**Лабораторная работа №7**

# ОСНОВНЫЕ КОНЦЕПЦИИ ТЕХНОЛОГИИ JAVASERVER PAGES

Для организации веб-взаимодействия клиента с сервером в платформе Java Enterprise Edition (Java EE) используются две базовые технологии – Java Servlet (сервлеты) и JavaServer Pages (JSP).

Сервлеты позволяют получать запросы от клиента, выполнять обработку информации и выводить результаты клиенту. В сервлете можно реализовать достаточно сложную логику, организовывать взаимодействие с базами данных и многое другое.

JavaServer Pages – технология, позволяющая разработчикам динамически генерировать XML, HTML и другие веб-страницы.

Страницы JSP позволяет легко разделить веб-содержимое на статическую и динамическую части, допускающую многократное использование ранее определенных компонентов. Технология позволяет внедрять Java-код, а также EL (Expression Language) в статичное содержимое страниц. Разработчики JavaServer Pages могут использовать компоненты JavaBeans и создавать собственные библиотеки нестандартных тегов, которые инкапсулируют сложные динамические функциональные средства.

Спецификация JavaServer Pages наследует и расширяет спецификацию сервлетов. Как и сервлеты, компоненты JSP развертываются в веб-контейнере и управляются им. При первом обращении страница JSP компилируется в сервлет. Специальная часть веб-сервера, называемая JSP Engine (среда исполнения JSP) выполняет анализ и разбор страницы. Затем JSP Engine, на основании статической и динамической части страницы, генерирует код сервлета, компилирует его и разворачивает в контексте веб-сервера. Далее все запросы к странице JSP обрабатываются этим сервлетом.

Жизненный цикл страницы JSP состоит из следующих этапов:

* трансляция;
* компиляция;
* загрузка и конкретизация класса;
* вызов метода jspInit();
* вызов метода \_jspService();
* вызов метода jspDestroy().

**Трансляция:** страница JSP прочитывается, синтаксически анализируется, проверяется. В случае отсутствия ошибки, создается java-файл, содержащий java-класс сервлета.

**Компиляция**: java-файл, созданный во время фазы трансляции, компилируется в class-файл. На данном этапе код Java проверяется на корректность, в случае обнаружения ошибки появляется необходимое сообщение.

**Загрузка и конкретизация класса**: в случае успешной компиляции, класс сервлета загружается в память и конкретизируется.

**Метод jspInit()** запрашивается только один раз за жизненный цикл и используется для того, чтобы задать все начальные условия, необходимые для данного сервлета.

**Метод \_jspService()** соответствует телу JSP-страницы, получает объекты запроса и ответа. В этом методе обрабатываются скриптлеты и выражения JSP.

**Метод jspDestroy()** запрашивается, когда экземпляр сервлета страницы JSP необходимо удалить. Любые операции по очистке, такие как освобождение ресурсов, могут производиться в рамках этого метода.

JSP страницы имеют расширение \*.jsp. Обращение к JSP производится так же, как и к обычной странице HTML, например, http://localhost:8080/HelloWorldWeb/helloWorld.jsp

Синтаксис JSP, определяющий правила записи, состоит из стандартных тегов HTML и тегов JSP. Набор стандартных тегов JSP довольно прост и включает в себя следующие категории:

* directives (директивы);
* declarations (объявления);
* scriptlets (скриптлеты);
* expressions (выражения);
* comments (комментарии).

**Директивы** являются сообщениями для JSP-контейнера, обеспечивают глобальную информацию, касающуюся конкретных запросов, и предоставляют сведения, необходимые на стадии трансляции.

Синтаксис директивы JSP:

|  |
| --- |
| <%@ директива имяАтрибута="значение" %> |

Эквивалентный XML синтаксис:

|  |
| --- |
| <jsp:directive.директива имяАтрибута="значение" /> |

Существует три типа директив:

* page (страница) ;
* taglib (библиотека тегов);
* include (включить).

