Знакомство с технологией Java Server Pages

Методические указания к лабораторной работе

Цель работы: ознакомиться и получить навыки создания простого веб-приложения, подключающегося к серверу баз данных. Ознакомиться с основными идеями и технологиями веб-разработки, такими как JavaServer Pages (JSP), JavaServer Pages Standard Tag Library (JSTL), JDBC, а также трехуровневой клиент-серверной архитектуры.

1. Теоретическая справка

1.1. JavaServer Pages

JSP (JavaServer Pages) — технология, позволяющая веб-разработчикам легко создавать содержимое, которое имеет как статические, так и динамические компоненты. По сути, страница JSP является текстовым документом, который содержит текст двух типов:

- статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML, и
- JSP элементы, которые конструируют динамическое содержимое.

Кроме этого могут использоваться библиотеки JSP тегов, а также EL (Expression Language), для внедрения Java-кода в статичное содержимое JSP-страниц.

JSP — одна из высокопроизводительных технологий, так как весь код страницы транслируется в java-код **сервлета** с помощью компилятора JSP страниц Jasper, и затем компилируется в байт-код виртуальной машины java (JVM).

Контейнеры сервлетов, способные исполнять JSP страницы, написаны на платформонезависимом языке Java, который может работать на различных платформах. JSP страницы загружаются на сервере и управляются из структуры специального Java server packet, который называется Java EE Web Application, в большинстве своём упакованная в ***файловые архивы .war и .ear.***

Выгода, которую дает технология JSP в сравнении с другими веб-технологиями заключается в том, что JSP является платформонезависимой, переносимой и легко расширяемой технологией для разработки веб-приложений.

1.2. Планирование структуры приложения

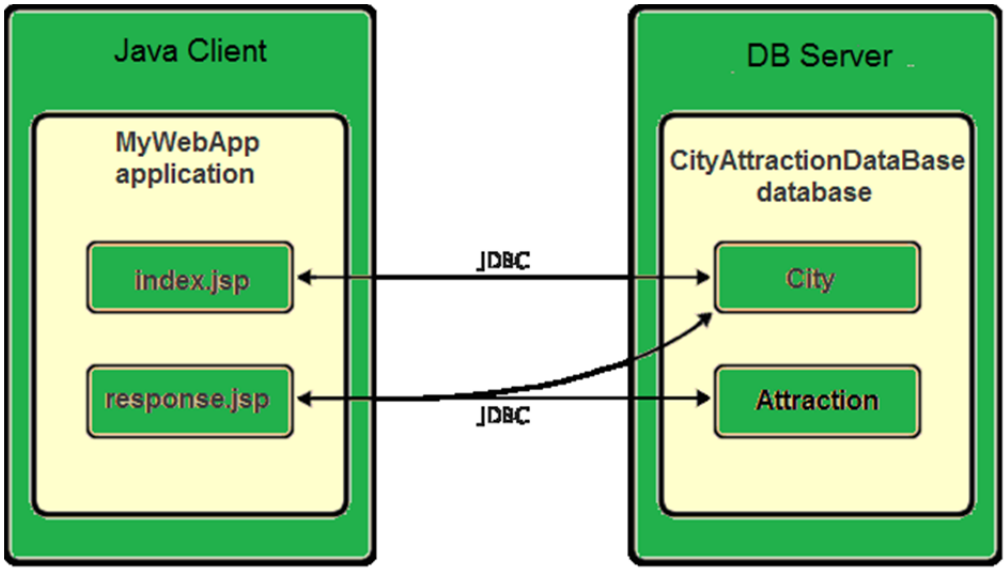
Простые веб-приложения могут быть разработаны с использованием двухуровневой архитектуры, в которой клиенты взаимодействуют с БД напрямую с сервера. Приложение взаимодействует непосредственно с базой данных (например, платформы MySQL) с использованием JDBC (Java Database Connectivity) API. По сути, это драйвер, обеспечивающий связь между кодом Java и сервером приложений (в качестве которого выступает сервер ***GlassFish***), а также SQL контентом, понятном серверу баз данных.

Рассмотрим пример приложения, демонстрирующего достопримечательности городов.

Пусть создаваемое приложение предполагает создание двух страниц JSP: index.jsp и response.jsp.

В каждой из этих страниц:

- используется HTML для реализации интерфейса;
- применяются технологии JSTL для динамического извлечения и отображения данных на каждой странице.



В базе данных, назовем ее CityAttractionDatabase, необходимо создать две таблицы: City и Attraction.

