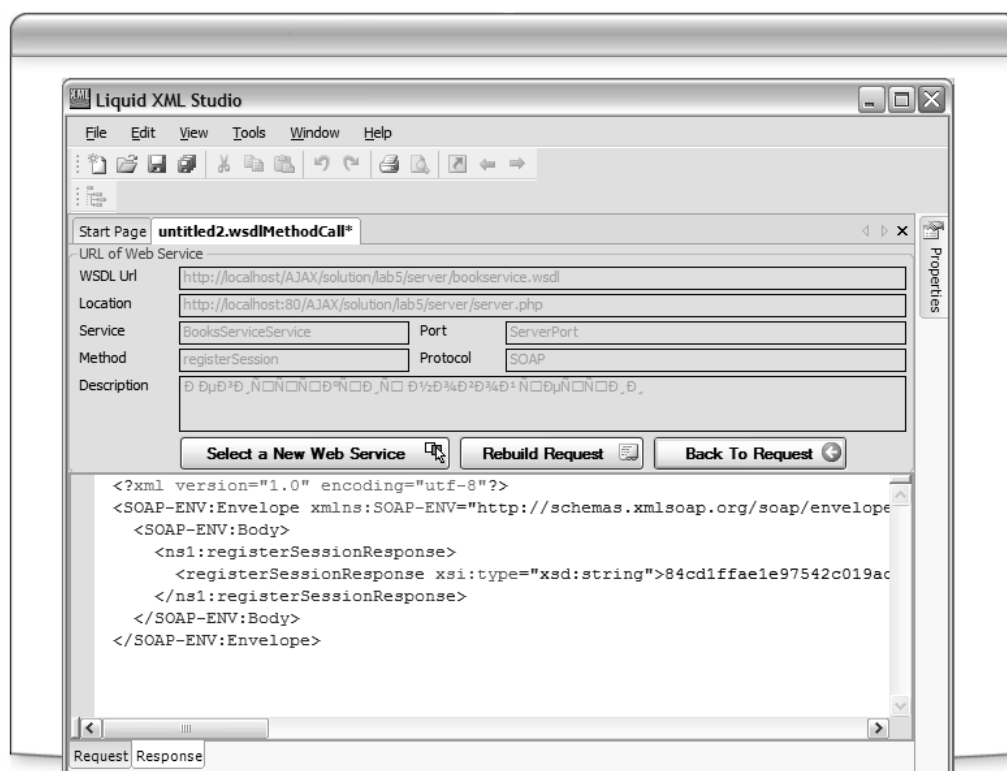
<!-- Формат методов -->
<binding name="ServerBinding" type="typens:ServerPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getTime">
    <soap:operation soapAction="urn:SampleServer1-getTime"/>
    <input>
      <soap:body namespace="urn:SampleServer1" use="encoded" encodingStyle=
    </input>
    <output>
      <soap:body namespace="urn:SampleServer1" use="encoded" encodingStyle=
    </output>
  </operation>
  <operation name="sayHello">
    <soap:operation soapAction="urn:SampleServer1-sayHello"/>
    <input>
      <soap:body namespace="urn:SampleServer1" use="encoded" encodingStyle=
    </input>
    <output>
      <soap:body namespace="urn:SampleServer1" use="encoded" encodingStyle=
    </output>
  </operation>
</binding>

```

## WSDL описание сервиса

```
<!-- Определение службы -->
<service name="SampleServer1Service">
  <port name="ServerPort" binding="typens:ServerBinding">
    <soap:address
      location="http://localhost:80/AJAX/demo/module5/sample_server_1/server.php"/>
  </port>
</service>
```

## Пример работы со службой





## Пример клиента XML Web-службы (C#)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ClientCS
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                SampleServer.SampleServer1Service client =
                    new ClientCS.SampleServer.SampleServer1Service();
                Console.WriteLine(client.getTime());
                Console.WriteLine(client.sayHello("Пользователь"));
                Console.WriteLine("Нажмите [Enter] для завершения...");
                Console.ReadLine();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Ошибка: " + ex.Message);
            }
        }
    }
}
```

## Пример обращения к XML Web службе

```
var req = getXMLHttpRequest()
req.open('POST', "gbook.asmx", false);

// Заголовок запроса SOAP
req.setRequestHeader("SOAPAction",
    "http://www.specialist.ru/XML-course/insertRecord", false);
req.setRequestHeader("Content-Type", "text/xml; charset=utf-8",
    false);
// Запрос
req.send(reqMessage);

var responseMessage = req.responseXML;
var resNodes =
    responseMessage.getElementsByTagName("insertRecordResult");
alert(resNodes[0].firstChild.nodeValue);
```

## Пример функции загрузки сообщения

```
// Функция загружает требуемый XML файл
// в синхронном режиме
function loadXML(url)
{
    var reqMessage = getXmlHttpRequest();
    reqMessage.open("GET", url, false);
    reqMessage.send(null);
    return reqMessage.responseXML;
}
```

## Пример функции модификации сообщения

```
// Функция модифицирует SOAP сообщение
function setParameter(tagName, value, message)
{
    var el =
        message.getElementsByTagName(tagName)[0];
    el.firstChild.nodeValue = value;
}
```

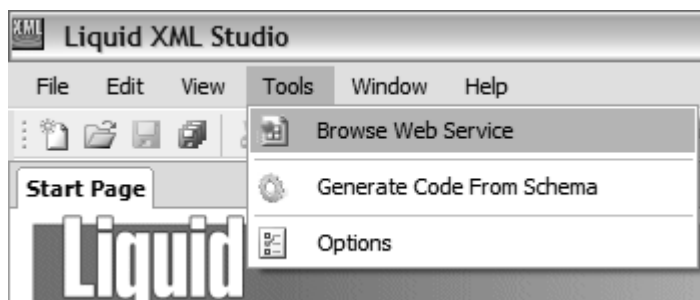
## Пример функции для отправки сообщения

```
// функция отправляет сообщение на сервер
function sendMessage(message, action, async, callback)
{
    var req = getXmlHttpRequest();
    // Если это асинхронный режим и указана функция callback,
    // ставим обработчик
    if (async && callback)
        req.onreadystatechange = function()
        {
            // Если данные получены, вызываем callback функцию
            if (req.readyState == 4)
                callback(req);
        }
    req.open("POST", server, async);
    req.setRequestHeader("SOAPAction", action);
    req.send(message);
    if (!async)
        return req.responseXML; // Синхронный вызов
}
```

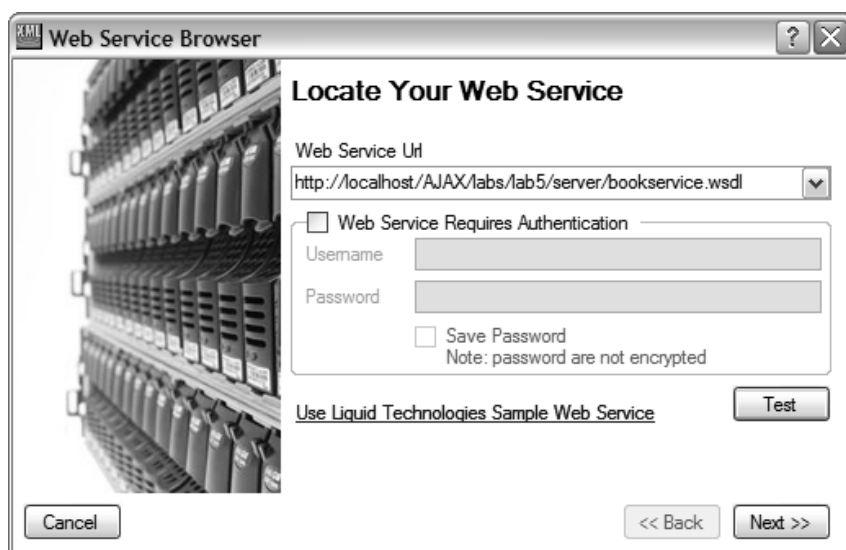
## Лабораторная работа 5

### Упражнение 1: Сохранение XML файлов с текстом SOAP сообщений

- ♦ Откройте приложение **Liquid XML Studio 2008**  
**Start → All Programs → Liquid XML Studio 2008 → Liquid XML Studio 2008 (Freeware)**
- ♦ Выберите в меню **Tools → Browse Web Service**

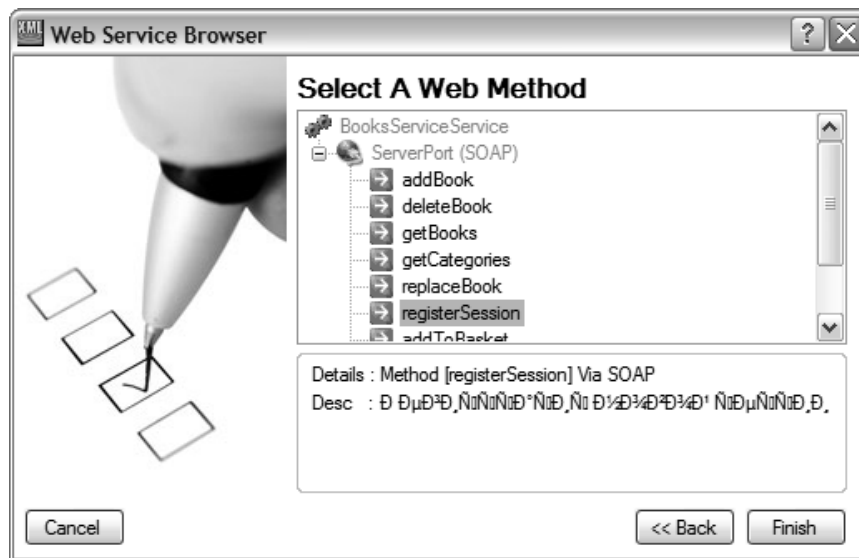


- ♦ В поле Web Service Url введите следующий URL:  
**`http://localhost/AJAX/labs/lab5/server/bookservice.wsdl`**



- ♦ Нажмите кнопку **[OK]**
- ♦ Если Вы сделали все правильно, Вы увидите сообщение «Connected OK», если же нет – убедитесь в правильности адреса и повторите этот шаг
- ♦ Нажмите кнопку **[Next >>]**

- ♦ Выберите метод **registerSession**



- ♦ Нажмите кнопку **[Finish]**
- ♦ Скопируйте и сохраните текст SOAP-сообщения в папку **server/messages** с именем "*название\_метода-request.xml*", то есть "**registerSession-request.xml**"
- ♦ Обратите внимание, в этом сообщении нет параметров. Если же в сообщении есть параметры, их следует указать, чтобы сервер вернул правильные данные