Директива **page** определяет общие свойства страницы JSP, такие как кодировка, создание сеанса, работа с буфером, обработка ошибок и др. Порядок следования атрибутов в директиве page не имеет значения, но нарушение синтаксиса или наличие нераспознанных атрибутов приводит к ошибке трансляции. Возможные атрибуты директивы page приведены в следующей таблице:

|  |  |
| --- | --- |
| language | Определяет язык, используемый в скриптлетах JSP, выражениях или любых включаемых файлах. По умолчанию принимается значение "java". |
| extends | Задает суперкласс для генерируемого сервлета. |
| import | Определяет импортируемые пакеты. |
| session | Указывает на возможность работы со встроенным объектом session (тип HttpSession), который привязан к существующей сессии, если таковая имеется. В противном случае создается новая сессия, к которой осуществляется привязка. |
| buffer | Задает размер буфера для JspWriter out. |
| autoFlush | Определяет, должен ли буфер освобождаться автоматически при переполнении или возникновении ошибки. |
| info | Определяет строку информации о странице JSP. |

|  |  |
| --- | --- |
| isThreadSafe | Определяет режим выполнения сервлета, когда множественные запросы обрабатываются одновременно с использованием одного экземпляра сервлета, в случае, что автор синхронизировал доступ к его переменным. При значении false сервлет должен наследовать однопоточную модельSingleThreadModel, при которой последовательные или одновременные запросы обрабатываются отдельными экземплярами сервлета. |
| errorPage | Определяет URL страницы, которая должна выводиться в случае возможных ситуаций, вызывающих исключения. |
| isErrorPage | Сигнализирует о том, используется ли эта страница для обработки ошибок для других JSP страниц. |
| contentType | Определяет кодировку для страницы JSP и ответа, а также MIME-тип ответа JSP.  Значение по умолчанию типа содержания – text/html, кодировки – ISO-8859-1*.* |
| pageEncoding | Определяет кодировку символов страницы JSP. По умолчанию используется значение charset из атрибута contentType, если оно там определено. В противном случае устанавливается равным ISO-8859-1. |

Директива **taglib** позволяет использовать на страницах JSP дополнительные теги, созданные разработчиком приложения (custom tags). Если контейнер не может найти библиотеку тегов, возникает ошибка трансляции.

Синтаксис директивы taglib:

|  |
| --- |
| <%@ taglib uri="URI библиотеки тегов" prefix="имяПрефикса"%> |

Директива **include** используется для помещения в JSP страницу текста и программного кода из других источников и ресурсов. Подстановка выполняется в момент трансляции страницы JSP в соответствующий сервлет.

Синтаксис директивы include:

|  |
| --- |
| <%@ include file="Относительный URI включаемого файла" %> |

Директива include имеет один атрибут file, который включает в себя текст специфицированного ресурса в файл JSP. Рассматриваемый источник, например, страница HTML, интерпретируется в качестве статического объекта.

Содержимое подключаемого файла обрабатывается как обычный текст JSP и может включать такие элементы, как статический HTML, элементы скриптов, директивы и действия. Возможности данной директивы можно использовать для размещения на каждой странице JSP неизменяемой информации, например, сведения об авторских правах.

**Объявления (declarations)** предназначены для определения переменных и методов на языке скриптов, которые в дальнейшем будут использоваться на странице JSP.

Синтаксис объявления:

|  |
| --- |
| <%! объявление %> |

Эквивалентный XML синтаксис:

|  |
| --- |
| <jsp:declaration> объявление </jsp:declaration> |

Объявление может содержать несколько строк, как например, в приведенном ниже методе summ(int a, int b)для вычисления суммы двух целых чисел.

|  |
| --- |
| <%!  public int summ (int a, int b){  return a+b;  }  %> |

Объявления не производят никакого вывода в стандартный выходной поток out. Переменные и методы, декларированные в объявлениях, инициализируются и становятся доступными для скриптлетов и других объявлений в момент инициализации страницы JSP.

**Скриптлеты (scriptlets)** состоят из фрагментов Java-кода и символов разметки скриптовых элементов. Скриптлет может содержать программный код и декларации локальных переменных, которые будут использованы для обработки запросов клиентов.

Синтаксис скриптлета:

|  |
| --- |
| <% скриптлет %> |

Эквивалентный XML синтаксис:

|  |
| --- |
| <jsp:scriptlet> скриптлет </jsp:scriptlet> |

Скриптлеты имеют доступ к тем же автоматически определенным переменным, что и выражения.