Страница приветствия (index.jsp) предоставляет пользователю простую HTML-форму. Когда браузер запрашивает index.jsp, JSTL код на странице инициирует запрос к CityAttractionDatabase. Он извлекает данные из таблицы City, и вставляет его в страницу перед отправкой в браузер. Когда пользователь отправляет свой выбор в HTML форме страницы приветствия, инициируется запрос к странице ответа (response.jsp). Вновь JSTL код инициирует запрос к CityAttractionDatabase. На этот раз, он извлекает данные из таблицы City и Attraction и вставляет его в страницу, что позволяет пользователю просматривать данные, основанные на его выборе, когда страница возвращается в браузер.

1.3. Создание нового проекта

Для создания и разработки проекта приложения необходимо войти в среду NetBeans.

1) Выберите File -> New Project в главном меню. Выберите категорию Java Web, а затем выберите Web Application. Нажмите кнопку Далее.

2) В панели Server and Settings, укажите сервер GlassFish, как сервер, который будет использоваться для запуска приложения.

3) В поле Java EE Version, выберите Java EE 5.

4) Нажмите кнопку Готово. В результате создается шаблон проекта для всего приложения, и открывается пустая страница JSP (index.jsp) в редакторе. Файл index.jsp используется в качестве начальной страницы приложения.

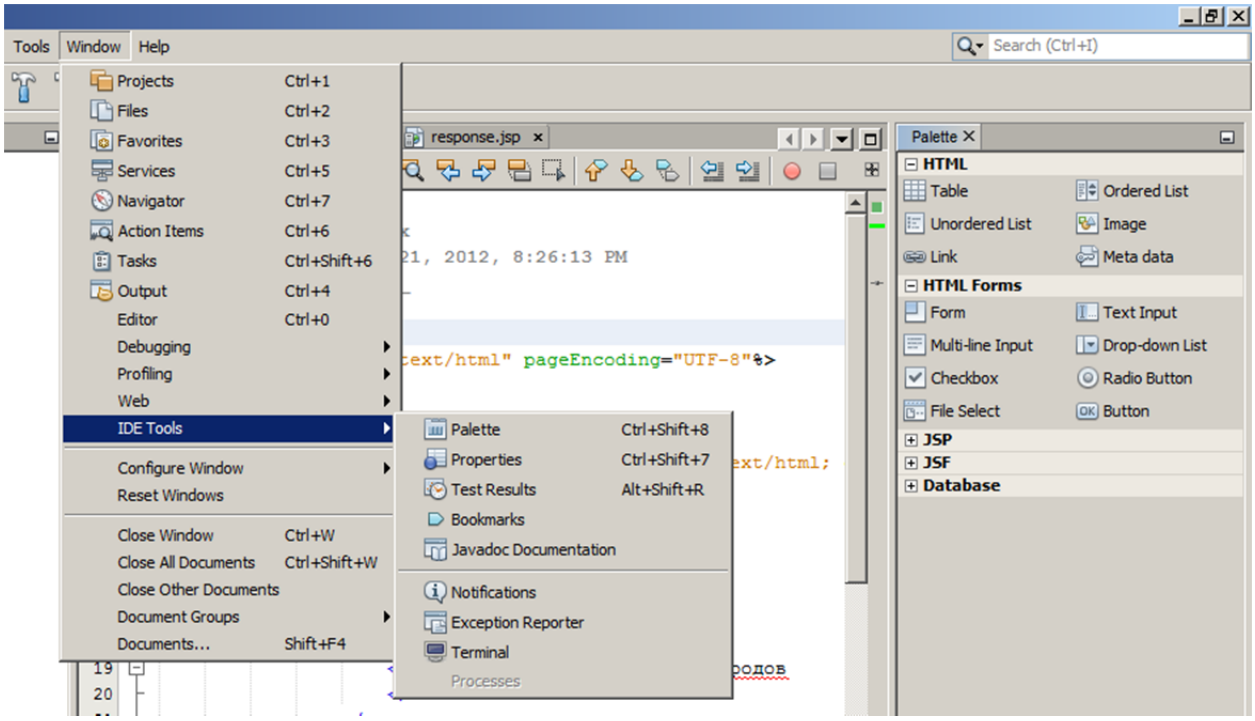
1.4. Подготовка веб-интерфейса

1.4.1 Настройка страницы приветствия

1) В index.jsp, измените текст между тегами <*title*>: MyWebApp.

2) Измените текст между тегами <*h1*>: Welcome to MyWebApp!