- ♦ Нажмите кнопку
- ♦ Скопируйте и сохраните текст SOAP-сообщения в папку **server/messages** с именем "*название\_метода-response.xml*", то есть "**registerSession-response.xml**"



- ♦ Нажмите кнопку
- ♦ В появившемся диалоге нажмите кнопку **[Next >>]**
- ♦ Выберите метод **addToBasket** и нажмите кнопку **[Finish]**
- ♦ Повторите все шаги и сохраните SOAP сообщения для этого метода
- ♦ Аналогичным образом сохраните SOAP сообщения для методов: **addToBasket**, **getBasket**, **emptyBasket**
- ♦ В результате в папке **server/messages** должно быть **18 файлов**. Проверьте это по списку

Имя	Размер
addBook-request.xml	1 КБ
addBook-response.xml	1 КБ
addToBasket-request.xml	1 КБ
addToBasket-response.xml	1 КБ
deleteBook-request.xml	1 КБ
deleteBook-response.xml	1 КБ
emptyBasket-request.xml	1 КБ
emptyBasket-response.xml	1 КБ
getBasket-request.xml	1 КБ
getBasket-response.xml	1 КБ
getBooks-request.xml	1 КБ
getBooks-response.xml	4 КБ
getCategories-request.xml	1 КБ
getCategories-response.xml	3 КБ
registerSession-request.xml	1 КБ
registerSession-response.xml	1 КБ
replaceBook-request.xml	1 КБ
replaceBook-response.xml	1 КБ

### Упражнение 2: Регистрация новой сессии пользователя

- ♦ Откройте проводник Windows и найдите папку lab5<sup>1</sup>
- ♦ Найдите файл **index.html**, щелкните по нему правой кнопкой мышки и выберите «**Edit with Notepad++**»
- ♦ Изучите HTML код этой страницы. Обратите внимание на подключения двух дополнительных файлов:

```
<script type="text/javascript" src="xslt.js"></script>
<script type="text/javascript" src="xmltools.js"></script>
```

- ♦ Откройте файл **xmltools.js** и изучите программный код в этом файле
- ♦ Найдите комментарий

```
/*
Задание 2.
...
*/
```

- ♦ Напишите функцию **registerSession()**, которая вызывая SOAP-метод **registerSession**, регистрирует новую сессию пользователя. URI этого метода **urn:SampleServer2-registerSession**
- ♦ Используя функцию **loadXML()** загрузите сообщение **server/messages/registerSession-request.xml**
- ♦ Выполните асинхронный вызов SOAP метода **registerSession** (см. uri) с помощью функции **sendMessage()**. Пример такого вызова можно посмотреть в уже написанной функции **showBooks()**
- ♦ Используйте сервер **server/server.php**
- ♦ Напишите функцию **registerSessionCallack(req)**, которая будет использоваться в асинхронном вызове сервера
- ♦ В этой функции получите SOAP сообщение ответа сервера (**req.responseXML**) и прочитайте в нем элемент **registerSessionResponse** (см. файл **registerSession-response.xml**). В этом элементе ID новой сессии пользователя. Сохраните его в глобальную переменную **sessionId**.
- ♦ Для отладки выведите ID новой сессии пользователя в объект HTML **<div id="sessId"></div>**
- ♦ Поставьте вызов функции в событие **window.onload** и проверьте работу в разных браузерах

---

<sup>1</sup> Лабораторные работы находятся в папке **C:\Program Files\Apache Group\Apache\htdocs\ajax\labs\**

**Упражнение 3:      Добавление книг в корзину**

- ♦ Найдите комментарий

```
/*  
Задание 3.  
...  
*/
```

- ♦ Напишите код функции **addToBasket(bookId)**, которая вызывая SOAP-метод **addToBasket**, добавляет книгу в корзину. URI этого метода **urn:SampleServer2-addToBasket**
- ♦ Для этого загрузите сообщение **addToBasket-request.xml**, используя функцию **loadXML()**
- ♦ В этом сообщении не забудьте установить параметры **sessionId**, **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, передавая построенное SOAP сообщение. Используйте сервер **server/server.php**
- ♦ Напишите функцию **addToBasketCallback(req)**, которая будет использоваться в асинхронном вызове сервера
- ♦ В этой функции сохраните ответ сервера (**req.responseXML**) в локальной переменной.
- ♦ Найдите и проверьте элемент **addToBasketResponse**, 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.
- ♦ Проверьте работу скрипта в различных браузерах.

### Упражнение 4: Отображение корзины на экране

- ♦ Найдите комментарий

```
/*  
Задание 4.  
...  
*/
```

- ♦ Напишите код функции **showBasket()**, которая вызывая SOAP-метод **getBasket**, добавляет книгу в корзину. URI этого метода **urn:SampleServer2-getBasket**
- ♦ Для этого 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, передавая построенное SOAP сообщение. Используйте сервер **server/server.php**
- ♦ Напишите функцию **showBasketCallback (req)**, 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 **loadXML()**
- ♦ Выполните XSLT-преобразование с помощью функции **xsltTransform()**, используя данные SOAP сообщения и загруженную таблицу преобразования.
- ♦ Результат преобразования выведите в HTML элемент **<div id="basketPlaceholder"></div>**
- ♦ Проверьте работу вашего сценария. Для этого просто в браузере посмотрите книги в любой категории и щелкните по любой книге.
- ♦ Проверьте работу скрипта в различных браузерах.



**Упражнение 5: Очистка корзины**

- ♦ Найдите комментарий

```
/*  
Задание 5.  
...  
*/
```

- ♦ Напишите код функции **emptyBasket ()**, которая вызывая SOAP-метод **emptyBasket**, удаляет книги из корзины. URI этого метода **urn:SampleServer2-emptyBasket**
- ♦ Для этого 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, передавая построенное SOAP сообщение. Используйте сервер **server/server.php**
- ♦ Напишите функцию **emptyBasketCallback(req)**, которая будет использоваться в асинхронном вызове сервера
- ♦ В этой функции сохраните ответ сервера (**req.responseXML**) в локальной переменной
- ♦ В этом результате (см. сообщение **emptyBasket-response.xml**) прочитайте элемент **emptyBasketResponse**. Если он равен **"true"** -- корзина очищена. Выведите пользователю сообщение об этом, и очистите элемент **<div id="basketPlaceholder"></div>**
- ♦ 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]**
- ♦ Проверьте работу скрипта в различных браузерах.

**Упражнение 6: Оптимизация сценария**

Измените сценарий так, чтобы сообщения и XSLT-преобразования загружались только один раз — при загрузке страницы, а не при каждом вызове методов. Этот шаг несколько удлинит загрузку страницы, но положительно скажется производительности.

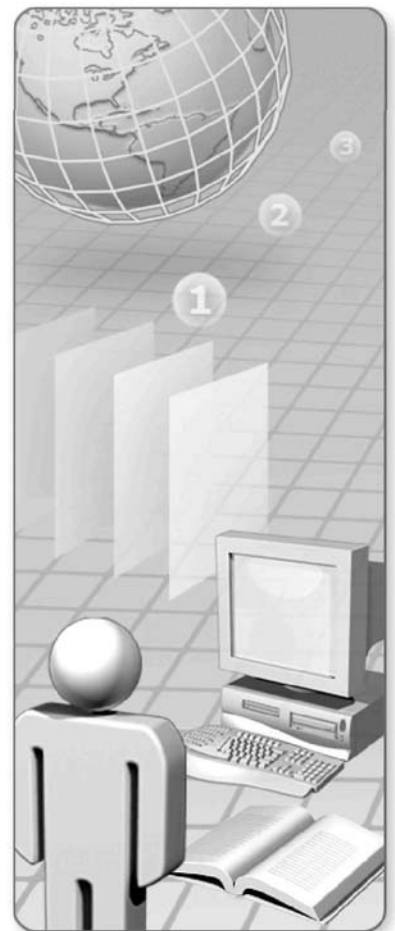
Реализуйте механизм извещения пользователя о окончании загрузки всех необходимых файлов и данных. Например, при старте страницы покажите пользователю сообщения «Данные загружаются», а после завершения загрузки уберите его.

## Выводы

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

## Модуль 6

# Безопасность и эффективность AJAX приложений



## Безопасность и эффективность AJAX приложений

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

## Вопросы безопасности AJAX приложений

- Аутентификация и авторизация пользователя
- Защита трафика

## Аутентификация и авторизация пользователя

- Аутентификация (*Authentication*) — подтверждение подлинности субъекта
- Авторизация (*Authorization*) — подтверждение прав субъекта на доступ к защищаемым объектам. Обычно проходит после аутентификации

## Аутентификация средствами HTTP RFC2617 и др. (обзорно)

- Basic аутентификация
- Digest аутентификация
- NTLM
- Kerberos
- X.509

## Запрос и прохождение аутентификации

```
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

## Пример включения базовой аутентификации (сервер Apache)

