ТЕМА 8
Создание веб-приложений на основе JSP

Цель лабораторной работы

Научиться создавать веб-приложения на основе JSP, библиотеки JSTL и самостоятельно разработанных тегов для платформы Java Enterprise Edition.

1 Разработка веб-приложений на основе JSP

Страница JSP – это веб-страница, которая вместе с HTML-тегами содержит исполняемые элементы. Как и любая другая веб-страница, JSP имеет уникальный адрес URL, используемый клиентом для доступа к ней. Особенностью JSP является то, что перед обработкой ответа на запрос пользователя, страница трансформируется в сервлет и комилируется.

Программная логика JSP может быть представлена следующими классами элементов:

- скриптовые элементы;
- директивы;
- действия (теги).

Прочее содержание JSP, не являющееся JSP-кодом, является шаблонной разметкой, которая может принимать любые текстовые формы. На практике, как правило, она ограничивается форматами HTML и XML.

1.1 Скриптовые элементы

Скриптовые элементы используются в JSP для манипулирования объектами и выполнения вычислений, позволяющих генерировать динамическое содержание. Их можно классифицировать на следующие категории:

- комментарии;
- декларации;
- скриптлеты;
- выражения;
- выражения на специализированном языке выражений (Expression Language – EL).

В данной лабораторной работе будут использоваться только категории комментариев и выражений на специализированном языке EL.

1.2 Директивы

Директивы применяются для передачи важной информации JSP-ядру. Хотя директивы ничего не отображают, они являются мощным механизмом для предоставления информации о страницах, которая обычно используется на этапах трансляции и компиляции. Разработчики JSP имеют в своём арсенале следующие типы директивы:

- `page` – для задания параметров, глобально воздействующих на страницу, содержащую директиву;
- `tag` – для задания параметров, глобально воздействующих на теговый файл, содержащий директиву;
- `attribute` – для задания параметров входного атрибута тега, содержащего директиву;
- `include` – для вставки в страницу текста и/или кода на этапе трансляции;
- `taglib` – для импорта специальной библиотеки тегов, уникально идентифицированной по URI и ассоциированной с некоторым префиксом, отличающим её набор тегов от других.

1.3 Теги (действия)

Использование *тегов* позволяет вынести реализацию логики из тела JSP-страницы в отдельные файлы или Java-классы, и повысить степень повторного использования, простоты и удобства при работе с ними. Существует три типа тегов:

- стандартные;
- теги библиотеки JSTL;
- специальные или собственные (custom).

1.3.1 Стандартные теги

Стандартные теги – это предопределённые теги, предназначенные для лёгкого и быстрого выполнения типовых операций в JSP. В арсенале разработчика имеются стандартные действия:

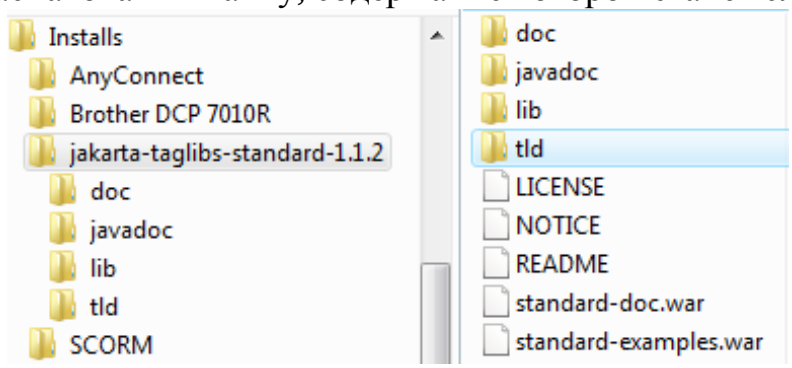
- `<jsp:include>`
- `<jsp:param>`
- `<jsp:useBean>`
- `<jsp:getProperty>`
- `<jsp:setProperty>`
- `<jsp:forward>`

1.3.2 JSTL-теги

Java Standard Template Library (JSTL) – это библиотека специализированных тегов, предназначенных для выполнения различных базовых операций по управлению потоком исполнения страницы, выводом и форматированием данных, обработкой XML-документов и т.д. Включенные в неё теги можно использовать сразу после подключения библиотеки к проекту.

JSTL является бесплатно доступной и может быть загружена по следующему адресу: http://jakarta.apache.org/site/downloads/downloads_taglibs-

standard.cgi (необходимо загрузить файл *1.1.2.zip*). Загруженный архив затем необходимо распаковать в папку, содержание которой станет следующим:



Библиотека состоит из двух основных компонентов:

- Двух JAR-файлов, содержащих откомпилированные классы тегов (архивы *jstl.jar* и *standard.jar* расположены во вложенной папке */lib*);
- Набора дескрипторов, описывающих теги библиотеки, какие Java-классы им соответствуют, какие параметры эти теги допускают и т.п. (дескрипторы *c.tld*, *fmt.tld*, *fn.tld*, *sql.tld*, *x.tld* расположены во вложенной папке */tld*).

Для использования JSTL необходимо подключить её к веб-приложению. Для этого необходимо:

1. Скопировать файлы *jstl.jar* и *standard.jar* в папку */WEB-INF/lib*.
2. Скопировать дескрипторы *c.tld*, *fmt.tld*, *fn.tld*, *sql.tld*, *x.tld* в папку */WEB-INF/tlds*.
3. Добавить в дескриптор веб-приложения */WEB-INF/web.xml* сведения о библиотеках тегов, для чего следует выполнить следующие действия:
   3.1. Открыть дескриптор веб-приложения для редактирования в XML-редакторе, для чего щёлкнув правой кнопкой мыши на файле */WEB-INF/web.xml* в контекстном меню выбрать «*Open With → XML Editor*».
   3.2. Распахнуть узел «*web-app*».
   3.3. Щёлкнув правой кнопкой мыши на узле «*web-app*» в контекстном меню выбрать «*Add Child → Context-param – login-config → JSP Config*».
   3.4. Щёлкнув правой кнопкой мыши на появившемся узле «*jsp-config*» (дочернем по отношению к «*web-app*»), в контекстном меню выбрать «*Add Child → Taglib*».
   3.5. Распахнуть появившийся узел «*taglib*» (дочерний по отношению к «*jsp-config*»).
   3.6. Значение вложенного узла «*taglib-uri*» установить равным URI подключаемой библиотеки (чтобы узнать URI подключаемой библиотеки следует открыть соответствующий ей файл с

расширением *.tld* в XML-редакторе, распахнуть узел «*taglib*» и посмотреть на значение вложенного узла «*uri*»).

3.7. Значение вложенного узла «*taglib-location*» установить равным пути к дескриптору библиотеки (например, */WEB-INF/tlds/c.tld*).

3.8. Повторить шаги 3.3–3.7 для всех подключаемых библиотек.

Замечание: Если после создания папок внутри */WEB-INF* или копирования файлов внутрь этих папок, изменения не видны в среде Eclipse, следует щёлкнуть правой кнопкой мыши на имени родительской папки и в контекстном меню выбрать пункт «*Refresh*».

Последним этапом перед использованием JSTL-тегов на JSP-странице является импорт подключенной библиотеки на страницу с помощью директивы `taglib` и связывание библиотеки с каким-либо префиксом для использования в именах тегов.

1.3.3 Custom-теги

Custom-теги (теговые расширения JSP) являются средствов инкапсуляции нестандартной функциональности в JSP. На данном этапе возможно создание трёх типов custom-тегов:

- теговые файлы;
- простые теги;
- классические теги.

В данной лабораторной работе рассматриваются только теговые файлы и простые теги.

Теговый файл представляет фрагмент JSP, содержащий разметку или JSP-код, который планируется многократно использовать. Этот фрагмент делается доступным посредством тега. В прошлом аналогичный результат достигался вынесением фрагментов кода из JSP в отдельный файл и подключения его по мере необходимости.

Для определения тегового файла следует желаемый фрагмент разметки (который и будет представлять тег) сохранить в отдельном файле с расширением *.tag* внутри какой-либо папки, вложенной в папку */WEB-INF*, например */WEB-INF/tags*. В качестве имени тега будет использоваться имя файла. Перед использованием теговые файлы также необходимо подключить. Полагая, что файлы находятся в папке */WEB-INF/tags*, можно сказать, что папка *tags* представляет множество тегов (или библиотеку). Исходя из этого, подключить теги можно с помощью директивы `taglib`.

В отличие от теговых файлов, логика *простых тегов* инкапсулируется в Java-классах, реализующих интерфейс `javax.servlet.jsp.tagext.SimpleTag` или являющихся потомками класса

`javax.servlet.jsp.tagext.SimpleTagSupport`. Объявление такого интерфейса служит двум целям:

- предоставляет тегу сведения об окружении, в котором он выполняется (посредством вызываемых контейнером методов `setJspBody()`, `setJspContext()` и `setParent()`);
- предоставляет метод для выполнения инкапсулированной функциональности (реализованной внутри метода `doTag()`).