3) Откройте палитры, выбрав Window → IDE Tools → Palette (Ctrl-Shift-8) из главного меню.

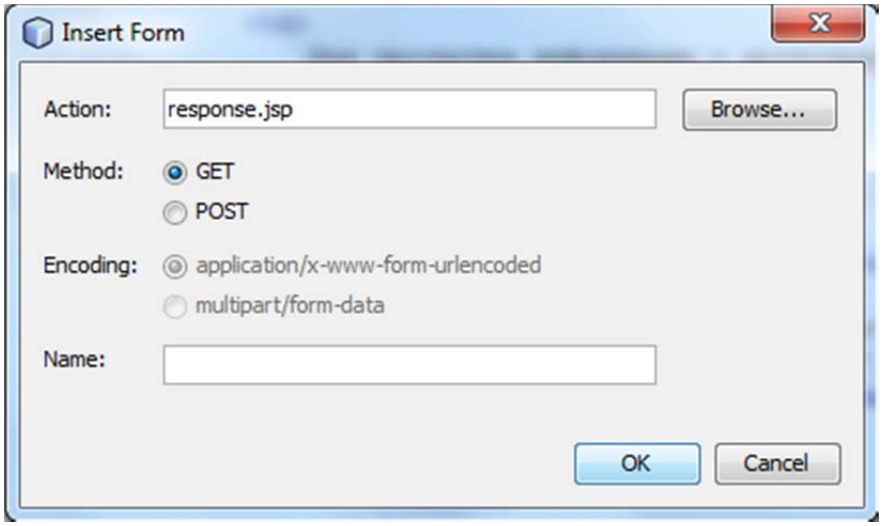


4) Добавьте таблицу из Palette после тега <*h1*> (строки: 2; колонки: 1; размер границы: 0)

5) Добавьте следующее содержимое в заголовок таблицы и ячейку первой строки таблицы:

```
<table border="0">
        <thead>
          <tr>
            <th>
               Cities Attractions
            </th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>
              Для просмотра достопримечательностей выберите  город
            </td>
          </tr>
         ...
</table>
```

6) В нижнюю строку таблицы добавьте HTML Form из Palette



7) Введите следующие содержимого между <form:

```
<td>
   <form action="response.jsp">
      <strong> City:</strong>
    </form>
</td>
```

8) После ** добавьте элемент Drop-down List (Name - city), после которого добавьте элемент Button(Label-OK).