```
# Требуется аутентификация
AuthType Basic
AuthName "Private zone. Only for valid users!"
AuthUserFile "htdocs/base_auth/.htpasswd"
require valid-user
```

## Доступ к защищенным объектам

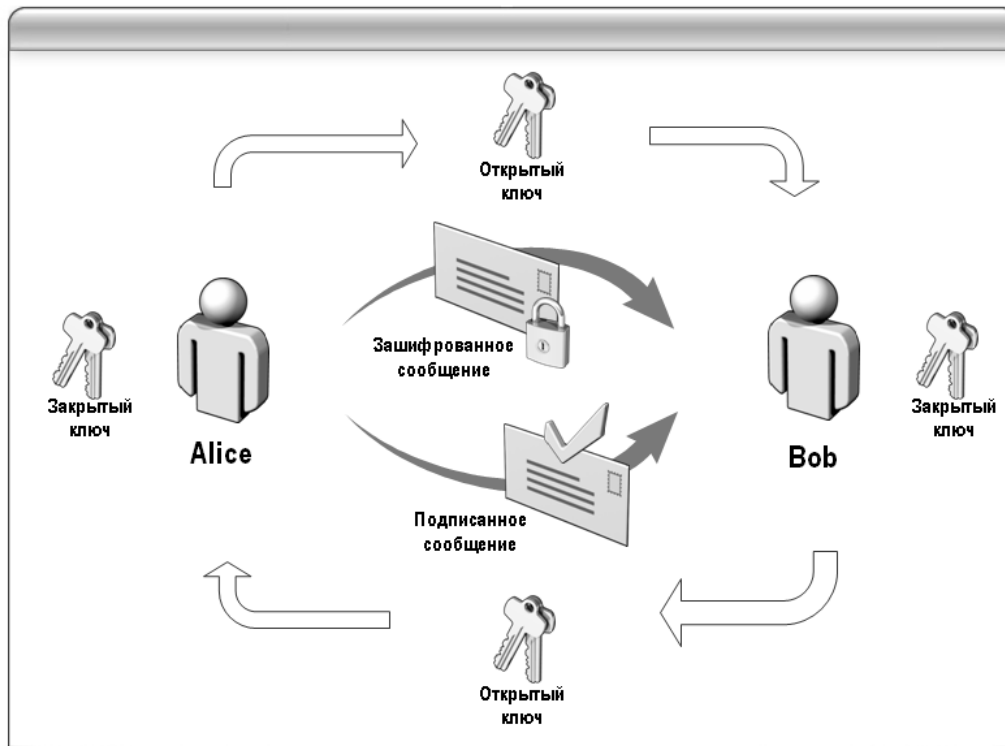
```
function getSecretFile()
{
    var login = "vasya";
    var password = "password";

    var req = getXmlHttpRequest();
    req.onreadystatechange = function()
    {
        if (req.readyState != 4) return;
        alert(req.responseText);
    }
    req.open("GET", "base_auth/secret.txt", true,
        login, password);
    req.send(null);
}
```

## Хеширование *(обзорно)*

- Хеширование (*хэширование*, англ. hashing) — преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины таким образом, чтобы изменение входных данных приводило к непредсказуемому изменению выходных данных
- “Hello, world” → bc6e6f16b8a077ef5fbc8d59d0b931b9
- “password” → 5f4dcc3b5aa765d61d8327deb882cf99
- “” → d41d8cd98f00b204e9800998ecf8427e

## Несимметричная криптография (обзорно)



## Защита трафика (SSL/TLS)

- Решает следующие проблемы:

- **Шифрование данных** (невозможность перехвата)
- **Подтверждение сервера** (невозможность подделки сервера)
- **Подтверждение клиента** (аутентификация клиента)



## Защита данных программными средствами

- **Правило № 1:**

Никогда, ни при каких обстоятельствах, не следует «изобретать» или использовать свои собственные (кустарные, «секретные») алгоритмы шифрования и аутентификации!

## Аутентификация на прикладном уровне

- **Безопасное хранение и безопасная передача пароля (например, RFC2898)**

*хэш(...(salt + хэш(salt + password))*

- **Алгоритмы хэширования:**

MD4	(RFC 1320)
MD5	(RFC 1321)
SHA1	(FIPS PUB 180-1)

## Пример функции многократного хэширования

```
function getSaltedHash(password, salt,
                        iterationCount)
{
    var saltedHash = password;
    if (iterationCount < 1) iterationCount = 1;
    for (var i=0; i<iterationCount; i++)
        saltedHash = hex_md5(salt + saltedHash);
    return saltedHash;
}
```

## Лабораторная работа 6.1

### Упражнение 1: Запрос «соли» хеширования с сервера

В этой лабораторной работе используется надежная передача пароля с помощью многократного хеширования пароля со случайной строкой. При формировании пароля в базе данных сохраняется "соль" - случайная строка и число итераций хеширования. Для передачи пароля между клиентом и сервером, хэш пароля в БД также "засоливается", то есть многократно хэшируется со случайной строкой. В этом задании вы передадите введенный E-mail пользователя на сервер с помощью объекта User. Сервер вернет вам также объект User, у которого будут инициализированы свойства энтропии, то есть объекты "соли". Этих свойств два: та энтропия, которая использовалась при хешировании пароля в БД, и новая энтропия, которая сгенерирована сервером для текущего сеанса связи. Вы должны сначала прохэшировать введенный пользователем пароль с "солью" `user.dbEntropy.salt` указанное число раз (`user.dbEntropy.iterationCount`) и полученную строку еще раз прохэшировать с "солью" `user.transferEntropy.salt` указанное число итераций (`user.transferEntropy.iterationCount`).

Результирующую строку следует записать в свойство `user.password` и передать на сервер. Если пользователь ввел правильный пароль, сервер вернет вам объект, у которого установлено свойство `user.name`, иначе это свойство будет пустым. Такой способ передачи пароля позволяет говорить о некоторой защищенности данных пользователя.

- ♦ Допишите функцию `loginUser()`. После комментария *"// Создадим объект User"* создайте новый объект User, установив у него свойство `email` из переменной `email`
- ♦ Передайте асинхронным вызовом этот объект на сервер (адрес сервера - глобальная переменная `server`), сериализовав его в JSON строку.
- ♦ 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 функции `alert` ответ сервера
- ♦ Запишите, какие свойства у полученного объекта установлены
- ♦ Обратите внимание на значения объектов энтропии
- ♦ Для проверки работы используйте E-mail пользователей:  
**`vasyap@mail.ru`, `fedyas@mail.ru`, `masha@mail.ru`**  
У всех пользователей пароль - **`password`**

### Упражнение 2: Хеширование пароля

- ♦ Найдите в коде комментарий

```
/*  
** Задание 2.  
*/
```

- ♦ После комментария `"/ Хэширование"` произведите хеширование введенного пароля (свойство `user.password`) с помощью функции `getSaltedHash()`. Передайте этой функции "соль" `user.dbEntropy.salt` и число итераций `user.dbEntropy.iterationCount`
- ♦ Сохраните результат в свойство `user.password`
- ♦ Проведите второе хэширование, передавая "соль" `user.transferEntropy.salt` и число итераций `user.transferEntropy.iterationCount`
- ♦ Сохраните результат в свойство `user.password`
- ♦ Сериализуйте объект `user` в JSON-строку и асинхронно передайте его на сервер (URL - 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 функции `alert` ответ сервера
- ♦ Запишите, какие свойства у полученного объекта установлены
- ♦ Обратите внимание на значения свойства `user.name`

**Упражнение 3:      Вывод данных**

- ♦ Найдите в коде комментариев

```
/*  
** Задание 3.  
*/
```

- ♦ 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 объекте `<div id="divResult"></div>`
- ♦ Проверьте работу скрипта в различных браузерах

## Проблемы юзабилити AJAX приложений

- 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 (информацию страницы) 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

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

## Лабораторная работа 6.2

*(если позволяет время)*

### Упражнение 1: Загрузка главы книги

В этой лабораторной работе Вы будете загружать большие объемы данных. В качестве примера мы выбрали загрузку глав книги А. и Б. Стругацких «Понедельник начинается в субботу». Эта книга представлена в XML формате FB2 (<http://www.fictionbook.org>). Задача скрипта — показать очередную главу книги на экране. Поскольку это XML, наиболее эффективный способ для этого — XSLT преобразование в HTML.

При загрузке данных необходимо выводить пользователю предупреждающее сообщение о загрузке данных, так как эта загрузка может продолжаться некоторое время.

- ♦ Откройте файл **lab2.6/index.html**
- ♦ Найдите комментарий