**Выражения (expressions)** –это исполняемые конструкции, написанные на языке Java, предназначенные для конвертации данных в выходящий поток в виде строковых значений результатов выполняемых методов.

Синтаксис выражения:

|  |
| --- |
| <%= выражение %> |

Эквивалентный XML синтаксис:

|  |
| --- |
| <jsp:expression> выражение </jsp:expression> |

Результат выражения JSP, имеющий обязательный тип String, направляется в стандартный поток вывода out с помощью текущего объекта JspWriter. Если результат выражения не может быть приведен к типу String, возникает либо ошибка трансляции, либо исключение ClassCastException, если несоответствие было выявлено в процессе выполнения запроса.

**Комментарии (comments)** – пояснения к исходному тексту программы, находящиеся непосредственно внутри кода. С точки зрения компилятора или интерпретатора, комментарии – часть текста программы, не влияющая на ее семантику.

В странице JSP могут быть использованы два типа комментариев:

|  |
| --- |
| <%-- JSP-комментарий --%>  <!-- HTML-комментарий --> |

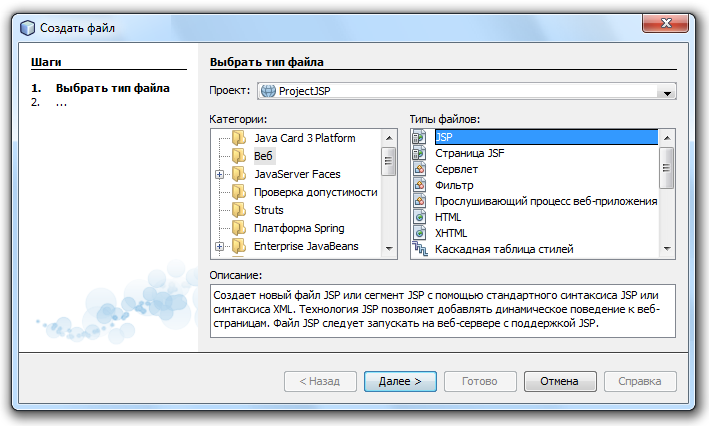
Также можно создавать комментарии с динамическим содержимым:

|  |
| --- |
| <!-- Начало комментария <%= expression %> конец --> |

***Лабораторная работа***

***Цель работы:*** описание процессасоздания простой страницы JSP в инструментальной среде разработки NetBeans IDE. Работа с директивами и стандартными тегами JSP.

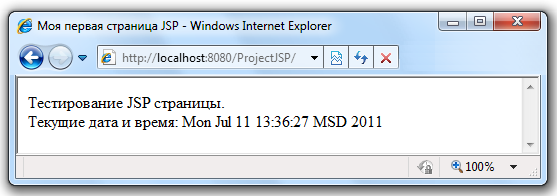
1. Создайте новый проект в среде разработки **NetBeans IDE**:
2. откройте среду разработки **NetBeans IDE**;
3. выполните команду **Файл → Создать проект…**;
4. выберите в списке **Категории** пункт **Java** **Web**;
5. выберите в списке **Проекты** пункт **Веб-приложение**;
6. нажмите на кнопку **Далее** и перейдите к следующему шагу;
7. введите имя проекта (например, **ProjectJSP)** в строку **Имя проекта**;
8. при помощи кнопки **Обзор** определите место для сохранения проекта;
9. нажмите на кнопку **Далее** и перейдите к следующему шагу;
10. выберите из списка **Сервер** необходимое название сервера;
11. нажмите на кнопку **Готово** для завершения создания проекта.
12. Создайте новую страницу JSP:
13. в созданном проекте выполните команду **Файл → Создать файл**;
14. выберите в списке **Категории** пункт **Веб**;
15. выберите в списке **Типы файлов** пункт **JSP** и перейдите к следующему шагу;



1. введите имя **firstJSP** в строку **Имя класса**;
2. в поле **Параметры** выберите **Файл JSP (стандартный синтаксис)**;
3. нажмите на кнопку **Готово** для завершения создания страницы.
4. Сформируйте код страницы **index.jsp**:

|  |
| --- |
| <%@ page contentType="text/html" pageEncoding="UTF-8"%>  <%@ page import="java.util.Date" %>  <html>  <head>  <meta http-equiv="Content-Type"  content="text/html; charset=UTF-8">  <title> Моя первая страница JSP </title>  </head>  <body>  Тестирование JSP страницы. <br>  Текущие дата и время: <%= new Date()%>  </body>  </html> |