Жизненный цикл простого тега представлен на рисунке 1.1:

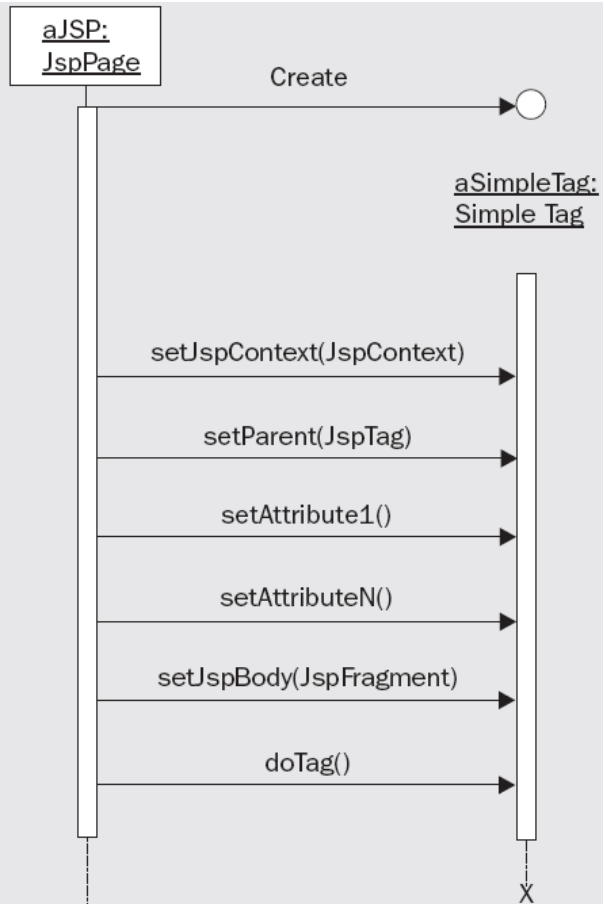


Рисунок 1.1 – Жизненный цикл простого тега

Для создания класса-обработчика тега следует стандартным образом (по аналогии с лабораторными работами 1–7) создать новый класс, сделав его потомком класса `javax.servlet.jsp.tagext.SimpleTagSupport`. Для каждого из атрибутов, поддерживаемых тегом, в классе необходимо создать поле данных экземпляра класса соответствующего типа, и определить метод-сеттер.

Характеристики тега, такие как его имя и список атрибутов, определяются в специальном дескрипторе библиотеки, который необходимо определить при создании новой библиотеки. Для создания нового дескриптора следует выполнить следующие действия:

1. Внутри папки */WEB-INF* создать (если она ещё не создана) вложенную папку, в которой будут находиться дескрипторы всех библиотек тегов, используемых приложением, например *tlds*.
2. Щёлкнув правой кнопкой мыши на имени созданной папки в контекстном меню выбрать «*New* → *Other*», в появившемся диалоговом окне раскрыть узел «*XML*» и в нём выбрать «*XML*».
3. В диалоговом окне (рисунок 1.2) указать имя файла (имеющее расширение *.tld*) и нажать «*Next*».

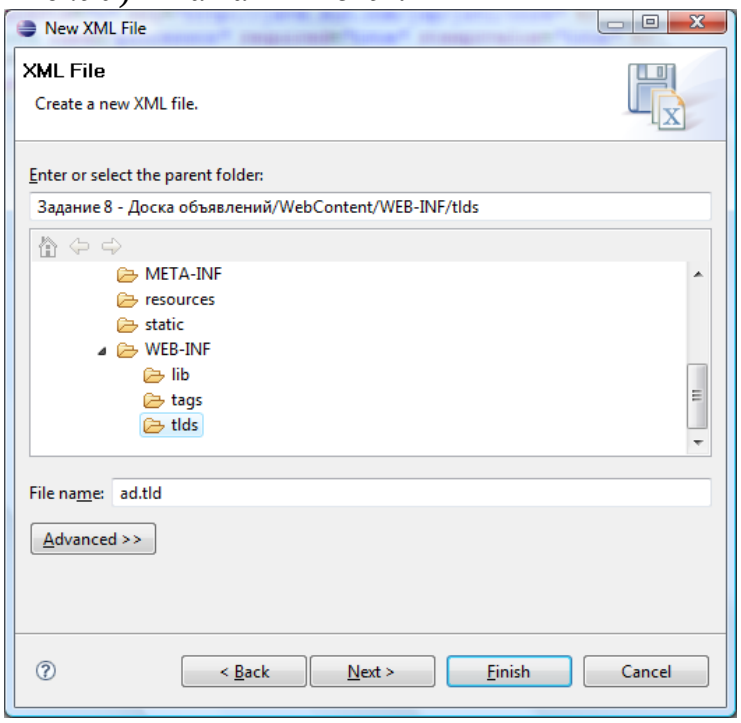


Рисунок 1.2 – Диалоговое окно создания нового XML-файла

4. В диалоговом окне (рисунок 1.3) выбрать «*Create XML file from an XML schema file*» и нажать «*Next*».

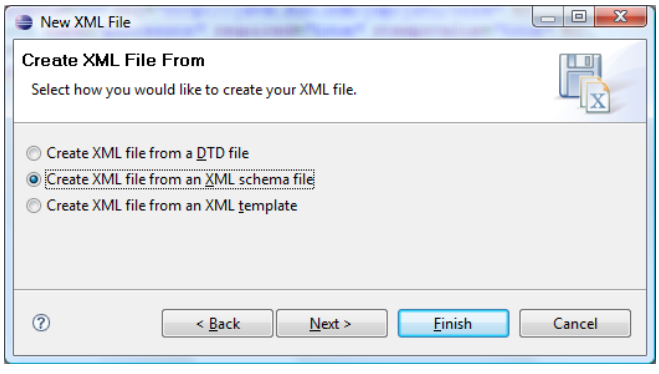


Рисунок 1.3 – Диалоговое окно выбора основы для создания файла

5. В диалоговом окне (рисунок 1.4) выбрать «*Select XML Catalog entry*», и в появившемся списке выбрать элемент «*http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd*». Нажать «*Next*».

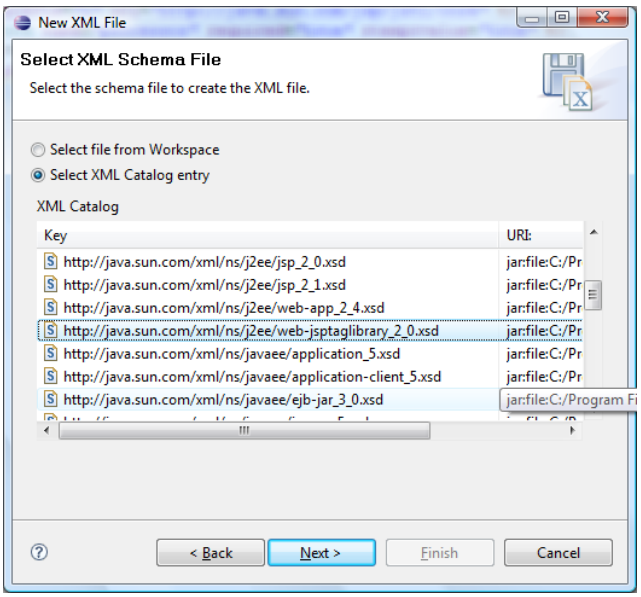


Рисунок 1.4 – Диалоговое окно выбора XML-схемы из каталога

6. В диалоговом окне (рисунок 1.5) щёлкнуть на элементе списка «*j2ee*» и нажать кнопку «*Edit*».

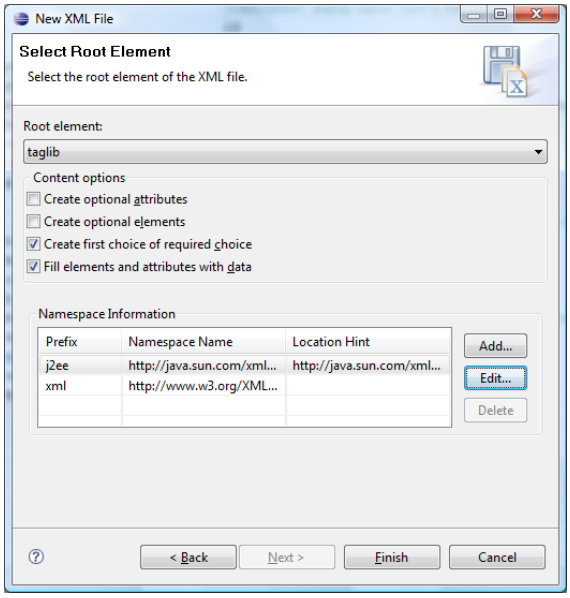


Рисунок 1.5 – Диалоговое окно выбора корневого элемента XML-файла

7. В появившемся окне (рисунок 1.6) полностью удалить значение поля «*Prefix*» и нажать «*OK*».

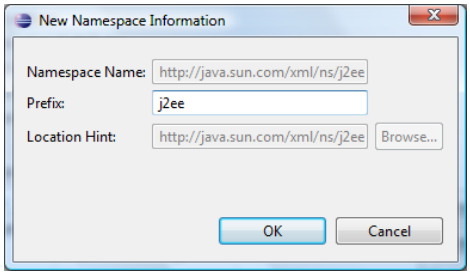


Рисунок 1.6 – Диалоговое окно изменения префикса XML-схемы

8. Нажать «*Finish*». Начальное содержание созданного XML-файла будет схоже с показанным на рисунке 1.7.

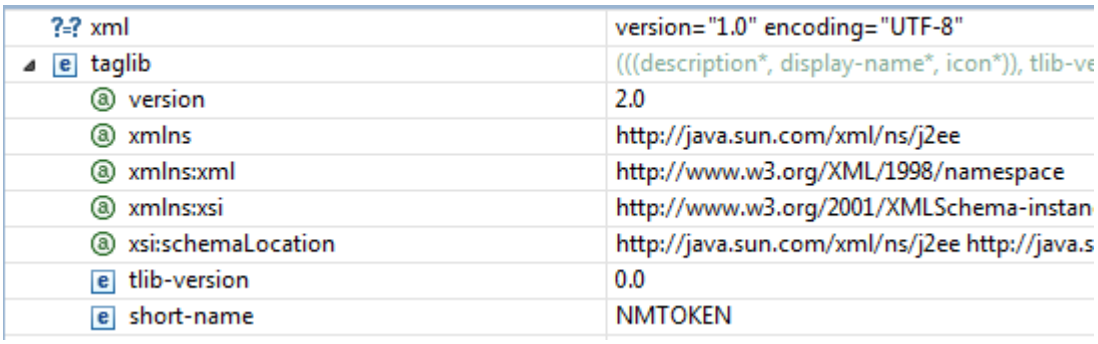


Рисунок 1.7 – Начальное содержание созданного XML-файла

9. Задать значение элемента «*tlib-version*» равным 1.0 (на данном этапе будем полагать, что этот параметр ни на что не влияет), а значение элемента «*short-name*» равным аббревиатуре имени библиотеки (например, *ad*).
10. Щёлкая правой кнопкой мыши на элементе «*taglib*» и выбирая пункты «*Add Child* → ...» последовательно добавить в него элементы «*description*», «*uri*». В «*description*» задать описание библиотеки (для чего она предназначена, кто автор и т.п.), в «*uri*» – уникальный идентификатор библиотеки, отформатированный по правилам записи адресов в Интернет (на своё усмотрение, например, *http://ivan.ivanov.rfe.bsu.by/lab8/mytags*).
11. Щёлкнув правой кнопкой мыши на элементе «*taglib*» и выбрав пункт «*Add Child* → *Tag*» добавить в него элемент «*tag*», являющийся контейнером и предназначенный для описания конкретных тегов, входящих в библиотеку.
12. Щёлкнув правой кнопкой мыши на элементе «*tag*» и выбрав пункт «*Add Child* → *Description*» добавить в него элемент «*Description*». В «*description*» задать назначение тега; в «*name*» – имя, под которым тег будет доступен на странице; в «*tag-class*» – полное имя (включая имя пакета) класса-обработчика, реализующего функциональность данного тега; в «*body-content*» выбрать либо *empty* (тег не имеет тела) либо *JSP* (тег может содержать любой JSP-фрагмент).
13. Если тег должен поддерживать атрибуты, то щёлкнув правой кнопкой мыши на элементе «*tag*» и выбрав пункт «*Add Child* → *Attribute*» добавить в него новый вложенный элемент описания атрибута.
14. Щёлкая правой кнопкой мыши на элементе «*attribute*» и выбирая пункты «*Add Child* → ...» последовательно добавить в него элементы «*description*», «*required*» и «*rtexprvalue*». В «*name*» задать имя

атрибута; в «*description*» – пояснить его назначение; в «*required*» – является ли атрибут обязательным; в «*rtexprvalue*» – может ли значение атрибута вычисляться динамически при обработке запроса.

15. Повторить шаги 11 – 12 для каждого атрибута, поддерживаемого тегом.
16. Если в библиотеке содержится более чем один тег, то необходимо повторить шаги 11–15 для каждого из них.

Замечание: При добавлении нового тега в библиотеку, в её дескриптор также необходимо добавить соответствующий элемент *tag*.

Перед использованием на JSP-страницах собственные библиотеки тегов также необходимо подключить по аналогии с подключением стандартных библиотек JSTL.

2 Краткая справка по необходимым программным компонентам

2.1 Комментарии

Комментарии являются хорошим средством объяснения логики работы JSP-страниц, и, позволяют не владеющим Java HTML-пользователям или дизайнерам получить общее представление о том, что делает конкретный блок JSP-кода.

JSP-комментарии определяются в JSP-странице как:

```
<%-- This is a JSP comment --%>
```

2.2 Выражения Expression Language (EL)

Синтаксис EL требует, чтобы все выражения находились внутри ограничителей `${` и `}`. EL-выражения могут использоваться в любом атрибуте, который допускает вычисляемое выражение: как правило, это стандартные или специализированные действия или обычная разметка шаблонов.

Для доступа к объекту применяется нотация:

`${objectName}`, например `${user}`

Для доступа к значению свойства объекта применяется нотация:

`${objectName.objectProperty}`, например `${user.name}`

Для доступа к элементу массива применяется нотация:

`${arrayName[index]}`, например `${users[0]}`

Для доступа к элементу ассоциативного массива применяется нотация:

`${arrayName[key]}`, например `${users["ivanov"]}`

Внутри EL-выражения могут находиться арифметические или логические операции. В этом случае значение выражения равно результату выполнения операции, например:

`${user!=null}` – если `user` не `null`, то результат – булево `true`;

`${user.name==null}` – если свойство `name` объекта `user` равно `null`, то результат – булево `true`;

`${complexA.re + complexB.re}` – результат равен сумме значений свойства `re` для объектов `complexA` и `complexB`.

Внутри JSP-страницы с использованием EL можно получить доступ к свойствам следующих неявно объявленных (объявленных контейнером без участия пользователя) объектов:

- `pageScope` – области текущей страницы;
- `requestScope` – области действия текущего запроса;
- `sessionScope` – области действия сессии (например, `${sessionScope.authUser}` – объект `authUser`, объявленный в области сессии);
- `applicationScope` – области действия приложения;
- `param` – входным аргументам (GET и POST), переданным странице (например, `${param.login}` – значение переданного пользователем параметра `login`).

2.3 Использование JSP-директив

2.3.1 Директива page

Используется для определения кодировки страницы (*UTF-8*), а также типа её содержания (в данном приложении это *text/html*, использующий набор символов *UTF-8*):

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
```

2.3.2 Директива tag

Используется для задания кодировки тела тегового файла (*UTF-8*), т.к. внутри тела тегового файла директива `page` использоваться не может:

```
<%@ tag pageEncoding="UTF-8" %>
```

2.3.3 Директива attribute

Используется для задания атрибутов тегового файла. При использовании тегов, реализованных как Java-классы, перечень атрибутов каждого тега указывается в файле дескриптора библиотеки. В случае теговых файлов, дескриптор библиотеки отсутствует, поэтому каждый тег должен самостоятельно заявить свои атрибуты, например:

```
<%@attribute name="ad" required="true" rtexprvalue="true"
type="bsu.rfe.java.teacher.entity.Ad" %>
```

Свойство `name` определяет имя атрибута; `required` – определяет, является ли данный атрибут обязательным; `rtexprvalue` – определяет, допускает ли атрибут значения, вычисляемые на этапе обработки запроса, или они должны быть константными (заданными ещё на этапе разработки страниц, использующих тег); `type` – задаёт тип атрибута (по умолчанию это `java.lang.String`).

2.3.4 Директива include

Используется для вставки в страницу тех фрагментов разметки или кода, которые не будут отличаться от запроса к запросу. Разовая (на этапе трансляции страницы) вставка этих фрагментов позволяет уменьшить время обработки страницей поступающих запросов:

```
<%@ include file="/static/footer.jsp" %>
```

Замечание: при задании включаемого ресурса можно использовать только ресурсы, принадлежащие тому же контексту приложения, что и страница, содержащая директиву `include`. Путь к ресурсу задаётся относительно корня приложения.

2.3.5 Директива taglib

Используется для подключения к странице библиотеки тегов или библиотеки теговых файлов. При импорте указывается префикс библиотеки, позволяющий разрешать конфликты имён тегов из нескольких библиотек, а также либо уникальный идентификатор (*uri*) библиотеки, либо расположение её теговых файлов:

```
<%-- Подключение библиотеки JSTL --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Подключение библиотеки теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
```

Замечание: При импорте библиотек тегов JAR-файлы библиотеки должны находиться в папке */WEB-INF/lib*, а дескрипторы библиотек – в какой-либо из папок, вложенных в */WEB-INF*, например */WEB-INF/tlds*. При импорте библиотек теговых файлов элементы библиотек с расширением *.tag* должны находиться внутри папок, вложенных в */WEB-INF*, например */WEB-INF/tags*.

2.4 Использование тегов

2.4.1 Стандартный тег <jsp:include>

Тег включаетедополнительных статические и динамические компоненты в JSP. Синтаксис тега следующий:

```
<jsp:include page="urlSpec"></jsp:include>
```

Атрибут `page` – задаёт относительный адрес ресурса для включения. Различие между директивой `include` и тегом `<jsp:include>` принципиально: директива выполняется только один раз во время трансляции, в то время как тег выполняется при обработке каждого запроса. Ресурсу, подключение которого выполняется, могут быть переданы дополнительные параметры с помощью вложенных тегов `<jsp:param>`.

2.4.2 Стандартный тег <jsp:forward>

Тег позволяет контейнеру осуществить переадресацию запроса (во время выполнения) с одного ресурса на другой, существующий в контексте текущего веб-приложения. Это может быть как статический ресурс, так и сервлет или JSP-страница. Действие `<jsp:forward>` немедленно прерывает выполнение текущей страницы. Синтаксис тега следующий:

```
<jsp:forward page="relativeURL"></jsp:forward>
```

Атрибут `page` идентифицирует относительный URL ресурса, на который выполняется переадресация. При этом могут передаваться дополнительные параметры с помощью вложенных тегов `<jsp:param>`.

2.4.3 Стандартный тег <jsp:attribute>

Тег применяется для определения атрибутов родительского тега в случаях, когда их задание в открывающем элементе затруднительно (например, из-за наличия кавычек в теле атрибута – *message="Объект "Пользователь" не найден"*). Синтаксис тега следующий:

```
<jsp:attribute name="attrName">attrValue</jsp:attribute>
```

Пример:

```
<jsp:attribute name="message">Объект "Пользователь" не найден</jsp:attribute>
```

2.4.4 Стандартный тег <jsp:doBody>

Тег позволяет указать место в теговом файле, куда должно быть подставлено фактическое тело тега. Синтаксис тега:

```
<jsp:doBody />
```

Пример:

Если определён файловый тег `<red>`, содержащий разметку `<font color="red"><jsp:doBody /></font>`, то фрагмент `<red>Привет!</red>` будет при отправке к пользователю преобразован в `<font color="red">Привет!</font>`.

2.4.5 Стандартный тег <jsp:useBean>

Локализует или создаёт JavaBean-компонент. До того, как с каким-либо JavaBean-компонентом можно будет работать в теле JSP, сначала необходимо

получить доступ к его экземпляру, либо путём извлечения существующего компонента из доступных областей видимости (`page`, `request`, `session`, `application`), либо созданием нового экземпляра. Изначально тег выполняет поиск существующего экземпляра компонента, если же он не может быть найден – то создаёт новый экземляр и сохраняет его в заданной области. Синтаксис тега следующий:

```
<jsp:useBean id="name" scope="scope" class="className" />
```

`id` – определяет по какому ключу экземпляр будет доступен в области (чувствителен к регистру символов); `scope` – определяет область существования (жизненный цикл) экземпляра; `class` – задаёт полное (включая пакет) имя класса, определяющего реализацию объекта (чувствительно в регистру символов).

2.4.6 Стандартный тег <jsp:setProperty>

Тег позволяет устанавливать значение свойств созданного с помощью `<jsp:useBean>` JavaBean-компонента. Синтаксис тега следующий:

```
<jsp:setProperty name="beanName" propexpr />
```

`name` – задаёт имя (идентификатор) компонента, который уже должен быть определён.

`propexpr` – может определяться одним из следующих способов:

- `property="*"` – заставляет тег проанализировать имена всех параметров, переданных в HTTP-запросе, и при совпадении имён параметров и свойств установить значения свойств компонента равными значениям параметров запроса;
- `property="propertyName"` – аналогично, но только для одного параметра;
- `property="propertyName" param="parameterName"` – установить значение свойства `propertyName` равным значению параметра `parameterName`;
- `property="propertyName" value="value"` – установить значение свойства `propertyName` равным значению `value`.

2.4.7 JSTL-тег <c:url>

Тег конструирует полный URL ресурса с учётом а) имени контекста, б) идентификатора сессии (в том случае, если поддержка cookies выключена), в) дополнительных параметров запроса (задаваемых с помощью `<c:param>`). Синтаксис тега следующий:

```
<c:url value="URL"></c:url>
```

2.4.8 JSTL-тег <c:redirect>

Тег выполняет перенаправление пользователя на указанный адрес. Если адрес локальный, то автоматически будут добавлены имя контекста приложения и идентификатор сессии (при необходимости, т.е. если поддержка cookies в браузере выключена). Синтаксис тега следующий:

```
<c:redirect url="URL"></c:redirect>
```

2.4.9 JSTL-тег <c:param>

Тег позволяет передавать дополнительные параметры запроса родительскому тегу `<c:url>` или `<c:redirect>`. Параметры будут автоматически включены в адрес URL. Синтаксис тега следующий:

```
<c:url value="URL">
      <c:param name="par1Name" value="par1Value"/>
      <c:param name="par2Name">param2Value</c:param>
</c:url>
```

Оба способа задания значения параметра могут применяться с равным успехом: как непосредственное задание значения в атрибуте `value`, так и его определение в теле тега.

2.4.10 JSTL-тег <c:out>

Тег вычисляет значение выражения и выводит его в текущий поток JSP-страницы. Выражение для вычисления передаётся через атрибут `value`, а его результат перед выводом преобразуется к строковому формату. Если результат вычисления является неопределённым (`null`), то можно задать значение по умолчанию с помощью атрибута `default` или записав его в теле тега между парными `<c:out>` и `</c:out>`. Примеры использования тега:

```
<c:out value="Good Afternoon!" />
<c:out value="${book.author}" default="Unknown"/>
<c:out value="${book.author.name}">Unknown</c:out>
```

2.4.11 JSTL-тег <c:if>

Тег используется для избирательного выполнения фрагментов страницы. Он вычисляет значение выражения, и, в зависимости от результата (true или false) исполняющий или пропускающий своё тело. Синтаксис тега:

```
<c:if test="${expression}">
  Тело тега
</c:if>
```

2.4.12 JSTL-теги <c:choose>, <c:when> и <c:otherwise>

Теги применяются для организации логического ветвления на основе взаимоисключающих друг друга альтернатив. Синтаксис тега следующий:

```
<c:choose>
  <c:when test="${expression1}">Набор действий 1</c:when>
  <c:when test="${expression2}">Набор действий 2</c:when>
  <c:otherwise>Действия по умолчанию</c:otherwise>
</c:choose>
```

2.4.13 JSTL-тег <c:forEach>

Тег используется для многократного включения некоторого содержания в страницу. Существует два альтернативных способа использования тега:

Синтаксис для перебора всех объектов в коллекции:

```
<c:forEach var="var" items="collection">
  Тело тега
</c:forEach>
```

Для повторения заданное число раз:

```
<c:forEach var="var" begin="begin" end="end">
  Тело тега
</c:forEach>
```

На каждой итерации в переменной `var` будет находиться либо текущий элемент коллекции (для первого случая), либо текущее значение счётчика (для второго случая).

2.4.14 JSTL-тег <c:remove>

Тег используется для удаления переменной из области действия (например, `sessionScope` или `applicationScope`). Синтаксис тега следующий:

```
<c:remove var="notify" scope="session" />
```

где `var` – идентификатор переменной для удаления, `scope` – заданная область.

2.4.15 JSTL-тег <fmt:requestEncoding>

Тег используется для задания кодировки, используемой для интерпретации переданных пользователем параметров. Неправильное использование или осутствие данного тега может привести к появлению на страницах странных символов вместо русских букв. Так как все разрабатываемые в лабораторной работе страницы имеют кодировку *UTF-8*, то и кодировкой входных параметров тоже должна быть установлена *UTF-8*:

```
<fmt:requestEncoding value="UTF-8" />
```

2.4.16 JSTL-тег <fmt:formatDate>

Тег используется для форматирования строкового представления даты и времени. Синтаксис тега следующий:

```
<fmt:formatDate pattern="pattern" value="${date}" />
```

Форматирует значение даты date по заданному шаблону `pattern`. Подробнее о способах форматирования даты и времени можно прочитать по адресу: http://www.roseindia.net/jstl/date-format-JstlFmt-tag.shtml.

2.5 Создание собственных тегов

Для создания собственного тега необходимо создать новый класс, потомок `javax.servlet.jsp.tagext.SimpleTagSupport`, создать в нём поля и методы-сеттеры для всех атрибутов, поддерживаемых тегом,

реализовать метод-обработчик `doTag()` и добавить сведения о теге в дескриптор библиотеки тегов.

Доступ к различным областям страницы реализуется посредством объекта класса `JspContext`:

```
// Чтение переменных из контекста приложения
User user = (User) getJspContext().getAttribute("user",
                  PageContext.APPLICATION_SCOPE);
// Сохранение переменной в контексте сессии
getJspContext().setAttribute("message", message, PageContext.SESSION_SCOPE);
```

Замечание: Так как все контексты (страницы, запроса и т.п.) хранят ссылки на переменные в виде самых общих указателей на Object, то при извлечении переменной из контекста необходимо выполнять явное преобразование типов (см. фрагмент кода выше).

3 Пример веб-приложения

Задание: составить приложение доски объявлений в Интернет на основе протокола HTTP. Доска объявлений позволяет посетителям просматривать активные объявления, регистрироваться в системе, по логину и паролю входить в личный кабинет, редактировать список своих объявлений, завершать сеанс работы с системой. Общая схема работы с приложением показана на рисунке 3.1.

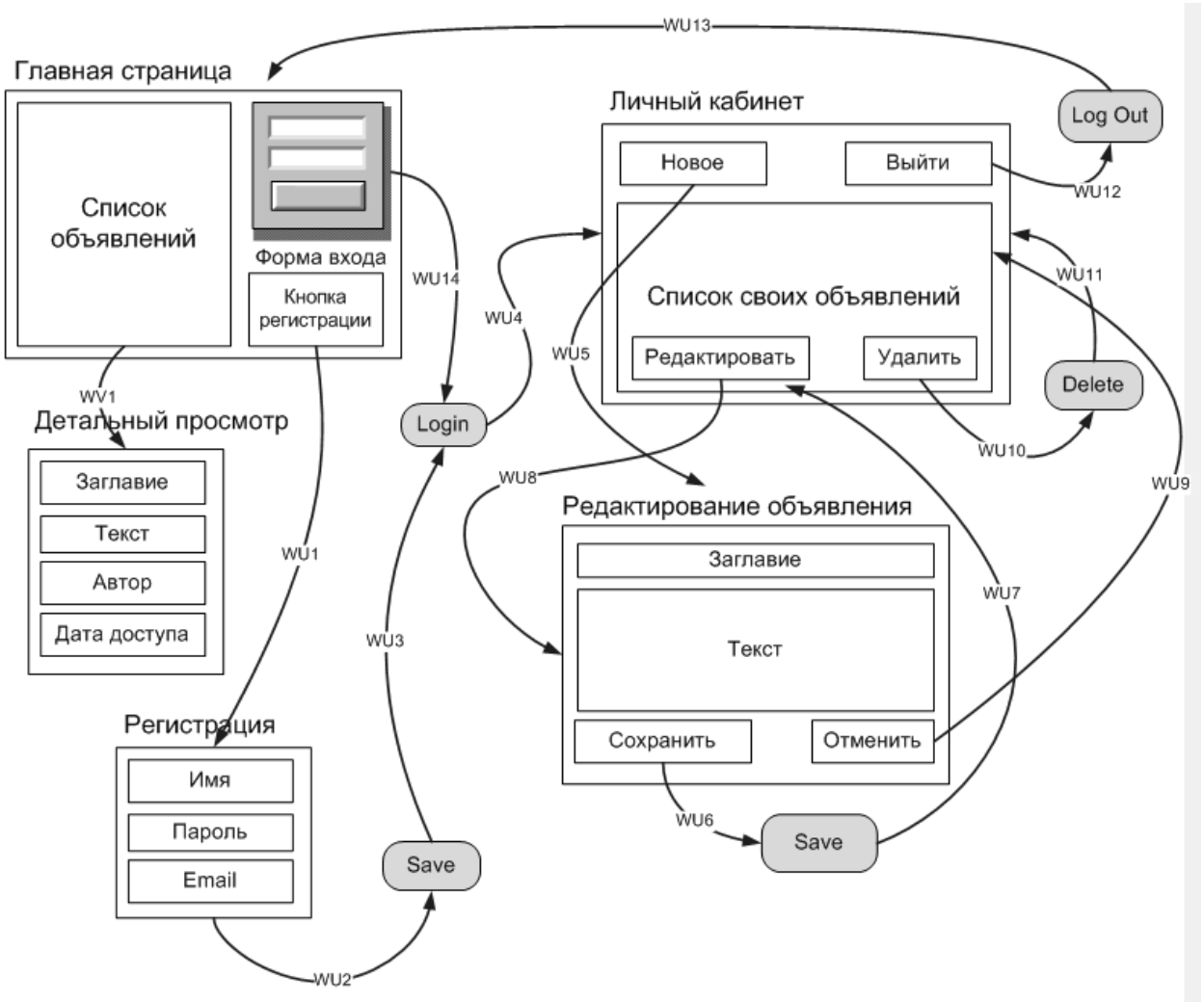


Рисунок 3.1 – Общая схема работы с приложением доски объявлений

На начальной странице пользователю отображается таблица всех активных объявлений, отсортированных определённым способом. В таблицу включены столбцы «Автор», «Тема объявления», «Дата последнего изменения объявления». При щелчке на стрелках в заголовках столбцов изменяется способ сортировки сообщений в таблице. Кроме этого, на начальной странице размещена форма для входа пользователя (содержит поля «Логин», «Пароль» и кнопку «Войти») и кнопка «Зарегистрироваться».

При щелчке на теме объявления (шаг WV1) выполняется переход на страницу детального описания объявления. Эта функциональность доступна даже незарегистрированным пользователям. На странице детального описания объявления отображаются её заглавие, полный текст, автор, дата последней модификации сообщения.

При нажатии на кнопку «Регистрация» (WU1) выполняется переход на страницу с формой регистрации нового пользователя, содержащей поля: «Логин», «Пароль», «Имя», «Email» и кнопки «Зарегистрироваться» и «Отменить». При нажатии на кнопку «Отменить» выполняется возврат на начальную страницу. При нажатии на кнопку «Зарегистрироваться» проверяется корректность введенных в полях значений (логин не пустой, логин ещё не занят, имя не пустое). Если введенные значения содержат ошибки, то пользователь возвращается на страницу регистрации и ему отображается соответствующее сообщение. Если введенные значения корректны, то создаётся новый пользователь (WU2) и автоматически выполнется процедура аутентификации (WU3).