```
/*
** Задание 1
*/
```
- ♦ Покажите сообщение пользователю о загрузке данных. Для этого установите у объекта **divMessageLoad** свойство `display = "block"`
- ♦ Сформируйте и выполните асинхронный **GET** запрос к серверу **serverXml** (глобальная переменная), передавая ему параметер **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

- ♦ Найдите комментарий

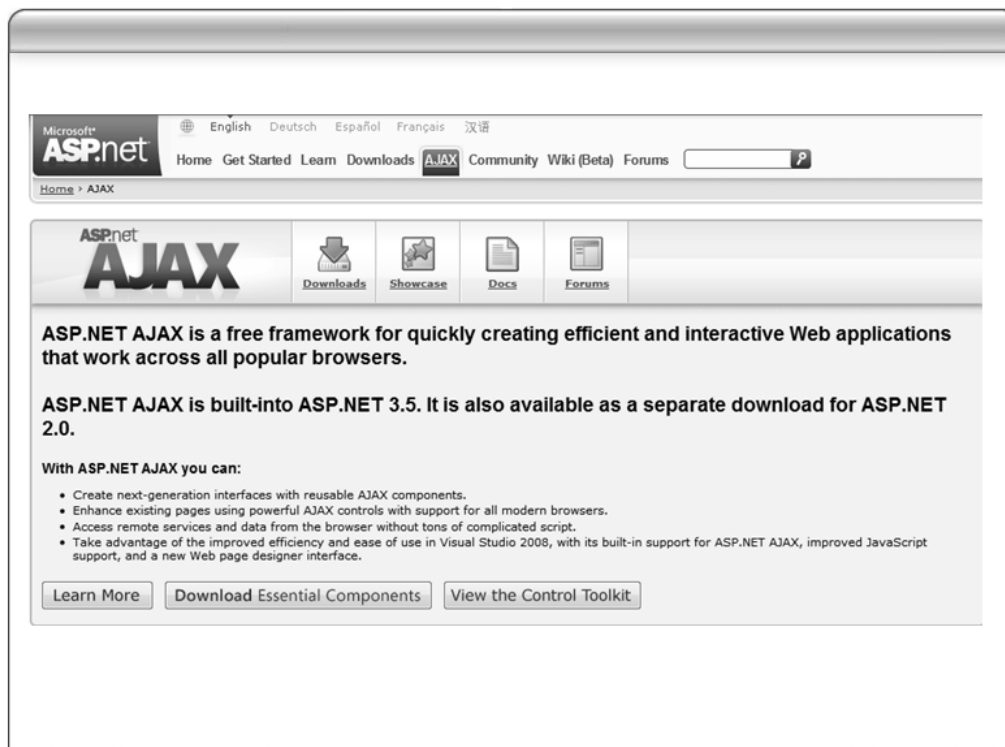
```
/*  
** Задание 2  
*/
```

- ♦ Допишите функцию **showChapter**. На основании переменной **currentChapter** сформируйте ссылки "Вперед" и "Назад", загружающие следующую и предыдущую главу книги. Для загрузки используйте функцию **getChapter(no)**.
- ♦ Выведите эти ссылки в объект **divChapters**
- ♦ Произведите XSLT преобразование полученной главы (переменная **xmlDOM**) с помощью загруженного преобразования **fb2html**. Преобразование выполните с помощью функции **xsltTransform(xmlDOM, fb2html)**
- ♦ Результат преобразования выведите в объект **divResult**
- ♦ Погасите сообщение пользователю о загрузке данных, устанавливая свойство **display = "none"** для объекта **divMessageLoad**
- ♦ Проверьте работу скрипта в различных браузерах

## Обзор решений AJAX

- **ASP.NET AJAX**
- **SAJAX**
- **Xajax**
- **JQuery**
- **другие**

## ASP.Net AJAX framework



The screenshot shows the Microsoft ASP.NET AJAX website. At the top, there is a navigation bar with the Microsoft ASP.NET logo and links for Home, Get Started, Learn, Downloads, AJAX (highlighted), Community, Wiki (Beta), and Forums. Below this is a sub-navigation bar with the ASP.NET AJAX logo and icons for Downloads, Showcase, Docs, and Forums. The main content area features a heading that reads: "ASP.NET AJAX is a free framework for quickly creating efficient and interactive Web applications that work across all popular browsers." This is followed by a paragraph stating: "ASP.NET AJAX is built-into ASP.NET 3.5. It is also available as a separate download for ASP.NET 2.0." Below this, a section titled "With ASP.NET AJAX you can:" lists four bullet points: "Create next-generation interfaces with reusable AJAX components.", "Enhance existing pages using powerful AJAX controls with support for all modern browsers.", "Access remote services and data from the browser without tons of complicated script.", and "Take advantage of the improved efficiency and ease of use in Visual Studio 2008, with its built-in support for ASP.NET AJAX, improved JavaScript support, and a new Web page designer interface." At the bottom of the main content area, there are three buttons: "Learn More", "Download Essential Components", and "View the Control Toolkit".

## 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**

## Обзор и подведение итогов



## Что дальше?

- Курс «XML и XSLT»
- Курс «2524 Developing XML Web Services Using Microsoft® ASP.NET»
- Курс «Профессиональное программирование на PHP»
- Курс «Безопасность Веб-приложений»

## **Курс «XML и XSLT. Современные технологии обработки данных для Web и Office 2007»**

Цель данного курса дать слушателям представление о современных XML технологиях, научить слушателей применять эти технологии на практике, продемонстрировать современные подходы к обработке XML данных. Курс рассчитан на подготовленных слушателей, имеющих опыт в построении сайтов и использовании Веб-технологий.

XML — это невероятно могучие технологии представления любых данных. Сегодня XML очень распространен в самых разных областях: с помощью XML описывают и представляют документы (офисные документы, книги, счета, платежки и др.), XML часто используется для обмена данными в программировании, для передачи информации в Вебе.

Предлагаемый курс рассматривает основные технологии XML, такие как описание данных, грамматика XML-разметки, XML схемы, XSL и XSLT (преобразования XML документов), а также возможности работы с XML, доступные в Microsoft Office 2003/2007

Этот курс может быть рекомендован широкому кругу слушателей, обучающихся или планирующих обучение по направлению «Разработка Веб-приложений». Курс является обязательным перед прохождением курсов «Профессиональное программирование на PHP» и «AJAX»

Курс также будет полезен тем, кто обладает знаниями в объёме программы, но хочет их систематизировать, а также повысить свою эффективность за счёт новых приёмов и методов работы.

После окончания курсы Вы будете уметь:

- Понимать назначение XML
- Формировать XML документов
- Использовать пространства имен XML
- Определять структуры документа с помощью DTD
- Определять структуры документа с помощью XML схем
- Понимать и использовать XSLT
- Производить преобразование данных с помощью XSLT
- Интегрировать XML данные в приложения и документы Microsoft Office 2003/2007

Подробности: <http://www.specialist.ru/programs/course.asp?idc=1424>

## **Курс «М2524 Разработка XML Web-сервисов с использованием Microsoft ASP .NET»**

Цель курса — получить знания и навыки, необходимые для разработки XML Web Services для решения сложных задач в среде распределённых приложений.

Вы получите практические навыки использования Web-сервисов при решении общих проблем в распределённых приложениях. Успешно окончив данный курс, Вы сможете определить место, построить, развернуть и поддерживать Web-сервисы.

Курс также будет полезен тем, кто обладает знаниями в объёме программы, но хочет их систематизировать, а также повысить свою эффективность за счёт новых приёмов и методов работы.

После окончания курса Вы будете уметь:

- Объяснять применение XML Web Services в качестве решения проблем при использовании традиционных подходов в разработке распределённых приложений
- Объяснять структуру решений XML Web Services
- Объяснять базовые технологии XML Web Services с использованием Microsoft .NET Framework
- Внедрять XML Web Service consumer с использованием Microsoft Visual Studio .NET
- Внедрять простой XML Web Service с использованием Visual Studio .NET
- Публиковать и применять XML Web Service
- Защищать XML Web Service
- Внедрять функцию кэширования в XML Web Service
- Оценивать задачи и проблемы, которые возникают при разработке реальных XML Web Service
- Внедрять нестандартные XML Web Services, например, HTML screen scraping и объединённые XML Web Services.

Подробнее: <http://www.specialist.ru/programs/course.asp?idc=325>

## Курс «Профессиональная разработка приложений на PHP5»

Цель курса — дать слушателям навыки и знания профессионального программирования на PHP, необходимые для сдачи сертификационного экзамена ZCE.

Данный курс является обязательным курсом в цепочке подготовки Веб-мастера к получению сертификации ZCE (Zend Certified Engineer).

Курс рекомендован слушателям, имеющим начальный опыт использования PHP, и желающим расширить свои знания и опыт.

В этом курсе рассматриваются сложные темы, такие как ООП, XML, Веб-сервисы, без которых немыслима профессиональная разработка приложений на PHP, что позволяет создавать сложные Интернет сайты, интегрированные с внешними данными.

Курс также будет полезен тем, кто обладает знаниями в объёме программы, но хочет их систематизировать, а также повысить свою эффективность за счёт новых приёмов и методов работы.

После окончания курсы Вы будете уметь:

- Использовать объектно-ориентированное программирование в PHP
- Использовать базы данных SQLite
- Использовать SAX и DOM разбор XML документов в PHP
- Применять модуль SimpleXML
- Использовать XSLT преобразования в PHP
- Создавать и использовать XML Web сервисы и протокол SOAP
- Использовать графический модуль GD2

Подробности: <http://www.specialist.ru/programs/course.asp?idc=1382>

## У вас есть вопросы?

- Конференция «Клуб выпускников»  
<http://forum.specialist.ru/>
- Бесплатные семинары и консультации  
<http://www.specialist.ru/Events/Seminars>



## **Типовые решения лабораторных работ**

## Лабораторная работа 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Лабораторная работа 1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <script type="text/javascript">
    /*
    ** Задание 1: Напишите функцию создания объекта XmlHttpRequest
    **      function getXmlHttpRequest()
    **
    */
    function getXmlHttpRequest()
    {
      if (window.XMLHttpRequest)
      {
        try
        {
          return new XMLHttpRequest();
        }
        catch (e){}
      }
      else if (window.ActiveXObject)
      {
        try
        {
          return new ActiveXObject('Msxml2.XMLHTTP');
        } catch (e){}
        try
        {
          return new ActiveXObject('Microsoft.XMLHTTP');
        }
        catch (e){}
      }
      return null;
    }

    /*
    ** Задание 2
    **      function getBookByNumber(number)
    **
    */
    function getBookByNumber(number)
    {
      var request = getXmlHttpRequest();
      request.onreadystatechange = function()
      {
        if (request.readyState != 4) return;

        // Вывод запроса на экран
        var divResult = document.getElementById("divResult");
        divResult.firstChild.nodeValue = request.responseText;
      }

      // Запрос
      request.open("GET", "getbooktxt.php?num=" + number, true);
      request.send(null);
    }

    /*
```

```

    ** Задание 3: Напишите код обработчика нажатия на кнопку
    ** Эта функция должна прочитать введенное значение в поле txtNum
    ** и вызвать функцию getBookByNumber(value)
    **     function showBook()
    */
function showBook()
{
    var txtNum = document.getElementById("txtNum");
    getBookByNumber(txtNum.value);
}
</script>
</head>
<body>
    <h1>Лабораторная работа 1</h1>
    <form onsubmit="return false">
        <label for="txtNum">Введите номер книги:</label>
        <input id="txtNum" type="text" />
        <!-- Задание 4: не забудьте поставить обработчик нажатия -->
        <button onclick="showBook()">Поиск</button>
    </form>
    <div id="divResult">&nbsp;</div>
</body>
</html>
```

## Лабораторная работа 2.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Книги по категориям</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <link rel="stylesheet" type="text/css" href="mod_2_1.css" />
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript">
    /*
    ** Задание 1: Заполните элемент selCategory списком категорий книг.
    ** Для этого напишите функцию fillCategories и вызовите ее из
    ** события window.onload. Список категорий вы можете получить
    ** из скрипта getcategories.php в виде текстовых строк
    ** разделенными \n - символ новой строки
    ** Формат строк: "код:название", например "1:Web"
    */
    function fillCategories()
    {
      // Запрос к серверу
      var req = getXmlHttpRequest();
      req.onreadystatechange = function()
      {
        if (req.readyState != 4) return;
        // Список selCategory
        var selCategory =
          document.getElementById("selCategory");
        // Получим строку ответа
        var responseText = new String(req.responseText);
        // Разделим строку на массив
        var cats = responseText.split("\n");
        // Создадим необходимое количество
        // элементов option с кодами категорий
        for (var i = 0; i < cats.length; i++)
        {
          if (cats[i] == '') continue;
          // Разделим строку по символу ":"
          var parts = cats[i].split(":");
          // Создадим новый элемент option
          var option =
            document.createElement("option");
          option.setAttribute("value", parts[0]);
          var optionText =
            document.createTextNode(parts[1]);
          option.appendChild(optionText);
          selCategory.appendChild(option);
        }
        // Сделаем список выбора с
        // нужным числом элементов
        selCategory.size = selCategory.options.length;
      }
      // Метод GET
      req.open("GET", "getcategories.php", true);
      req.send(null);
    }
    // При завершении загрузки страницы
    window.onload = function()
    {
```