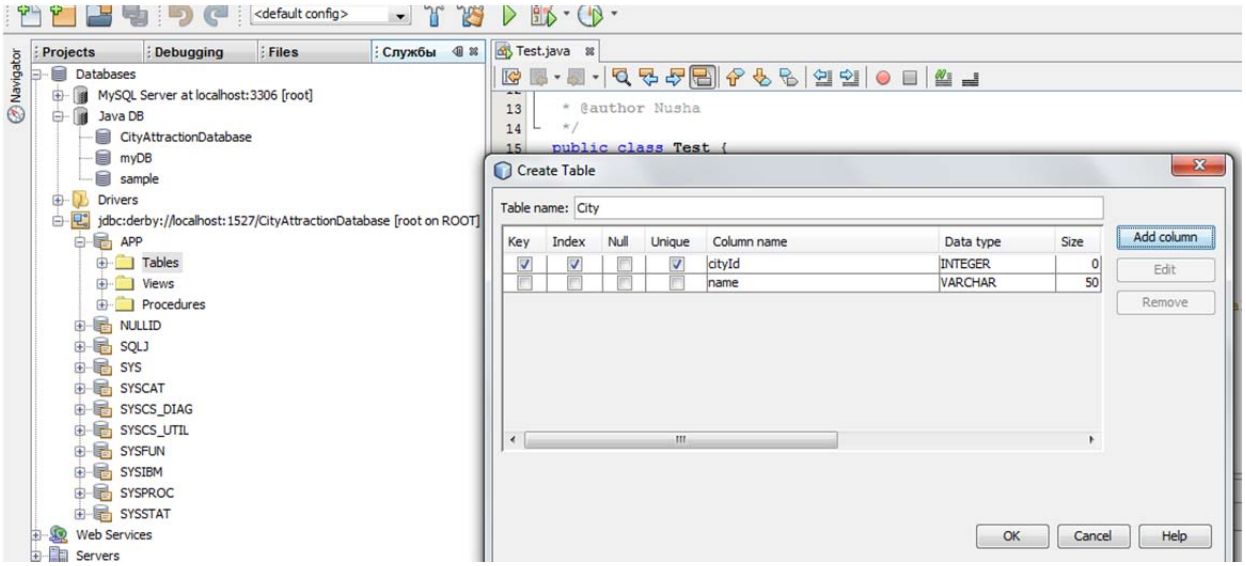
1.4.2 Настройка страницы ответа

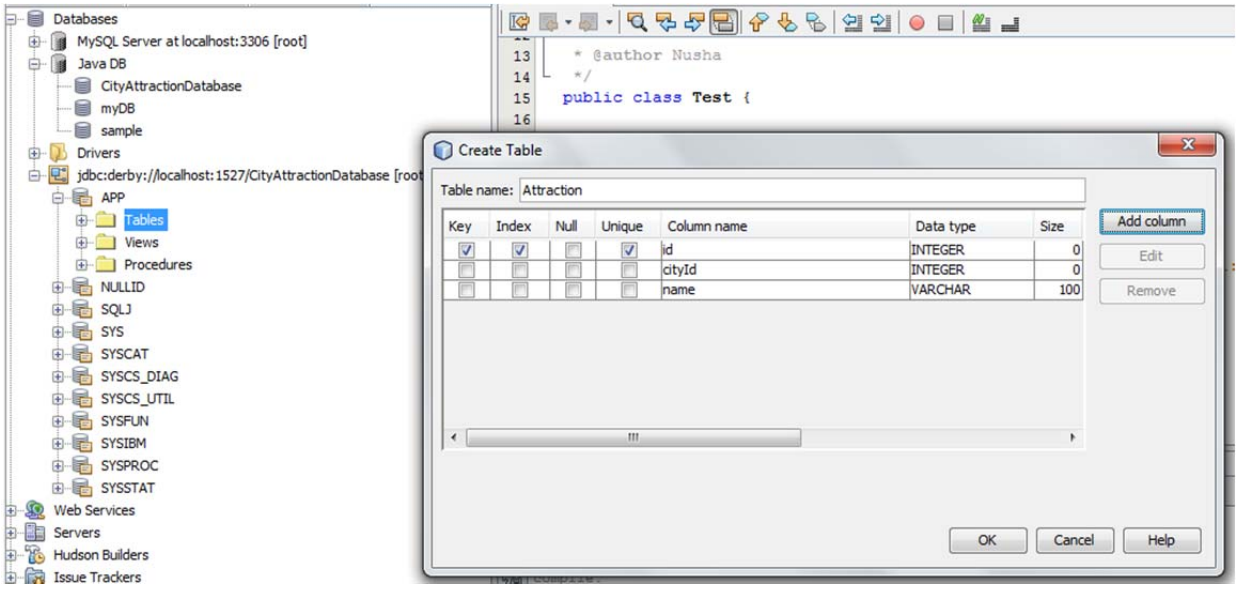
Создайте новый jsp-файл response.jsp в той же папке что и index.jsp.

1.5 Подключение к базе данных

NetBeans поставляется в комплекте с поддержкой СУБД Derby (сервер Java DB).

В окне Services разверните узел Databases, разверните Java DB и создайте базу данных CityAttractionDataBase. Убедившись, что сервер баз данных работает на Вашем компьютере, щелкните в узле Java DB правой кнопкой мыши по CityAttractionDataBase и выберите Connect. В CityAttractionDataBase в схеме APP создайте таблицы City и Attraction и заполните их данными:





1.6 Реализация механизма взаимодействия между сервером и базой данных

Наиболее эффективным способом реализации механизма взаимодействия между сервером и базой данных является создание *пула соединений с базой данных*. Создание нового подключения для каждого запроса клиента может потребовать много времени, особенно для приложений, которые постоянно выполняют большое количество запросов. Чтобы исправить это, многочисленные связи создаются и поддерживаются в пуле соединений. Любые входящие запросы, которым требуется доступ к данным, используют уже созданное подключение из пула.

После подготовки источника данных и пула подключений для сервера, нужно указать приложению источник данных. Обычно это делается путем создания записи в web.xml.

Далее необходимо убедиться, что драйвер базы данных является доступным для сервера.

1.6.1 Настройка источников данных JDBC и пула соединений

Сервер GlassFish содержит Database Connection Pooling (DBCP), предоставляющий функциональность пула соединений. Для этого, необходимо настроить JDBC (Java Database Connectivity) источник данных для сервера.

Вы можете настроить источник данных непосредственно в консоли администратора сервера GlassFish.