При заполнении формы входа на начальной странице (или автоматически после регистрации) выполняется процедура логина (WU3, WU14). При этом проверяется наличие в системе пользователя с таким именем и паролем. Если пользователь существует, то выполняется переход на страницу личного кабинета (WU4). В противном случае пользователь переадресуется на начальную страницу, где ему показывается сообщение об ошибке.

Страница личного кабинета содержит список всех объявлений пользователя, отсортированных по дате последней модификации. Для каждого объявления отображаются тема, дата последней модификации и кнопки «Редактировать», «Удалить». Кроме этого в личном кабинете показываются кнопки «Новое объявление» и «Выйти».

При нажатии кнопки «Новое объявление» пользователь переадресуется на страницу с формой публикации нового объявления (WU5), содержащей поля «Тема» и «Текст», а также кнопки «Сохранить», «Отменить». В исходном состоянии поля формы являются пустыми. При нажатии на кнопку «Сохранить» в систему добавляется новое сообщение (WU6), и пользователь переадресуется на страницу личного кабинета (WU7). При нажатии на кнопку «Отменить» (WU9) выполняется переход на страницу личного кабинета без сохранения объявлений.

При нажатии на кнопку «Редактировать» рядом с каким-либо объявлением пользователь переадресуется на страницу , содержание которой аналогично описанному выше для случая нового объявления, но с некоторым

отличием: поля формы содержат текущие значения темы и текста сообщения (WU8).

При нажатии на кнопку «Удалить» рядом с каким-либо объявлением выполняется удаление объявления из системы (WU10), после чего выполняется переход на страницу личного кабинета (WU11).

При нажатии на кнопку «Выйти» (WU12) сеанс работы пользователя с системой завершается и выполняется переадресация на начальную страницу системы (WU13).

Общими требованиями к приложению являются:

1. Хранение списка пользователей и объявлений в файлах. При перезагрузке сервера объекты загружаются из файлов и помещаются в контекст приложения.
2. Приложение должно быть построено на JSP-страницах и не должно использовать скриптлеты.
3. Все страницы имеют рбщие элементы графического оформления сайта (верхний и нижний заголовки).

Внешний вид страниц приложения показан на рисунках 3.2–3.6.

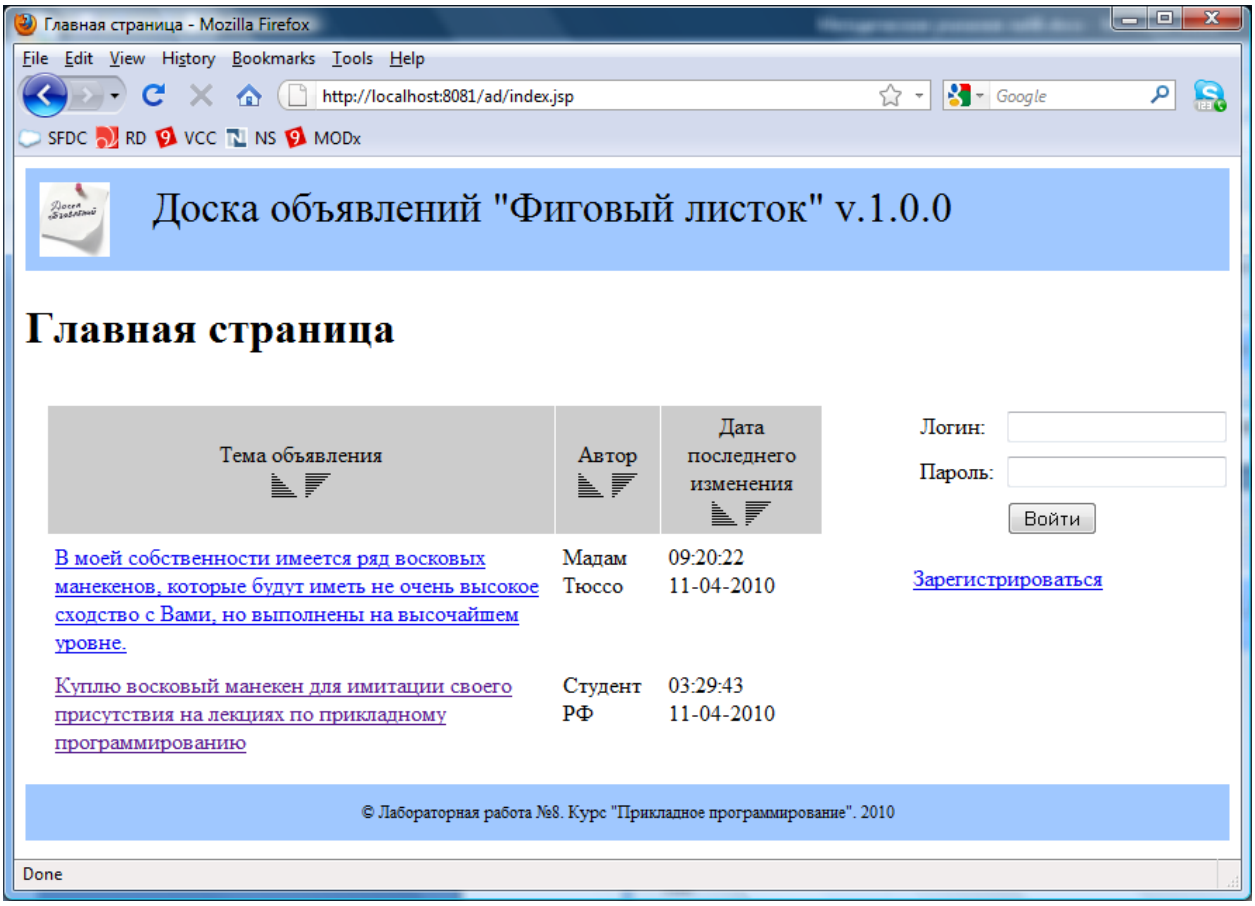


Рисунок 3.2 – Вид начальной страницы приложения

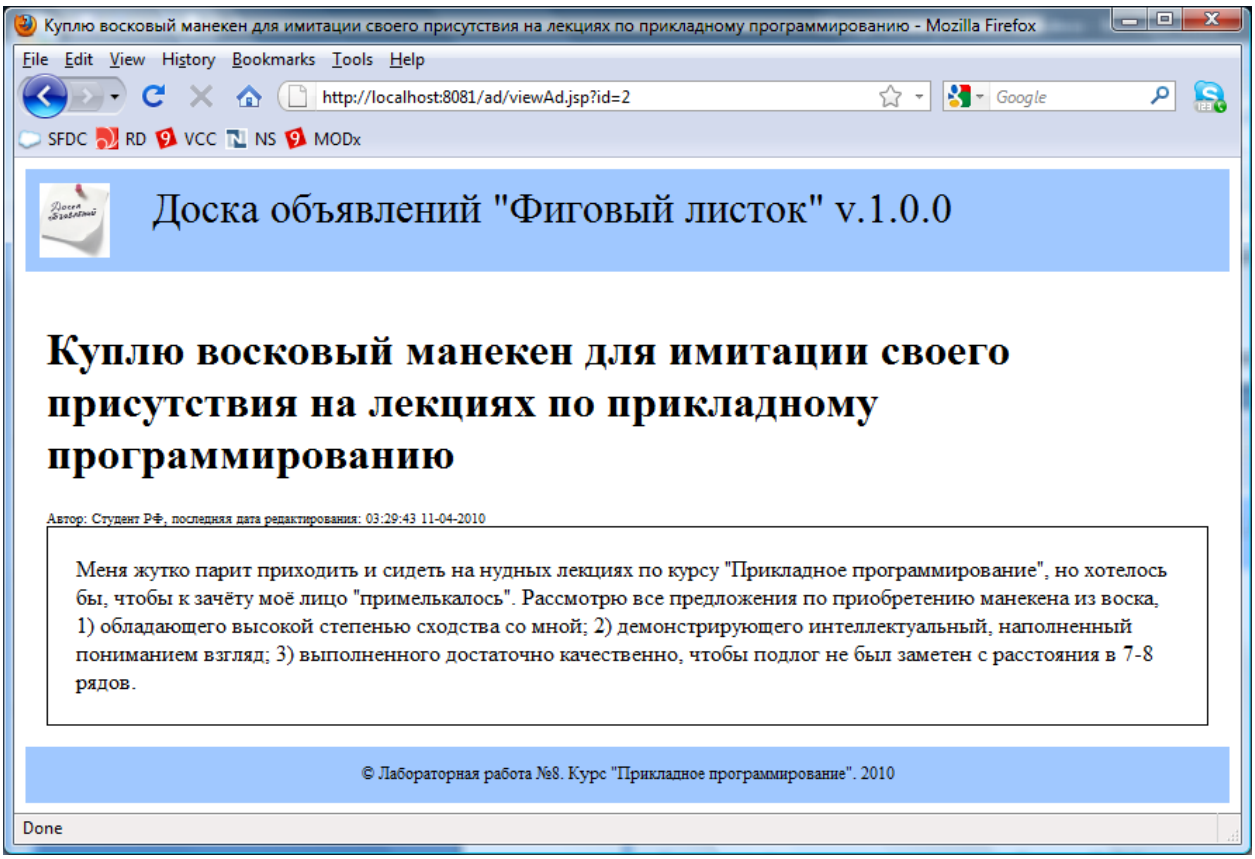


Рисунок 3.3 – Вид страницы детального просмотра объявления

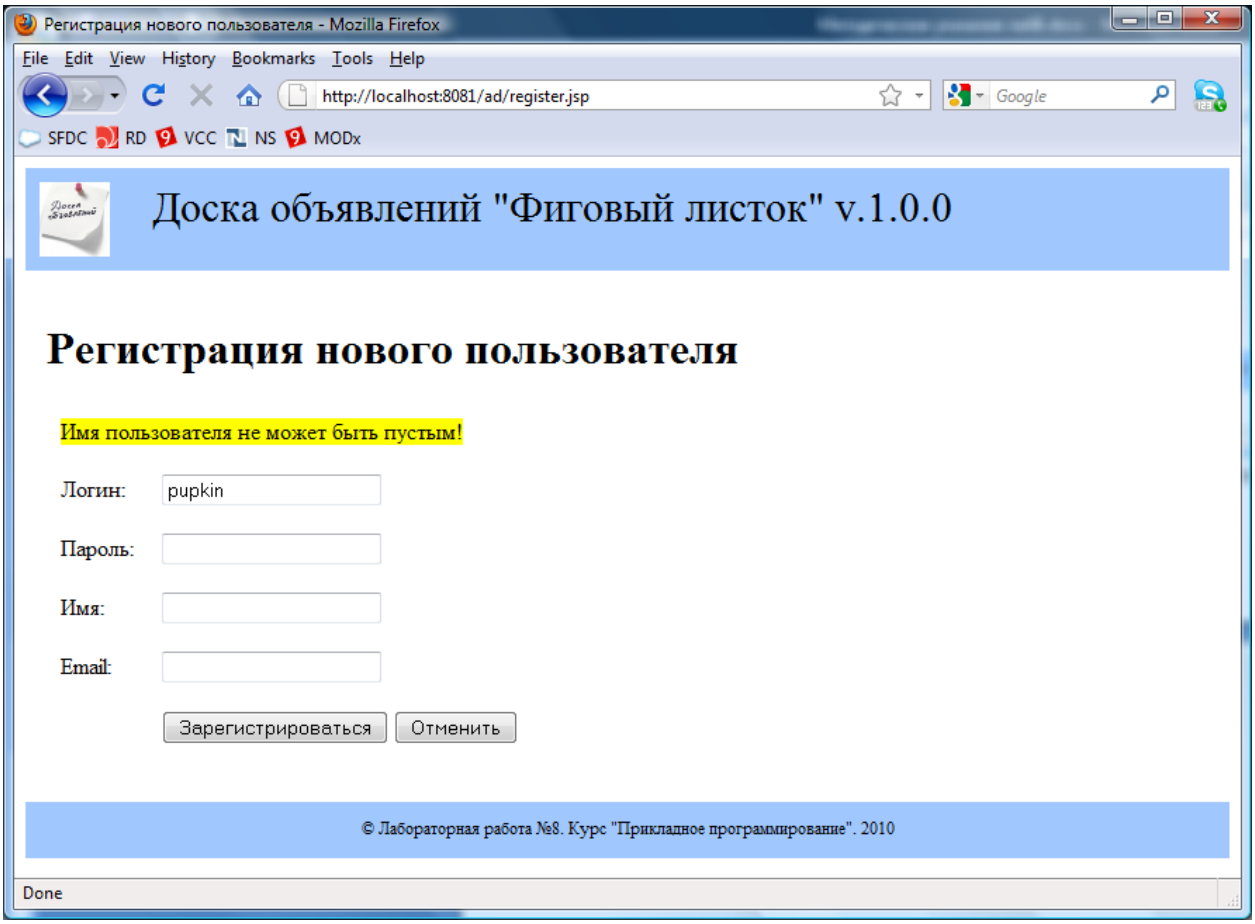


Рисунок 3.4 – Вид страницы регистрации нового пользователя

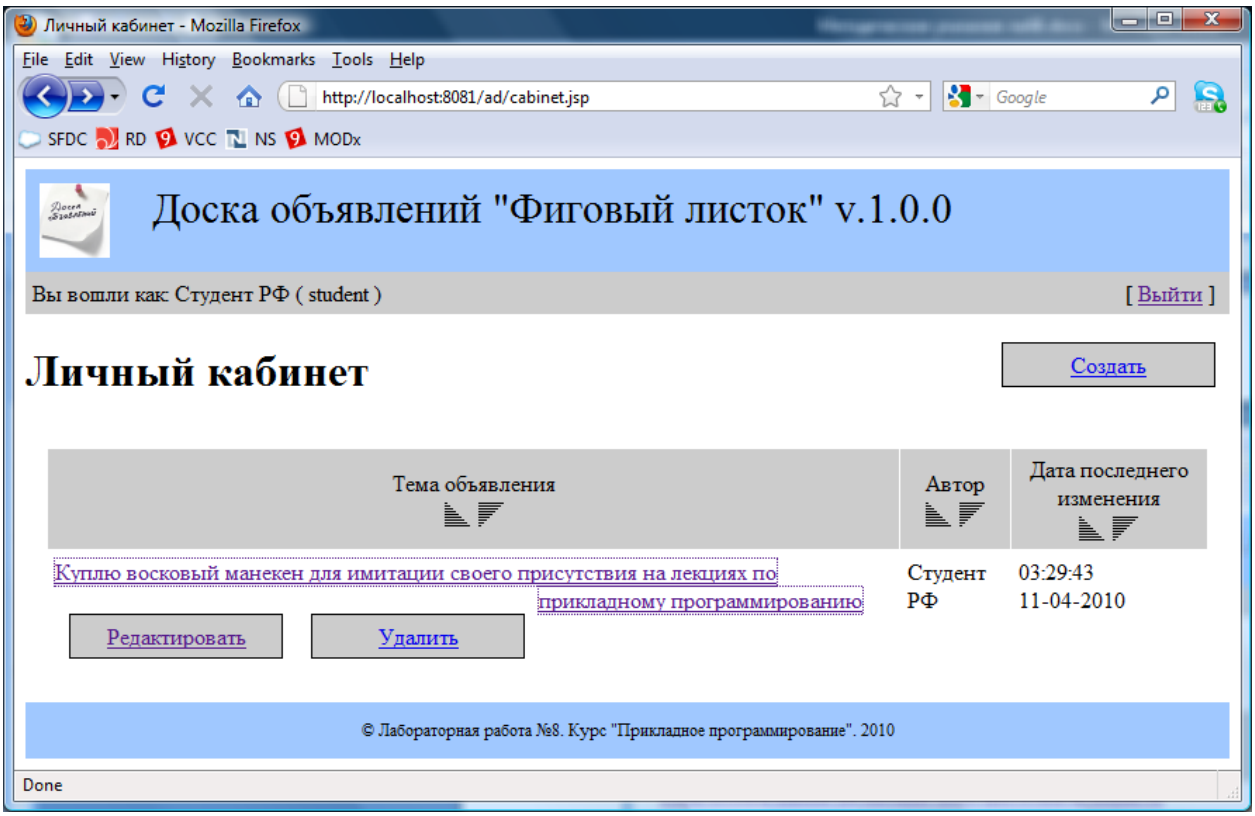


Рисунок 3.5 – Вид страницы личного кабинета пользователя

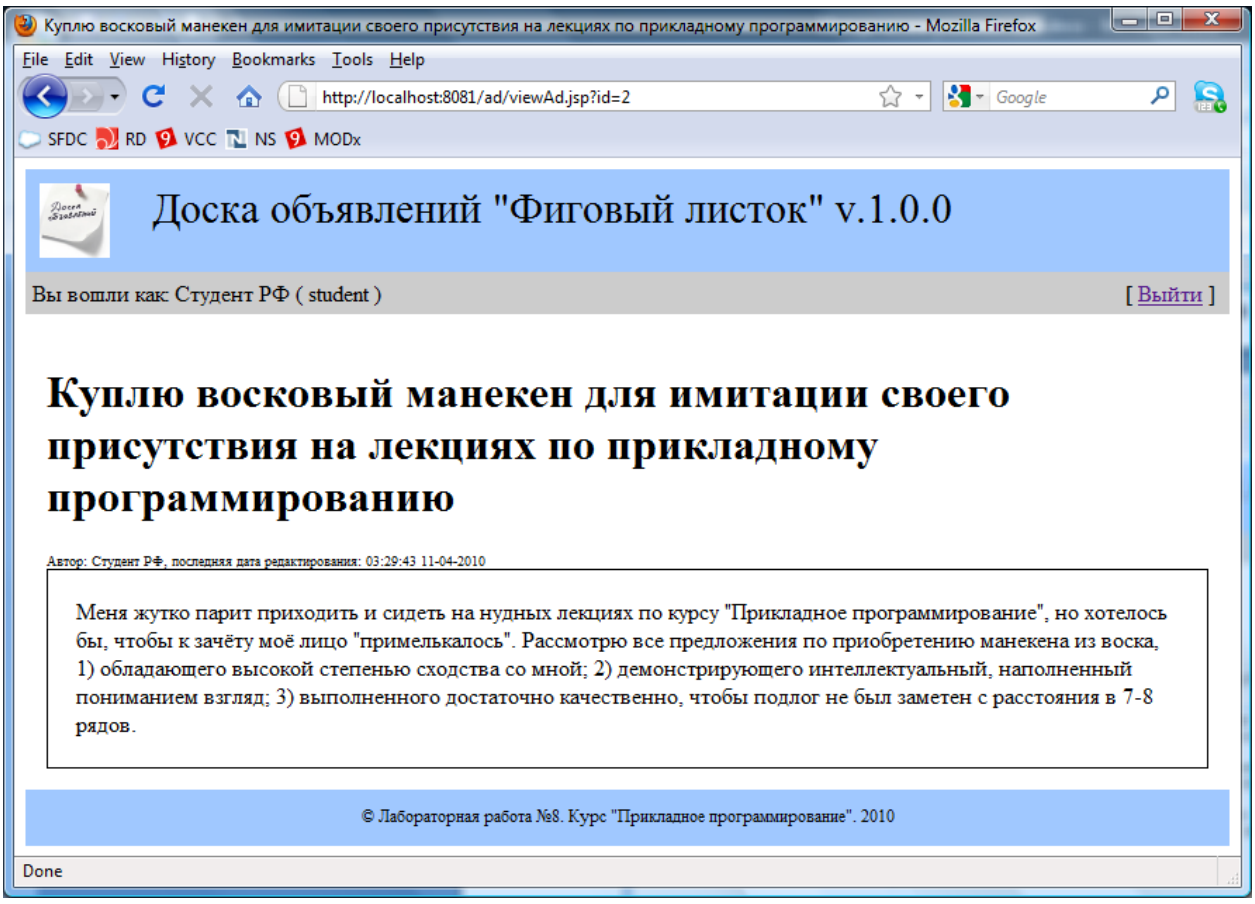


Рисунок 3.6 – Вид страницы редактирования объявления

Замечание: Фрагменты текста, выделенные на странице жёлтым цветом, являются сообщениями об ошибках и должны показываться только при возникновении ошибок, а не быть видимыми постоянно.

3.1 Структура веб-приложения

Данное приложение является самым большим и сложным из всех, рассматриваемых в лабораторном курсе, поэтому для описания его структуры будут использоваться не одна, а три диаграммы: диаграмма классов, диаграмма компонентов «Страницы приложения», диаграмма компонентов «Обработчики приложения».

3.1.1 Диаграмма классов приложения

На рисунке 3.7 приведены следующие классы приложения:

- классы, представляющие сущности приложения (пользователей, сообщения) в виде JavaBean'ов;
- классы, представляющие списки пользователей и сообщений;
- вспомогательные классы для загрузки/сохранения в файл данных приложения;
- класс сервлета, активируемый автоматически при загрузке контейнера, и выполняющий инициализацию совместно используемых структур данных приложения (списка пользователей и сообщений).

Интерфейс `Identifiable` публикует единственный метод `getId()` и указывает на свойство «идентифицируемости» экземпляров тех классов, которые его реализуют. В данном приложении под идентифицируемостью будет пониматься способность отличать экземпляры классов друг от друга по их целочисленным идентификаторам (номерам).