```
        fillCategories();
    }
</script>
</head>
<body>
    <h1>Книги по категориям</h1>
    <form onsubmit="return false">
        <div>
            <label for="selCategory">Категория</label>
            <select id="selCategory"></select>
            <button onclick="alert('Это следующая лабораторная
работа')">Показать</button>
        </div>
    </form>
</body>
</html>
```

## Лабораторная работа 2.2

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
    <title>Книги по категориям</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Language" content="ru" />
    <link rel="stylesheet" type="text/css" href="mod_2_2.css" />
    <script type="text/javascript" src="xmlhttprequest.js"></script>
    <script type="text/javascript">
        /*
        ** Задание 1: Заполните элемент selCategory списком категорий книг.
        ** Для этого напишите функцию fillCategories и вызовите ее из
        ** события window.onload. Список категорий вы можете получить
        ** из скрипта getcategories.php в виде текстового документа со строками,
        ** разделенными \n - символ новой строки
        ** Формат строк: "код:название", например "1:Web"
        */
        function fillCategories()
        {
            // Запрос к серверу
            var req = getXmlHttpRequest();
            req.onreadystatechange = function()
            {
                if (req.readyState != 4) return;
                // Список selCategory
                var selCategory =
                    document.getElementById("selCategory");
                // Получим строку ответа
                var responseText = new String(req.responseText);
                // Разделим строку на массив
                var cats = responseText.split("\n");
                // Создадим необходимое количество
                // элементов option с кодами категорий
                for (var i = 0; i < cats.length; i++)
                {
                    if (cats[i] == '') continue;
                    // Разделим строку по символу ":"
                    var parts = cats[i].split(":");
                    // Создадим новый элемент option
                    var option = document.createElement("option");
                    option.setAttribute("value", parts[0]);
                    var optionText =
                        document.createTextNode(parts[1]);
                    option.appendChild(optionText);
                    selCategory.appendChild(option);
                }
                // Сделаем список выбора с нужным числом элементов
                selCategory.size = selCategory.options.length;
            }
            // Метод GET
            req.open("GET", "getcategories.php", true);
            req.send(null);
        }

        // При завершении загрузки страницы
        window.onload = function()
        {
            fillCategories();
        }

        /*
        ** Задание 2: Напишите функцию showBooks,
        ** выводящую все книги указанной категории

```

```

** в таблицу tableBooks. Список книг можно получить
** из сценария postbooksbycat.php
** передав ему параметром POST код категории.
** Список книг возвращается в формате
**      автор|название|картинка
** Поставьте вызов этой функции на кнопку "Показать"
*/

// Класс книга
function Book(author, title, image)
{
    this.author = author;
    this.title = title;
    this.image = image;
}

// Массив книг указанной категории
var books = new Array();

function showBooks()
{
    // Узнаем код выбранной категории
    var selCategory = document.getElementById("selCategory");
    if (selCategory.selectedIndex < 0)
    {
        alert("Необходимо выбрать категорию в списке");
        return;
    }
    var catId = selCategory.options[selCategory.selectedIndex].value;

    // Запрос к серверу
    var req = getXmlHttpRequest();
    req.onreadystatechange = function()
    {
        if (req.readyState != 4) return;
        // Получим строку ответа
        var responseText = new String(req.responseText);
        // Разделим строку на массив
        var bookStrings = responseText.split("\n");

        // Сформируем и заполним массив books
        books = new Array();
        for (var i = 0; i < bookStrings.length; i++)
        {
            if (bookStrings[i] == "") continue;
            var parts = bookStrings[i].split("|");
            books[books.length] = new Book(parts[0],
parts[1], parts[2]);
        }
        // Таблица tableBooks
        var tableBooks =
document.getElementById("tableBooks");
        // Очистка таблицы от предыдущей информации
        while (tableBooks.hasChildNodes())
            tableBooks.removeChild(tableBooks.lastChild);
        // Заполним таблицу данными по книгам
        for (var i = 0; i < books.length; i++)
        {
            // Создадим новый ряд таблицы
            var tr =
tableBooks.insertRow(tableBooks.rows.length);
            // Добавим ячейки в таблицу
            var tdAuthor = tr.insertCell(tr.cells.length);

            tdAuthor.appendChild(document.createTextNode(books[i].author));
            var tdTitle = tr.insertCell(tr.cells.length);

            tdTitle.appendChild(document.createTextNode(books[i].title));

```

```
        // Добавим подсветку при наведении мышки
        tr.onmouseover = new
            Function("trHighLight(this, '#fcc')");
        tr.onmouseout = new
            Function("trHighLight(this, '')");
        // Сохраним картинку книги в атрибуте
        // title элемента TR
        tr.title = books[i].image;
        // Добавим обработку щелчка
        tr.onclick = new Function("showImage(this)");
    }

    }

    // Метод POST
    var postData = "cat=" + catId;
    req.open("POST", "postbooksbycat.php", true);
    req.setRequestHeader("Content-Type", "application/x-www-form-
urlencoded");
    req.setRequestHeader("Content-Length", postData.length);
    req.send(postData);
}

/*
** Задание 3 (дополнительно)
** Измените сценарий так, чтобы список полученных книг
** запоминался в некоем массиве
** При щелчке по строке таблицы, проверьте наличие этого
** файла на сервере и
** выводите на экран изображение полученной книги
*/

// Функция подсветки ряда таблицы
function trHighLight(trObject, color)
{
    if (color != "")
        trObject.style.backgroundColor = color;
    else
        trObject.style.backgroundColor = "";
}

// Функция проверки файла на сервере
function isExists(url)
{
    // Запрос к серверу
    var req = getXmlHttpRequest();
    // Запрашиваем URL методом HEAD в синхронном режиме
    req.open("HEAD", url, false);
    req.send(null);
    // Если файл есть - статус == 200
    return (req.status == 200);
}

// Функция показа картинки
function showImage(trObject)
{
    // Путь к файлам изображений на сервере
    var imagePath = "../images/";
    var image = imagePath + trObject.title;

    var divBookInfo = document.getElementById("divBookInfo");
    var img = divBookInfo.getElementsByTagName("img")[0];

    if (isExists(image))
    {
        // Файл есть, покажем картинку
        img.src = image;
    }
}
```



```
        divBookInfo.style.display = "block";
    }
    else
    {
        // Файла нет, картинку не показываем
        img.src = "";
        divBookInfo.style.display = "";
    }
}

</script>
</head>
<body>
    <h1>Книги по категориям</h1>
    <form onsubmit="return false">
        <div>
            <label for="selCategory">Категория</label>
            <select id="selCategory"></select>
            <button onclick="showBooks()">Показать</button>
        </div>

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

    <table id="tableBooks"></table>

</body>
</html>
```

### Лабораторная работа 3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Пользователи на сайте</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <link rel="stylesheet" type="text/css" href="mod_3.css" />
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript" src="json2.js"></script>
  <script type="text/javascript">
    // Класс UserInfo
    function UserInfo()
    {
      this.name = "";
      this.login = "";
      this.password = "";
    }

    // Класс Билет сессии
    function Ticket()
    {
      this.id = "";
      this.valid = false;
    }

    /*
    ** Задание 1. Напишите сценарий отображения
    ** формы frmLogin при нажатии
    ** на кнопку "Вход" в блоке divUsers.
    */
    function showLoginForm()
    {
      document.body.style.backgroundColor = "gray";
      var frmLogin = document.getElementById("frmLogin");
      frmLogin.style.display = "block";
    }

    /*
    ** Задание 2. Напишите функцию проверки пользователя
    ** и поставьте ее на
    ** обработчик onclick кнопки в форме авторизации пользователя.
    ** Для проверки пользователя создайте объект класса UserInfo
    ** и передайте объект JSON в сценарий user_auth.php
    ** Этот сценарий вернет объект класса Ticket.
    ** Свойство Ticket.valid показывает пройденную авторизацию
    ** Сохраните билет (Ticket) в глобальной переменной
    */

    // Билет пользователя
    var ticket;

    // Функция проверки пользователя
    function validateUser()
    {
      // Формирование объекта UserInfo
      var txtLogin = document.getElementById("txtLogin");
```

```

var txtPassword = document.getElementById("txtPassword");
var userInfo = new UserInfo();
userInfo.login = txtLogin.value;
userInfo.password = txtPassword.value;

// Передача данных серверу
var jsonUserInfo = JSON.stringify(userInfo);
var req = getXmlHttpRequest();
req.onreadystatechange = function()
{
    if (req.readyState != 4) return;
    // Чтение полученного билета
    ticket = JSON.parse(req.responseText);
    // Проверка билета
    if (ticket.valid)
    {
        // Билет правильный
        var frmLogin =
            document.getElementById("frmLogin");
        frmLogin.style.display = "";
        document.body.style.backgroundColor = "";

        // Гашение кнопки ВХОД
        var btnEnter =
document.getElementById("divUsers").getElementsByTagName("button")[0];
        btnEnter.style.display = "none";

        // Отображение списка пользователей online
        showOnLineUsers();
    }
    else
    {
        // Сообщение об ошибке
        var divMessage =
            document.getElementById("divMessage");
        divMessage.style.display = "block";
    }
}
req.open("POST", "user_auth.php", true);
req.setRequestHeader("Content-Type", "text/plain");
req.setRequestHeader("Content-Length", jsonUserInfo.length);

req.send(jsonUserInfo);
}

/*
** Задание 3. На основе полученного билета
** (Ticket) покажите пользователю
** сообщение об ошибке, установив свойство CSS
** display = "block"
** объекту divMessage (не забудьте создать функцию гашения
** это сообщения, присвоив пустую строку в свойство
** CSS display! Это
** нужно сделать по нажатию кнопки в слое divMessage)
** Если же билет правильный, сохраните его в глобальной переменной
** и переходите к заданию 4
** Не забудьте погасить кнопку ВХОД в объекте divUsers
*/

// Функция гашения сообщения об ошибке
function hideErrorMessage()
{

```

```

        var divMessage = document.getElementById("divMessage");
        divMessage.style.display = "";
    }

    /*
    **     Задание 4. Напишите функцию запроса пользователей,
    **     которые находятся в режиме online
    **     Для этого передайте сценарию get_online_users.php
    **     Этот сценарий возвращает массив объектов UserInfo
    **     На основе этого массива выведите список
    **     пользователей online в
    **     список объекта divUsers.
    **     Установите таймер на вызов этой же функции в диапазоне 30 -
60 сек...
    */
    function showOnLineUsers()
    {
        var divUsers = document.getElementById("divUsers");
        var ul = divUsers.getElementsByTagName("ul")[0];

        // Запрос пользователей ONLINE
        var jsonTicket = JSON.stringify(ticket);
        var req = getXmlHttpRequest();
        req.onreadystatechange = function()
        {
            if (req.readyState != 4) return;

            // Получение списка пользователей
            users = JSON.parse(req.responseText);

            // Очистка списка пользователей
            while (ul.hasChildNodes())
                ul.removeChild(ul.lastChild);

            // Отображение списка пользователей
            for (var i = 0; i < users.length; i++)
            {
                var li = document.createElement("li");
                li.id = users[i].id;
                var liText =
                    document.createTextNode(users[i].name);
                li.appendChild(liText);
                ul.appendChild(li);
            }

            // Таймер на исполнение следующего цикла
            window.setTimeout("showOnLineUsers()", 3000);
        }
        req.open("POST", "get_online_users.php", true);
        req.setRequestHeader("Content-Type", "text/plain");
        req.setRequestHeader("Content-Length", jsonTicket.length);

        req.send(jsonTicket);
    }

</script>
</head>
<body>
    <h1>Пользователи на сайте</h1>

    <!-- форма входа -->

```

```
<form id="frmLogin" onsubmit="return false" class="block">
  <h2>Авторизация пользователя</h2>
  <div>
    <label for="txtLogin">Логин</label>
    <input id="txtLogin" type="text" />
  </div>
  <div>
    <label for="txtPassword">Пароль</label>
    <input id="txtPassword" type="password" />
  </div>
  <button onclick="validateUser()">Вход</button>
  <div id="divMessage" class="block">
    <h2>Ошибка</h2>
    <div>Неправильный логин или пароль!</div>
    <button onclick="hideErrorMessage()">Закрыть</button>
  </div>
</form>

<!-- Панель списка пользователей -->
<div id="divUsers" class="block">
  <h2>Пользователи на сайте</h2>
  <button onclick="showLoginForm()">Вход</button>
  <ul></ul>
</div>
</body>
</html>
```

## Лабораторная работа 4

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Расчет суммы товаров электронного магазина</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <link rel="stylesheet" type="text/css" href="lab4.css" />
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript" src="xslt.js"></script>
  <script type="text/javascript" src="xmlrpc.js"></script>
  <script type="text/javascript">
    // XSLT документы
    var xslDelivery, xslOrder;

    // Функция преобразует xml DOM документ в строку
    function showXML(xml)
    {
      if (window.XMLSerializer)
      {
        // Это Mozilla
        var serializer = new XMLSerializer();
        return serializer.serializeToString(xml);
      }
      else if (window.ActiveXObject)
      {
        // Это Internet Explorer
        return xml.xml;
      }
      else
      {
        return "Сериализация в строку не поддерживается!";
      }
    }

    // Функция проверяет и показывает сообщение об ошибке
    function isError(xmlDOM)
    {
      try
      {
        if (xmlDOM.getElementsByTagName("fault").length > 0)
        {
          var errorString =
xmlDOM.getElementsByTagName("string")[0].firstChild.nodeValue;
          alert("Ошибка!\n" + errorString);
          return true;
        }
        else return false;
      }
      catch (e)
      {
        alert("Ошибка обработки XML!\n" + xmlDOM);
        return true;
      }
    }

    /*
    Задание 1. Напишите функцию getDeliveryMethods()
    для получения способов доставки
```

```

эта функция должна обратиться к серверу XML-RPC lab4-server.php
и вызвать метод eshop.getDeliveryMethods (параметров нет)
Сервер вернет XML-RPC ответ
(примерное содержание - файл messages/getDeliveryMethods.xml)
Выведите это сообщение на экран с помощью функции showXML()
Не забудьте добавить вызов этой функции в window.onload
*/
function getDeliveryMethods()
{
    // формируем сообщение
    var msg =
        new XMLRPCMessage("eshop.getDeliveryMethods", "utf-8");
    var rawData = msg.xml();
    //alert(rawData);

    // Запрос сервера
    var req = getXmlHttpRequest();
    req.onreadystatechange = function()
    {
        if (req.readyState != 4) return;
        //Получаем DOM документ
        var dom = req.responseXML;
        // Если нет ошибок...
        if (!isError(dom)) showDelivery(dom);
    }
    req.open("POST", "lab4-server.php", true);
    req.setRequestHeader("Content-Type", "text/xml");
    req.setRequestHeader("Content-Length", rawData.length);
    req.send(rawData);
}

/*
Задание 2. Напишите функцию showDelivery(), которая, используя
преобразование delivery.xsl,
сформирует список выбора способа доставки
Результат преобразования выведите в элемент divDelivery
*/
function showDelivery(xmlDOM)
{
    var html = xsltTransform(xmlDOM, xslDelivery);
    //alert(html);
    var divDelivery = document.getElementById("divDelivery");
    divDelivery.innerHTML += html;
}

/*
Задание 3. Напишите функцию calculateOrder(), которая сформирует
XML-RPC сообщение и вызовет метод eshop.calculateOrder, передавая
следующие параметры:
    sum          - число, сумма заказа
    deliveryId   - код способа доставки (значение value списка
                  доставки)
Метод вернет детализацию расчета общей суммы заказа. Выведите это
сообщение
на экран с помощью функции showXML()
*/
function calculateOrder()
{
    // Параметры запроса
    var sum, deliveryId;

```

```
try
{
    sum = document.getElementById("txtOrderSum").value;
    selDelivery = document.getElementById("selDelivery");
    deviveryId =
    selDelivery.options[selDelivery.selectedIndex].value;
}
catch (e)
{
    alert("Параметры не определены!\n" + e);
    return false;
}

// Формируем сообщение
var msg = new
    XMLRPCMessage("eshop.calculateOrder", "utf-8");
msg.addParameter(sum);
msg.addParameter(deviveryId);
var rawData = msg.xml();
//alert(rawData);

// Запрос сервера
var req = getXmlHttpRequest();
req.onreadystatechange = function()
{
    if (req.readyState != 4) return;
    //Получаем DOM документ
    var dom = req.responseXML;
    // Если нет ошибок...
    if (!isError(dom)) showOder(dom);
}
req.open("POST", "lab4-server.php", true);
req.setRequestHeader("Content-Type", "text/xml");
req.setRequestHeader("Content-Length", rawData.length);
req.send(rawData);
}

/*
Задание 4. Напишите функцию showOder(), которая,
используя преобразование
order.xsl, сформирует таблицу с результирующей таблицей.
Результат преобразования выведите в элемент страницы divOrder
*/
function showOder(xmlDOM)
{
    var html = xsltTransform(xmlDOM, xslOrder);
    //alert(html);
    var divOrder = document.getElementById("divOrder");
    divOrder.innerHTML = html;
}

// Инициализация страницы
window.onload = function()
{
    // Загрузка преобразований
    var req = getXmlHttpRequest();
    req.open("GET", "delivery.xsl", false);
    req.send(null);
    xslDelivery = req.responseXML;

    req.open("GET", "order.xsl", false);
    req.send(null);
}
```



```
        xslOrder = req.responseXML;

        // Получим способы доставки
        getDeliveryMethods();
    }
</script>
</head>
<body>
    <h1>Расчет суммы заказа электронного магазина</h1>
    <form onsubmit="return false">
        <!-- Сумма заказа -->
        <div>
            <label for="txtName">Сумма заказа</label>
            <input id="txtOrderSum" type="text" value="1000" />
            <span>руб.</span>
        </div>
        <!-- Доставка -->
        <div id="divDelivery">
            <label for="selDelivery">Доставка</label>
        </div>
        <!-- Расчет стоимости заказа -->
        <div id="divOrder"></div>
        <button onclick="calculateOrder()">Расчитать</button>
    </form>
</body>
</html>
```

## Лабораторная работа 5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Выбор книг в корзину</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <link rel="stylesheet" type="text/css" href="lab5.css" />
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript" src="xslt.js"></script>
  <script type="text/javascript" src="xmltools.js"></script>
  <script type="text/javascript">
    /*
    **      Задание 1.  Сохранение SOAP сообщений в виде XML файлов
    **      Откройте приложение Liquid XML Studio 2008 (Freeware)
    **      Выберите в меню Tools --> Browse Web Service
    **      Введите URL:
    **      http://localhost/AJAX/labs/lab5/server/bookservice.wsdl
    **      (убедитесь в правильности этого URL, просто
    **      открывая его в браузере)
    **      Нажмите кнопку Test и убедитесь, что подключение
    **      к XML Web-сервису есть
    **      Нажмите кнопку [Next >>]
    **      Выберите метод registerSession и нажмите кнопку [Finish]
    **      Скопируйте и сохраните текст SOAP-сообщения
    **      в папку server/messages
    **      с именем "название_метода-request.xml",
    **      то есть "registerSession-request.xml"
    **      Переключитесь в Liquid XML Studio 2008 (Freeware)
    **      при необходимости в SOAP-сообщении укажите параметры
    **      и нажмите кнопку [Execute]
    **      Скопируйте и сохраните текст SOAP-сообщения
    **      в папку server/messages
    **      с именем "название_метода-response.xml",
    **      то есть "registerSession-response.xml"
    **      Повторите эти шаги для методов:
    **          addToBasket, getBasket, emptyBasket
    **
    */

    // URL сервера, взято из WSDL
    var serverBase = "server/";
    var server = serverBase + "server.php";

    // Текущий набор книг
    var xmlBooks = null;

    // Идентификатор сессии пользователя
    var sessionId = "";

    // функция показывает книги указанной категории
    function showBooks()
    {
      var selCategory =
document.getElementById("selectPlaceholder").firstChild;
      if (!selCategory) return;
```

```

        // Загрузка сообщения getBooks-request
        var msgGetBooks = loadXML(serverBase +
            "messages/getBooks-request.xml");
        setParameter("categoryId", selCategory.value, msgGetBooks);
        // Асинхронный запрос сервера
        sendMessage(msgGetBooks, "urn:SampleServer2-getBooks",
            true, showBooksCallBack);
    }

    // функция обратного вызова при завершении метода showBooks
    function showBooksCallBack(req)
    {
        // Считаем полученное сообщение
        xmlBooks = req.responseXML;
        // Загрузка преобразования
        var xsltBooks = loadXML(serverBase + "xslt/getBooks.xsl");
        // Отображение таблицы книг
        document.getElementById("tablePlaceholder").innerHTML =
        xsltTransform(xmlBooks, xsltBooks);
    }