1) Откройте мастер создания файла, нажав New File на главной панели инструментов среды IDE. Выберите категорию GlassFish сервер, а затем выберите JDBC Resource и нажмите Далее.

2) В General Attributes, выберите опцию Create New JDBC Connection Pool, а затем установите JNDI Name.

3) Перейдите к шагу Choose Database Connection и укажите DBC Connection Pool Name. Выберите, созданное вами соединение из раскрывающегося списка (например, jdbc:derby://localhost:1527/cityattractiondatabase), и нажмите Далее.

4) В итоге создастся файл sun-resources.xml (или glassfish-resources.xml).

1.6.2 Ссылка на источник данных из приложения

Для того, чтобы ссылаться на ресурс JDBC, который Вы только что настроили, необходимо создать запись в дескрипторе развертывания web.xml:

1) В окне Projects раскройте папку файлов конфигурации (Configuration Files) и дважды щелкните web.xml.

2) Щелкните вкладку References и нажмите Add.

3) Введите имя ресурса, которое вы дали при настройке источника данных для сервера выше (jdbc/myDatasource). Нажмите OK.

1.7 Добавление Dynamic Logic

для доступа и отображения данных, взятых из базы данных, можно применять стандартные классы библиотеки JavaServer Pages Tag Library (JSTL). Сервер GlassFish JSTL включает библиотеку по умолчанию (ее файл: jstl-impl.jar).

JSTL— в переводе с английского «стандартная библиотека тегов JSP». Она расширяет спецификацию JSP, добавляя библиотеку JSP тегов для общих нужд, таких как разбор XML данных, условная обработка, создание циклов и поддержка интернационализации.

JSTL является альтернативой такому виду встроенной в JSP логики, как скриптлеты, т.е. прямые вставки Java кода. Использование стандартизованного множества тегов предпочтительнее, поскольку получаемый код легче поддерживать и проще отделять бизнес-логику от логики отображения.

JSTL предоставляет четыре основных функциональных областей:

- **core**: общие, структурные задачи, такие как итераторы и условные операторы для обработки управления потоком;

- fmt: поддержка интернационализации;

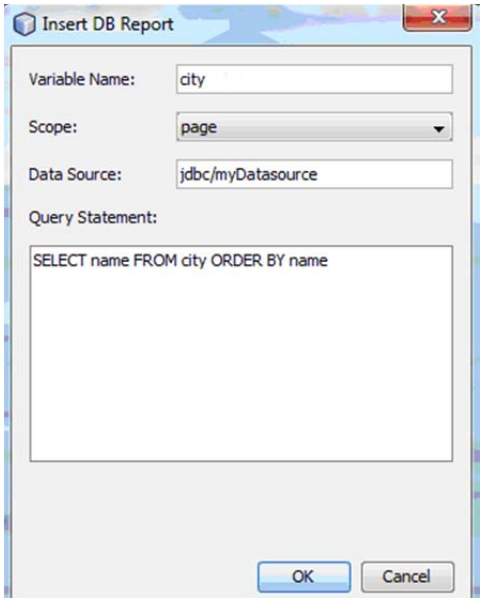
- sql: простой доступ к базе данных;

- xml: обработка XML.

1.7.1 Настройка страницы приветствия

Для того, чтобы динамически отображать содержимое формы в index.jsp, необходимо получить доступ к таблице City.

1) Наведите курсор перед *<%@page ... %>* и добавьте DB Report информацию, показанную на рисунке:



2) Запустите проект, чтобы протестировать соединение с базой данных

3) Удалите текст выделенный зачеркнутым шрифтом:

```
<sql:query var="city" dataSource="jdbc/myDatasource">
  SELECT name FROM city ORDER BY name
</sql:query>
<table border="1">
  <!-- column headers -->
  <tr>
  <c:forEach var="columnName" items="${city.columnNames}">
    <th><c:out value="${columnName}"/></th>
  </c:forEach>
</tr>
<!-- column data -->
<c:forEach var="row" items="${city.rowsByIndex}">
  <tr>
  <c:forEach var="column" items="${row}">
    <td><c:out value="${column}"/></td>
```