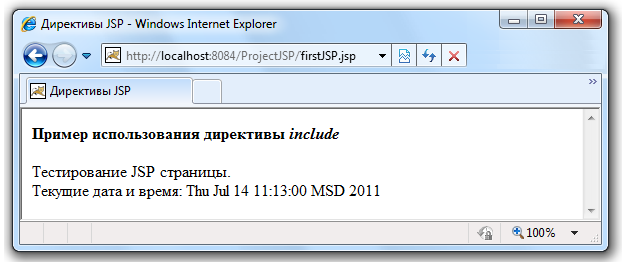
1. Запустите проект на выполнение и просмотрите результат работы:



1. Сформируйте код страницы **firstJSP.jsp**:

|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%>  <html>  <head>  <meta http-equiv="Content-Type"  content="text/html; charset=UTF-8">  <title> Директивы JSP </title>  </head>  <body>  <h4> Пример использования директивы <i>include</i></h4>  <%@ include file="index.jsp" %>  </body>  </html> |

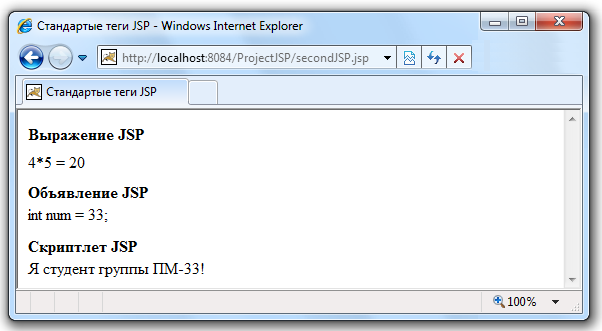
1. Запустите страницу **firstJSP.jsp** на выполнение:



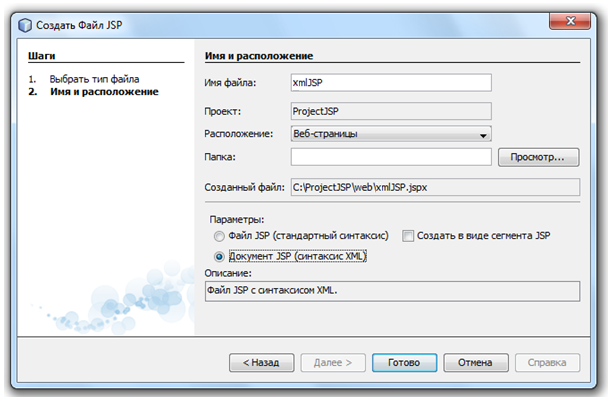
1. Создайте страницу **secondJSP.jsp**:

|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%>  <html>  <head>  <meta http-equiv="Content-Type"  content="text/html; charset=UTF-8">  <title> Стандартые теги JSP </title>  </head>  <body>  <h4> Выражение JSP </h4>  4\*5 = <%= 4\*5 %>  <h4> Объявление JSP </h4>  <%! int num = 33; %> int num = 33;  <h4> Скриптлет JSP </h4>  <%out.println("Я студент группы ПМ-" + num + "!");%>  </body>  </html> |

1. Запустите страницу **secondJSP.jsp** на выполнение и просмотрите результат работы в браузере.



1. Ознакомьтесь с основными фазами жизненного цикла страницы JSP.
2. Создайте страницу **xmlJSP.jspx**, выбрав опцию **Документ** **JSP (синтаксис XML).**



1. Рассмотрите сформированный программный код в формате **XML**.
2. Сформируйте программный код блока <jsp:body>…</jsp:body>:

|  |
| --- |
| <!--объявление переменной для подсчета количества подключений-->  <jsp:declaration>  private int accessCount = 0;  </jsp:declaration>  <!-- вывод значения количества подключений -->  <h4> Количество подключений =  <jsp:expression> ++accessCount </jsp:expression>  </h4> |

1. Запустите страницу **xmlJSP.jspx** на выполнение и просмотрите результат работы в браузере:

