**11. Asynchronous JavaScript and XML (AJAX)**

1. **Ajax** – методология построения интерактивного пользовательского интерфейса web-приложения.
2. **Методология** **Ajax описывает** способы разработки страниц сайта, которые могут динамически изменяться на основе данных с сервера, без перезагрузки страницы.
3. **В основе методологии Ajax лежат следующие технологии:**  HTML, JS, XML, модель DOM, протоколы HTTP, JSON, объект XMLHttpRequest
4. **HTML** **–** гипертекстовый язык разметки. Интерпретируется браузером. В Ajax динамически изменяется содержимое html-документа.
5. **JavaScript** – скриптовый язык, предназначенный для создания сценариев поведения браузера. Интерпретируется браузером. В Ajax html-документ динамически изменяется на стороне клиента с помощью сценариев написанных на языке JavaScript.
6. **DOM –** объектная модель, позволяющая сценариям JavaScript получить доступ (читать и изменять содержимое) к элементам html-документа (к атрибутам и содержимому тегов). В Ajax ответ сервера “встраивается” с помощью JavaScript-сценария в загруженную ранее браузером страницу. При этом доступ к элементам html-документа осуществляется а соответствии с моделью DOM.
7. **HTTP –** сетевой протокол передачи гипертекста. Используется для обмена данными между двумя приложениями (клиентом и сервером). В Ajax обмен данными между JavaScript-сценарием на клиенте и серверным приложением (например, сервлетом) осуществляется по правилам HTTP.
8. **XML –** расширяемый язык разметки данных. Предназначен для структуризации данных с целью хранения или/и передачи. В Ajax язык XML является одним из форматов, который используется для структуризации данных пересылаемых между JavaScript-сценарием и серверным приложением.
9. **JSON (JavaScript Object Notation)**  - текстовый формат обмена данными, применяемый обычно в JS-сценариях**.** В Ajax формат JSON является одним из форматов, который используется для структуризации данных пересылаемых между JS-сценарием и серверным приложением. Формат JSON основывается на функции **eval()** языка JavaScript.
10. **XMLHttpRequest** –специальный API (предопределенный объект), используемый в JS для обмена данными между JS-сценарием и серверным приложением по HTTP.

В Ajax методы объекта XMLHttpRequestиспользуется для отправки и получения данных между JS-сценарием и серверным приложением. Данные мб получены в виде текста, XML-документа и формате JSON.

**Задание 1.**

Разработайте сервлет **Sss\_Header**, который принимает два заголовка **Value-x** и **Value-y** (значения **x** **y**) и формирует заголовок ответа **Value-z** (значение **x+y**)

Разработайте форму для ввода значений **x**, **y** и **z**.

Поля **x** и **y**  упаковать в заголовки **Value-x** и **Value-y**, оправить http-запрос сервлету **Sss\_Header,** извлечь из ответа **Value-z** и поместить в тег **z**. (**index.html**)

**Задание 2.**

Разработайте сервлет **Sss\_XML**, который принимает заголовок **XRand-N** (значение int **n**), формирует **[5,10]** случайных чисел **(-n, n)**. Полученный ряд отправляет клиенту в виде xml-документа.

Разработайте форму. Значение **n** упаковать в заголовок **XRand-N** http-запроса, отправить сервлету **Sss\_XML,** принять ответ, разобрать полученный xml-документ в тело тега, предназначенного для отображения результата.

**Задание 3. Sss\_JSON**, аналогично, но ряд чисел отправить в формате JSON.

**Задание\_4. Применение асинхронной обработки.**

1. Установите в вызове **open** объекта **XMLHTTPRequest** третий параметр (асинхронность) в **false**.
2. Добавить задержку в сервлеты: Sss\_Header(10 сек), Sss\_XML (5 сек), Sss\_JSON (1 сек.)
3. Выполните приложение и объясните эффект.
4. Верните значение параметра в **open** в **true**.
5. Проверьте возможность асинхронной работы вашего приложения (не дожидаясь результата в первой форме, сразу выполнять вторую и третью). Объясните полученный результат.

**12. Технология JAAS**

* 1. ***JAAS (Java Authentication and Authorization Service)*** – Java-сервис по аутентификации и авторизации.
  2. Для хранения информации о ролях, именах пользователей и их паролях в сервере Apache Tomcat используется: ***tomcat-users.xml***
  3. Каждый пользователь характеризуется: именем, паролем и списком ролей, приписываемых ему.
  4. Защищенный ресурс (html –страница, jsp-страница, сервлет) привязывается к ролям. Все пользователи, содержащие в своем списке роль, обеспечивающую доступ (привязанную роль) к некоторому ресурсу, имеют возможность доступа к данному ресурсу.
  5. Технология JAAS предполагает две формы аутентификации пользователей: ***BASIC*** и ***FORM*** .
  6. BASIC-аутентификация подразумевает использование стандартной формы ввода данных для аутентификации (имени и пароля).
  7. FORM-аутентификация подразумевает использование собственной формы (разработанной пользователем, но по определенным правилам jsp-страницы) для аутентификации.
  8. Применение JAAS-технологии сводится к xml-описанию необходимой информации в дескрипторе приложения web.xml и к разработке форм для ввода имени и пароля пользователя и сообщения об ошибке (для FORM-аутентификации).

**Задание 1 (BASIC-аутетификация)**

1. index.jsp: вызов Sss с помощью тега ***<a href = … >*** .
2. сервлет Sss: выводит на консоль ***Servlet:SSS***.
3. ***tomcat-users.xml***: роль Role51 + user
4. web.xml: обесп. аутентифицированный доступ к ресурсу index.jsp
5. Измените в web.xml-файл таким образом, чтобы доступ к index.jsp был свободным, а доступ к сервлету Sss был аутентифицированным.
6. Измените в web.xml-файл таким образом, чтобы доступ к index.jsp и к сервлету Sss был аутентифицированным.

**Задание 2 (FORM-аутетификация)**

1. web.xml: FORM-аутентификация.
2. Разработайте форму для ввода имени и пароля пользователя.

стандартные значения для параметра ***action*** тега ***form***  и атрибутов ***name*** тегов полей ввода имени и пароля.

1. Разработайте форму для вывода сообщения об ошибке аутентификации.

**Задание 3 ( BASIC/FORM аутетификация)**

Измените web.xml, разработанный в предыдущих заданиях таким образом, чтобы использовалась BASIC-аутентификация для доступа к index.jsp и FORM-аутентификация для доступа к сервлету Sss.

***Задание 3 невыполнимо. Почему ?***

мы определяем два свойства secutiry-constraint для доступа к ресурсу

каждому должен соотв один login-config

**13. Загрузка файлов**

1. ***Загрузка файла*** – перемещение копии файла с компьютера сервера на компьютер клиента с использованием протокола HTTP.
2. Для загрузки файла необходимо иметь ***следующую информацию***: 1) имя директории с пересылаемым файлом; 2) имя пересылаемого файла; 3) тип пересылаемого файла и соответствующий тип ***mime*** (тип контента) протокола HTTP.
3. Загрузка файлов может быть выполнена сервлетом.
4. ***Директорию клиента***, в которую перемещается файл, определяет браузер с помощью дополнительного запроса.

**Задание**

1. **web.xml**: задать параметр инициации docs с именем директория (files)
2. **Sss**: получение параметра инициации
3. Создайте папку *files*, создать несколько Word-файлов.
4. **index.jsp**: вывод имен всех Word-файлов хранящихся в папке files в виде ссылок (<a>) на сервлет Sss.
5. **Sss**: пересылал Word-файл, выбранный из списка на jsp-странице (показать заголовки).
6. Зарегистрируйте в журнале сервера имена пересылаемых файлов.
7. Убедитесь, что файл пересылается на клиентский компьютер, а в журнале регистрации осуществлена, соответствующая запись.