~~</c:forEach>~~

~~</tr>~~

~~</c:forEach>~~

~~</table>~~

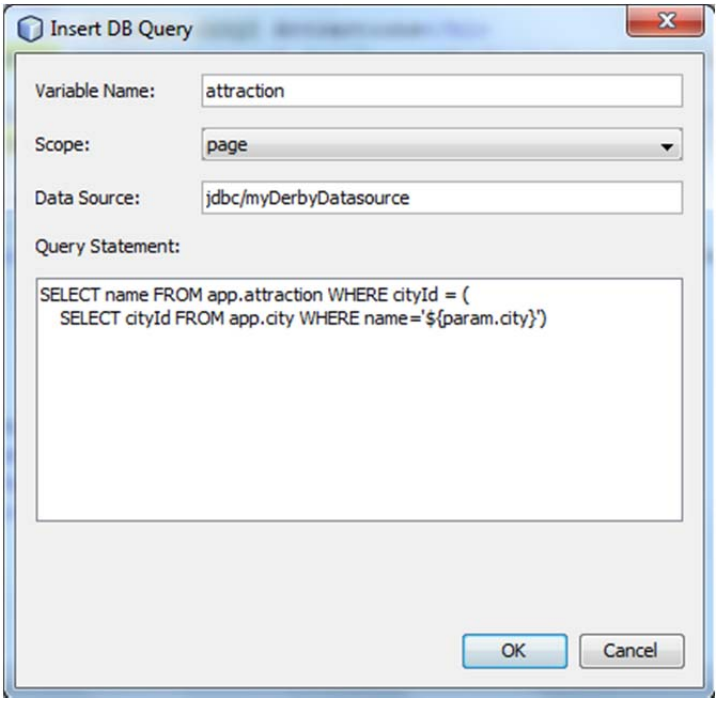
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
...
```

4) Замените код между *<select>:*

```
<select name="city">
        <c:forEach var="row" items="${city.rowsByIndex}">
            <c:forEach var="column" items="${row}">
                <option value='<c:out value="${column}"/>'>
                    <c:out value="${column}"/>
                </option>
            </c:forEach>
        </c:forEach>
</select>
```

5) Запустите проект. Система NetBeans инициирует запуск приложения в Web – интерфейсе, показанном на рисунке:

5) Для отображения достопримечательностей выбранного города в файле response.jsp добавьте генерацию отчета (DB Report)



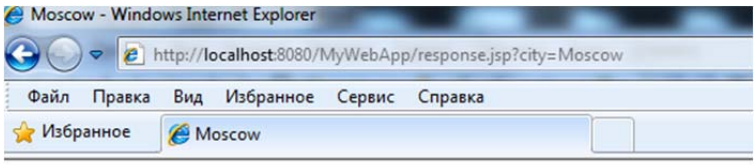
6) Удалите текст выделенный зачеркнутым шрифтом:

```
<table border="0">
    <!-- column headers -->
    <tr>
    <c:forEach var="columnName" items="${attraction.columnNames}">
        <th><c:out value="${columnName}"/></th>
    </c:forEach>
  </tr>
  <!-- column data -->
  <c:forEach var="row" items="${attraction.rowsByIndex}">
    <tr>
    <c:forEach var="column" items="${row}">
      <td><c:out value="${column}"/></td>
    </c:forEach>
    </tr>
  </c:forEach>
</table>
```

7) Замените код заголовков:

```
<head>
    <title>${param.city}</title>
  </head>
  <body>
    <h1>${param.city} Attractions</h1>
```

8) Запустите проект и выберите город после чего получите страницу ответа, например, как показанную на рисунке:



Moscow Attractions

Kremlin
Red Square
Poklonnaja mountain
St. Basil's Cathedral
The Cathedral of Christ the Savoir
Novodevichy Convent
Mausoleum
Bolshoy Theater
State Tretyakov Gallery

Требуемое ПО:

- NetBeans IDE 6.8, 6.9 или 7.0
- Java Development Kit (JDK) версии 6
- Сервер баз данных (Java DB)
- Сервер приложений GlassFish Server Open Source 3.x

Порядок выполнения работы

1. С использованием технологии Java Server Pages реализовать рассмотренный пример клиент-серверного приложения.

2. Реализовать собственный пример клиент-серверного web – приложения? используя подключение к базе данных SAP-Sybase.

Контрольные вопросы

1. Что такое сервлет?
2. Что такое контейнер?
3. Для чего нужен пул соединений?
4. Чем отличается установка от размещения приложения на сервере приложений?
5. Чем отличается сервер приложений от Web – сервера?
6. Сравните СУБД Derby с известными СУБД MS SQL, Sybase, Oracle.

7. Каким образом реализуется многопользовательский режим доступа к Web – приложениям, работающим с базами данных?
8. Какие серверы применяются в трехслойной архитектуре "клиент-сервер", работающей в Интернет?
9. Какой вид программирования применяется в данной работе?