    /*
    ** Задание 2. Регистрация новой сессии пользователя
    ** Напишите код функции registerSession(), которая
    ** вызывая SOAP-метод registerSession,
    ** регистрирует новую сессию пользователя. URI этого
    ** метода urn:SampleServer2-registerSession
    ** Для этого загрузите сообщение registerSession-request.xml
    ** (см. пример в функции showBooks()) и передайте его
    ** на сервер server/server.php
    ** Результат (сообщение registerSession-response.xml)
    ** проанализируйте: найдите в нем элемент
    ** <registerSessionResponse
    ** xsi:type="xsd:string">...</registerSessionResponse>
    ** и сохраните переданный идентификатор в глобальной
    ** переменной sessionId
    ** Для отладки выведите эту информацию в объект HTML
    ** <div id="sessId"></div>
    ** Вызовите эту функцию из автозагрузки
    */
    function registerSession()
    {
        // Загрузка сообщения registerSession-request
        var msgRegSession = loadXML(serverBase +
            "messages/registerSession-request.xml");
        // Асинхронный запрос сервера
        sendMessage(msgRegSession,
            "urn:SampleServer2-registerSession",
            true, registerSessionCallack);
    }

    // функция обратного вызова регистрации сессии
    function registerSessionCallack(req)
    {
        // Считаем полученное сообщение
        var xmlSessionId = req.responseXML;
        // Запомним ID сессии
        sessionId =
        xmlSessionId.getElementsByTagName("registerSessionResponse")[0].firstChild.no
        deValue;

        var divSessId = document.getElementById("sessId");
    }

```

```
        divSessId.innerHTML = sessionId;
    }

    /*
    **      Задание 3. Добавление книг в корзину
    **      Напишите код функции addToBasket(), которая вызывая
    **      SOAP-метод addToBasket,
    **      добавляет книгу в корзину. URI этого
    **      метода urn:SampleServer2-addToBasket
    **      Для этого загрузите сообщение addToBasket-request.xml
    **      (см. пример в функции showBooks()) и передайте его на
    **      сервер server/server.php
    **      В этом сообщении не забудьте установить параметры
    **      sessionId, bookId
    **      Внимательно изучите XML файл с примером сообщения.
    **      Получите результат (см. сообщение addToBasket-response.xml)
    **      и прочитайте
    **      элемент addToBasketResponse. Если он равен "true" -
    **      книга добавлена в корзину
    **      Выведите пользователю сообщение об этом.
    **      На следующей лабораторной работе
    **      Вы отобразите корзину
    */
    function addToBasket(bookId)
    {
        // Загрузка сообщения addToBasket-request
        var msgAddToBasket = loadXML(serverBase +
            "messages/addToBasket-request.xml");
        setParameter("sessionId", sessionId, msgAddToBasket);
        setParameter("bookId", bookId, msgAddToBasket);
        // Асинхронный запрос сервера
        sendMessage(msgAddToBasket,
            "urn:SampleServer2-addToBasket",
            true, addToBasketCallback);
    }

    // Функция обратного вызова добавления в корзину
    function addToBasketCallback(req)
    {
        // Считаем полученное сообщение
        var xmlResult = req.responseXML;
        // Результат выполнения операции
        var result =
xmlResult.getElementsByTagName("addToBasketResponse")[0].firstChild.nodeValue
;
        if (result == "true")
        {
            // Книга в корзину добавлена!
            showBasket()
        }
        else
        {
            // Ошибка
            alert("Ошибка добавления книги");
        }
    }
}
```

```

/*
** Задание 4. Отображение корзины на экране
** Напишите код функции showBasket(), которая вызывая
** SOAP-метод getBasket,
** получает и показывает книги в корзине. URI этого
** метода urn:SampleServer2-getBasket
** Для этого загрузите сообщение getBasket-request.xml
** и установите параметр sessionId
** (см. пример в функции showBooks()) и передайте его
** на сервер server/server.php
** Получите результат (см. сообщение getBasket-response.xml)
** и выполните
** XSL-преобразование с помощью таблицы getBasket.xsl
** Результат преобразования выведите в HTML элемент
** <div id="basketPlaceholder"></div>
*/
function showBasket()
{
    // Загрузка сообщения getBasket-request
    var msgGetBasket = loadXML(serverBase +
        "messages/getBasket-request.xml");
    setParameter("sessionId", sessionId, msgGetBasket);
    // Асинхронный запрос сервера
    sendMessage(msgGetBasket,
        "urn:SampleServer2-getBasket",
        true, showBasketCallback);
}

function showBasketCallback(req)
{
    // Считаем полученное сообщение
    var msgGetBasketResponse = req.responseXML;
    // Загрузка преобразования
    var xsltBooks = loadXML(serverBase + "xslt/getBasket.xsl");
    // Отображение таблицы книг
    document.getElementById("basketPlaceholder").innerHTML =
        xsltTransform(msgGetBasketResponse, xsltBooks);
}

/*
** Задание 5. Очистка корзины
** Напишите код функции emptyBasket(), которая вызывая
** SOAP-метод emptyBasket,
** получает и показывает книги в корзине. URI этого
** метода urn:SampleServer2-emptyBasket
** Для этого загрузите сообщение emptyBasket-request.xml и
** установите параметр sessionId
** (см. пример в функции addToBasket()) и передайте его на
** сервер server/server.php
** Получите результат (см. сообщение
** emptyBasket-response.xml) и прочитайте
** элемент emptyBasketResponse. Если он равен "true" -
** корзина очищена.
** Выведите пользователю сообщение об этом, и очистите элемент
** <div id="basketPlaceholder"></div>
*/

```

```
function emptyBasket()
{
    // Загрузка сообщения getBasket-request
    var msgEmptyBasket = loadXML(serverBase +
        "messages/emptyBasket-request.xml");
    setParameter("sessionId", sessionId, msgEmptyBasket);
    // Асинхронный запрос сервера
    sendMessage(msgEmptyBasket,
        "urn:SampleServer2-emptyBasket",
        true, emptyBasketCallback);
}

function emptyBasketCallback(req)
{
    // Считаем полученное сообщение
    var xmlResult = req.responseXML;
    // Результат выполнения операции
    var result =
xmlResult.getElementsByTagName("emptyBasketResponse")[0].firstChild.nodeValue
;
    if (result == "true")
    {
        // Корзина очищена
        document.getElementById("basketPlaceholder").innerHTML
= "";
    }
    else
    {
        // Ошибка
        alert("Ошибка очистки корзины");
    }
}

// Инициализация страницы
window.onload = function()
{
    // Загрузка сообщения getCategories
    var msgGetCat = loadXML(serverBase +
        "messages/getCategories-request.xml");
    // Синхронный запрос сервера
    var msgGetCatResponse = sendMessage(msgGetCat,
        "urn:SampleServer2-getCategories", false, null);
    // Загрузка преобразования
    var xsltCat = loadXML(serverBase +
        "xslt/getCategories.xsl");
    // Отображение списка категорий
    document.getElementById("selectPlaceholder").innerHTML =
        xsltTransform(msgGetCatResponse, xsltCat);
    // Регистрация новой сессии
    registerSession();
}
</script>
</head>
```

```
<body>
  <h1>Выбор книг в корзину</h1>
  <div id="sessId"></div>
  <div id="container">
    <form action="#" onsubmit="return false">
      <div>
        <label for="">Категория</label>
        <span id="selectPlaceholder"></span>
        <button onclick="showBooks()">Показать</button>
      </div>
    </form>
    <div id="basketPlaceholder"></div>
    <div id="tablePlaceholder"></div>
  </div>
</body>
</html>
```

## Лабораторная работа 6.1

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Аутентификация пользователя</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <style type="text/css">
    #divShadowScreen { display:none; position:absolute; top:0px;
left:0px; z-index:1; width:100%; height:100%; background-color:#000; opacity:
70%; filter:alpha(opacity=70); -moz-opacity: 0.7; }
    #divFrmLogin { display:none; width:25em; position:absolute;
top:50%; left:50%; font-family:Verdana,Sans-serif; font-size:small; z-
index:2; }
    #frmLogin { border:1px solid #000; background-color:#fff;
padding:1em; position:relative; top:-10em; left:-12em; }
    #frmLogin div { margin:1em; }
    #frmLogin div label { display:block; width:8em; float:left; }
    #divButtons { text-align:center; }
    #divButtons button { margin:1em; width:6em; }
    #divResult { margin:2em; }
  </style>
  <script type="text/javascript" src="md5.js"></script>
  <script type="text/javascript" src="json2.js"></script>
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript">
    /*
    ** Адрес сервера
    */
    var server = "server/server.php";

    /*
    ** Конструктор объекта Entropy
    */
    function Entropy()
    {
      this.salt = arguments.length > 0 ? arguments[0] : "";
      this.iterationCount = arguments.length > 1 ? arguments[1] :
"";
    }

    /*
    ** Конструктор объекта User
    */
    function User()
    {
      // Свойства объекта
      this.name = arguments.length > 0 ? arguments[0] : "";
      this.email = arguments.length > 1 ? arguments[1] : "";
      this.password = arguments.length > 2 ? arguments[2] : "";
      this.dbEntropy = new Entropy();
      this.transferEntropy = new Entropy();
    }

    /*
    ** "Засоливание" пароля RFC2898
    */
    function getSaltedHash(password, salt, iterationCount)

```



```

    {
        var saltedHash = password;
        if (iterationCount < 1) iterationCount = 1;
        for (var i=0; i<iterationCount; i++)
            saltedHash = hex_md5(salt + saltedHash);
        return saltedHash;
    }

    /*
    ** функция показывает/расит форму входа на сайт
    */
    function showHideLoginForm(visibility)
    {
        if (visibility)
        {
            document.getElementById("divShadowScreen").style.display = "block";
            document.getElementById("divFrmLogin").style.display =
"block";

            if (document.getElementById("txtEmail").value)
                document.getElementById("txtPassword").focus();
            else
                document.getElementById("txtEmail").focus();
        }
        else
        {
            document.getElementById("divShadowScreen").style.display = "";
            document.getElementById("divFrmLogin").style.display =
"";
        }
    }

    /* Задание 1. Запрос "соли" хэширования с сервера.
    **
    ** Общий сценарий:
    ** В этой лабораторной работе используется
    ** надежная передача пароля с помощью
    ** многократного хэширования пароля со случайной строкой.
    ** При формировании пароля в базе данных
    ** сохраняется "соль" - случайная строка и число итераций
    ** хэширования. Для передачи пароля между клиентом
    ** и сервером, хэш пароля в БД также "засоливается",
    ** то есть многократно хэшируется
    ** со случайной строкой. В этом задании вы передадите
    ** введенный Е-mail пользователя на сервер
    ** с помощью объекта User. Сервер вернет вам также объект User,
    ** у которого будут инициализированы
    ** свойства энтропии, то есть объекты "соли". Этих свойств два:
    ** та энтропия, которая использовалась
    ** при хэшировании пароля в БД, и новая энтропия, которая
    ** сгенерирована сервером для
    ** текущего сеанса связи. Вы должны сначала прохэшировать
    ** введенный пользователем пароль
    ** с "солью" user.dbEntropy.salt указанное число раз
    ** (user.dbEntropy.iterationCount) и
    ** полученную строку еще раз прохэшировать с "солью"
    ** user.transferEntropy.salt
    ** указанное число итераций (user.transferEntropy.iterationCount)
    ** Результирующую строку следует записать в свойство user.password
    ** и передать на сервер
    ** Если пользователь ввел правильный пароль,

```

```
** сервер вернет вам объект, у которого  
** установлено свойство user.name, иначе это свойство  
** будет пустым.  
** Такой способ передачи пароля позволяет говорить  
** о некоторой защищенности данных пользователя  
**  
** Допишите функцию loginUser(). После комментария  
** "// Создадим объект User" создайте новый  
** объект User, установив у него свойство email из  
** переменной email  
** Передайте асинхронным вызовом этот объект на сервер  
** (адрес сервера - глобальная переменная server)  
** сериализовав его в JSON строку. Получите данные от сервера  
** и десериализуйте его в объект  
** Вызовите функцию sendPassword, передав ей полученный объект  
** от сервера  
** В целях отладки выведите с помощью функции alert ответ сервера  
** Запишите, какие свойства у полученного объекта установлены.  
** Обратите внимание на значения объектов энтропии  
** Для проверки работы используйте E-mail пользователей:  
** vasyar@mail.ru, fedyas@mail.ru, masha@mail.ru  
** У всех пользователей пароль - password  
*/  
function loginUser()  
{  
    // Введенный пользователем E-mail  
    var email = document.getElementById("txtEmail").value;  
  
    // Создадим объект User  
    var user = new User("", email, "");  
  
    // Сериализуем его и отправим серверу  
    var jsonUser = JSON.stringify(user);  
    var req = getXmlHttpRequest();  
    req.onreadystatechange = function()  
    {  
        if (req.readyState != 4) return;  
        // Десериализация полученного объекта  
        var challenge = JSON.parse(req.responseText);  
        // Вызов функции хэширования пароля  
        sendPassword(challenge);  
    }  
    req.open("POST", server, true);  
    req.setRequestHeader("Content-Type", "text/plain");  
    req.setRequestHeader("Content-Length", jsonUser.length);  
  
    req.send(jsonUser);  
  
    // Гашение формы ввода пароля  
    showHideLoginForm(false);  
}  
  
/* Задание 2.  
** После комментария "// Хэширование" произведите хэширование  
** введенного пароля  
** (свойство user.password) с помощью функции getSaltedHash().  
** Передайте этой функции "соль" user.dbEntropy.salt и  
** число итераций user.dbEntropy.iterationCount  
** Сохраните результат в свойство user.password  
** Проведите второе хэширование, Передавая "соль"  
** user.transferEntropy.salt  
** и число итераций user.transferEntropy.iterationCount
```

```

** Сохраните результат в свойство user.password
** Сериализуйте объект user в JSON-строку и асинхронно
** передайте его на сервер
** (URL - 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 функции alert ответ сервера
** Запишите, какие свойства у полученного объекта установлены.
** Обратите внимание на значения свойства user.name
** Для проверки работы используйте E-mail пользователей:
** vasyar@mail.ru, fedyas@mail.ru, masha@mail.ru
** У всех пользователей пароль - password
*/
function sendPassword(user)
{
    // Введенный пользователем пароль
    user.password =
document.getElementById("txtPassword").value;
    // Сброс введенного пароля
    document.getElementById("txtPassword").value = "";

    // Хэширование
    // 1-е хэширование пароля - с энтропией БД
    user.password = getSaltedHash(user.password,
        user.dbEntropy.salt, user.dbEntropy.iterationCount);
    // 2-е хэширование пароля -
    // с энтропией используемой при передаче
    user.password = getSaltedHash(user.password,
        user.transferEntropy.salt,
        user.transferEntropy.iterationCount);

    // Сериализуем его и отправим серверу
    var jsonUser = JSON.stringify(user);
    var req = getXmlHttpRequest();
    req.onreadystatechange = function()
    {
        if (req.readyState != 4) return;
        // Десериализация полученного объекта
        var response = JSON.parse(req.responseText);
        // Вывод результатов
        showUserData(response);
    }
    req.open("POST", server, true);
    req.setRequestHeader("Content-Type", "text/plain");
    req.setRequestHeader("Content-Length", jsonUser.length);

    req.send(jsonUser);
}

/* Задание 3
** 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 объекте
** <div id="divResult"></div>
*/

```

```
function showUserData(user)
{
    // Объект вывода результатов
    var divResult = document.getElementById("divResult");

    // Вывод результатов
    if (user.name)
    {
        // Проверка пользователя прошла успешно
        divResult.style.color = "";
        divResult.style.fontWeight = "";
        divResult.innerHTML = "<strong>" + user.name +
            "</strong>! Добро пожаловать на сайт!";
    }
    else
    {
        divResult.style.color = "red";
        divResult.style.fontWeight = "bold";
        divResult.innerHTML =
            "Вы неправильно ввели логин/пароль!" +
            " Повторите снова...";
    }
}
</script>
</head>
<body>
    <h1>Аутентификация пользователя</h1>
    <form onsubmit="return false">
        <button onclick="showHideLoginForm(true)">Вход</button>
    </form>
    <div id="divFrmLogin">
        <form id="frmLogin" onsubmit="return false">
            <div>
                <label for="txtEmail">Ваш E-mail:</label>
                <input id="txtEmail" type="text" />
            </div>
            <div>
                <label for="txtPassword">Ваш пароль:</label>
                <input id="txtPassword" type="password" />
            </div>
            <div id="divButtons">
                <button onclick="loginUser()">Вход</button>
                <button
onclick="showHideLoginForm(false)">Закрыть</button>
            </div>
        </form>
    </div>
    <div id="divResult"></div>
    <div id="divShadowScreen"></div>
</body>
</html>
```

## Лабораторная работа 6.2

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru" dir="ltr">
<head>
  <title>Понедельник начинается в субботу</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Language" content="ru" />
  <link rel="stylesheet" type="text/css" href="lab6.2.css" />
  <script type="text/javascript" src="xmlhttprequest.js"></script>
  <script type="text/javascript" src="xslt.js"></script>
  <script type="text/javascript">
    // Адреса сервера
    var serverXml = "server/get-section-xml.php";
    var serverHtml = "server/get-section-html.php";

    // XSLT преобразование
    var fb2html;

    /* Задание 1. Функция загрузки главы книги
    ** Покажите сообщение пользователю о загрузке данных.
    ** Для этого установите у
    ** объекта divMessageLoad свойство display = "block"
    ** Сформируйте и выполните асинхронный GET запрос
    ** к серверу serverXml (глобальная переменная),
    ** передавая ему параметер no с номером текущей главы
    ** (аргумент функции)
    ** Получите XML данные и вызовите функцию showChapter,
    ** передавая ей параметры
    ** полученный DOM документ и номер текущей главы
    */
    function getChapter(no)
    {
      // Сообщение пользователю
      document.getElementById("divMessageLoad").style.display =
        "block";

      // Запрос главы с сервера
      var req = getXmlHttpRequest();
      req.onreadystatechange = function()
      {
        if (req.readyState != 4) return;
        //Получаем DOM документ
        var xmlDOM = req.responseXML;
        // Если нет ошибок...
        if (xmlDOM) showChapter(xmlDOM, no);
      }
      req.open("GET", serverXml + "?no=" + no , true);
      req.setRequestHeader("Content-Type", "text/xml");
      req.send(null);
    }

    /* 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(no). Выведите эти ссылки
    ** в объект divChapters

```

```

    ** Произведите XSLT преобразование полученной главы
    ** (переменная xmlDoc) с помощью загруженного
    ** преобразования fb2html. Преобразование выполните
    ** с помощью функции xsltTransform(xmlDoc, fb2html)
    ** Результат преобразования выведите в объект divResult
    ** Погасите сообщение пользователю о загрузке данных,
    ** устанавливая свойство display = "none"
    ** для объекта divMessageLoad
    */
function showChapter(xmlDoc, currentChapter)
{
    // Узнаем общее число глав
    var chapterCount =
xmlDoc.documentElement.firstChild.getAttribute("count");

    // Выведем ссылки вперед/назад
    var divChapters = document.getElementById("divChapters");
    var previousChapter = currentChapter - 1;
    var nextChapter = currentChapter + 1;

    divChapters.innerHTML = "";
    if (previousChapter > 0) divChapters.innerHTML +=
        '<a href="' + serverHtml + '?no=' + previousChapter +
        '" onclick="getChapter(' + previousChapter +
        ');return false">Предыдущая глава</a>&nbsp;';
    if (nextChapter < chapterCount) divChapters.innerHTML +=
        '&nbsp;<a href="' + serverHtml + '?no=' +
        nextChapter +
        '" onclick="getChapter(' + nextChapter +
        ');return false">Следующая глава</a>';

    // Произведем преобразование
    var html = xsltTransform(xmlDoc, fb2html);
    var divResult = document.getElementById("divResult");
    divResult.innerHTML = html;

    // Гашение сообщения пользователю
    document.getElementById("divMessageLoad").style.display =
        "";
}

/*
** Инициализация страницы
*/
window.onload = function()
{
    // Покажем сообщение пользователю
    var divMessageLoad =
document.getElementById("divMessageLoad");
    divMessageLoad.style.display = "block";

    // Загрузка XSLT преобразования
    var req = getXmlHttpRequest();
    req.open("GET", "server/fb2html.xsl", false);
    req.send(null);
    fb2html = req.responseXML;

    // Уберем сообщение пользователю
    divMessageLoad.style.display = "none";

    // Показ первой главы
    getChapter(1);
}
```

```
        }
    </script>
</head>
<body>
    <h1>Понедельник начинается в субботу</h1>
    <div id="divMessageLoad">Идет загрузка данных...</div>
    <noscript><a href="server/get-section-html.php?no=1">
        Перейти к чтению книги
    </a></noscript>
    <div id="divChapters"></div>
    <div id="divResult"></div>
</body>
</html>
```