Классы `Ad` и `User` являются JavaBean-компонентами, содержащими ряд полей, описывающих соответствующие им сущности. Так как экземпляры этих классов являются различаемыми по идентификаторам, и должны иметь возможность сохранения в файл и загрузки из файла, то классы реализуют интерфейсы `Serializable` и `Identifiable`.

Параметризованный (generic) класс `ListOfIdentifiables` предназначен для хранения множества идентифицируемых объектов (реализующих `Identifiable`). Класс способен вычислять следующий свободный идентификатор (посредством метода `getNextId()`), а также сохранять/восстанавливать своё состояние в/из файл(а).

Классы `AdList` и `UserList` являются потомками `ListOfIdentifiables`, и дополнительно реализуют специфические для них функции. Для списка пользователей это: добавление нового пользователя – `addUser()`, поиск пользователя по логину и по идентификатору – `findUser()`. Для списка

сообщений это: добавление нового объявления – `addAd()`, удаление объявления – `deleteAd()`, обновление объявления – `updateAd()`, получение множества всех объявлений – `getAds()`.

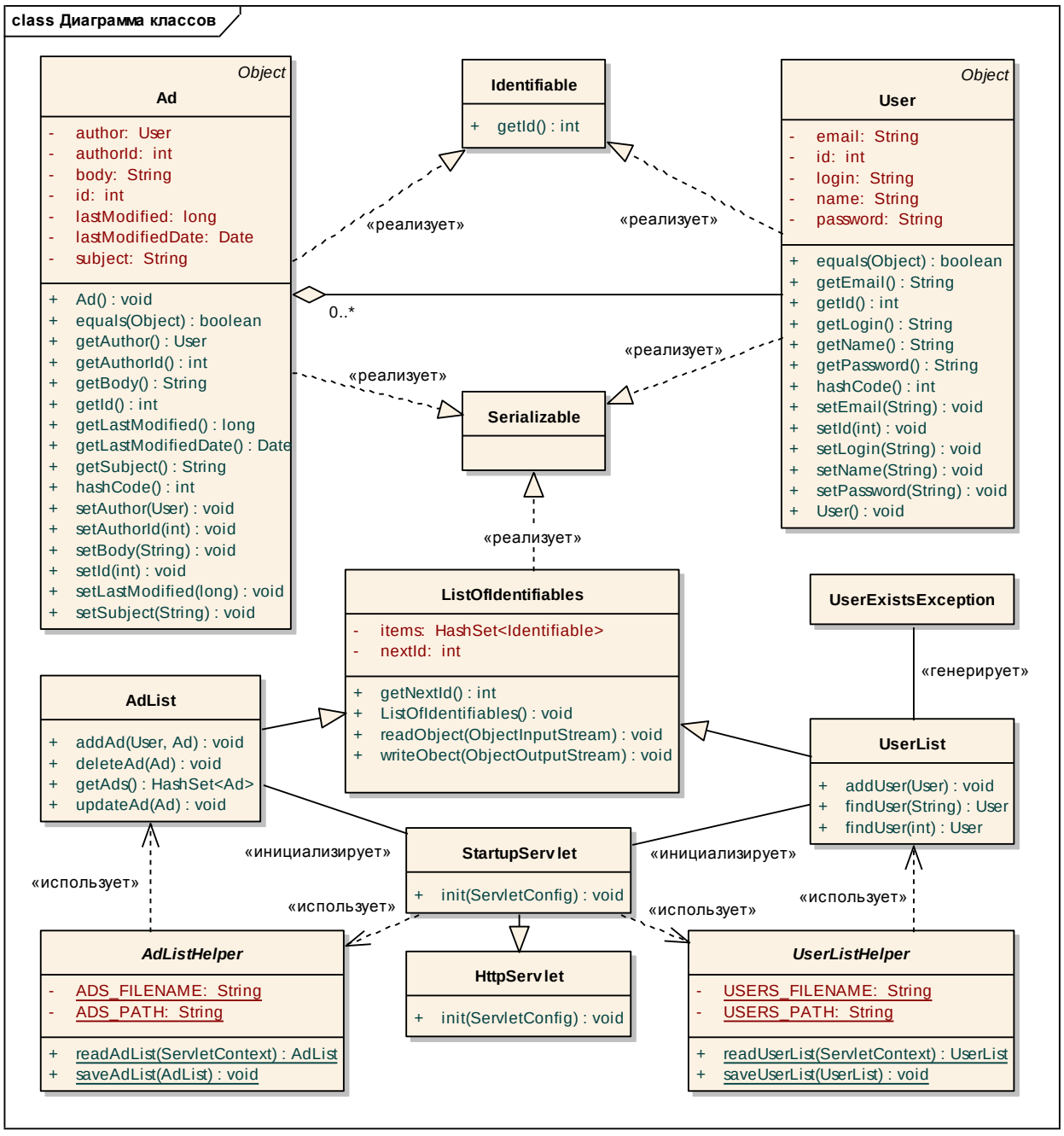


Рисунок 3.7 – Диаграмма классов приложения

Классы `AdListHelper` и `AdUserHelper` реализуют операции сохранения текущих множеств сообщений и пользователей в файлы, а также из загрузки из файлов.

Класс-сервлет `StartupServlet`, являющийся потомком базового класса `HttpServlet`, активизируется автоматически при загрузке контейнера, и его задачей является начальная загрузка множества пользователей и сообщений из файла на диске.

3.1.2 Диаграмма компонентов «Страницы приложения»

На рисунке 3.8 показаны следующие компоненты нашего приложения: страницы JSP (имеют голубой фон), теговые файлы (стандартный фон), теги (жёлтый фон).

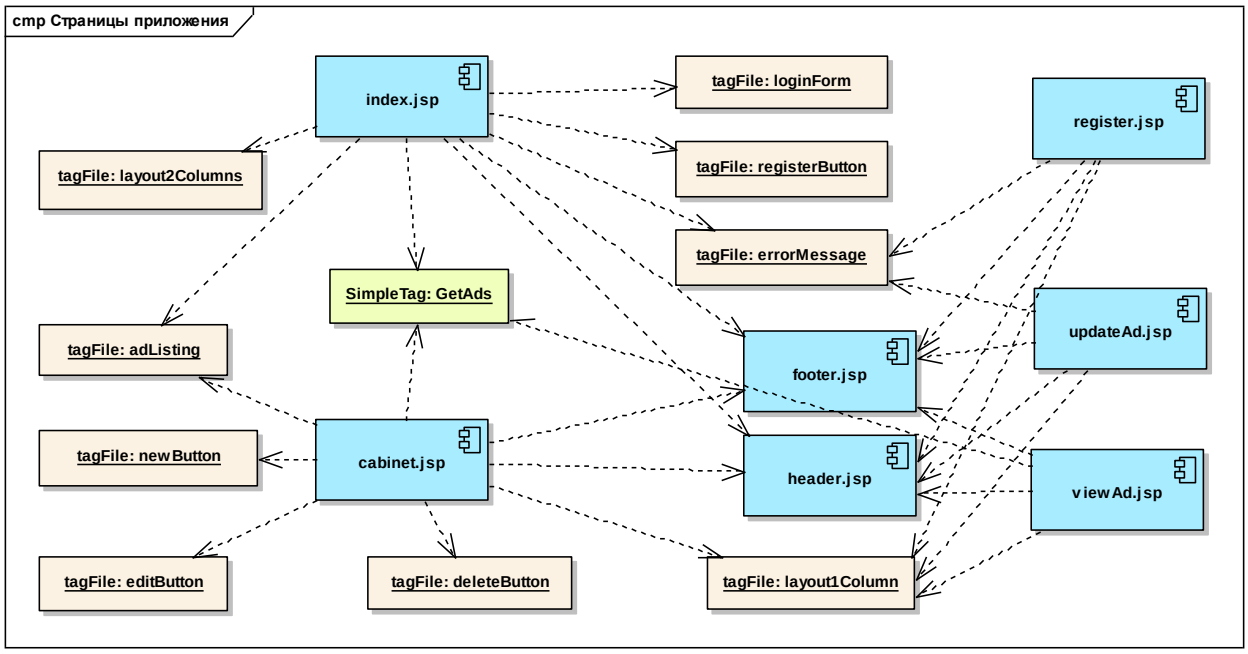


Рисунок 3.8 – Диаграмма компонентов «Страницы приложения»

Напомним, что теговые файлы представляют собой фрагменты JSP-разметки, выделенные в отдельный файл для повторного использования, а теги являются Java-классами, реализующими специальные интерфейсы.

Страницы `header.jsp` и `footer.jsp` содержат общую для всех страниц разметку верхнего и нижнего заголовков.

Стартовая страница `index.jsp` использует следующие теги:

- `layout2Columns` – для задания HTML-компоновки, содержащей две колонки содержания;
- `loginForm` – для отображения формы аутентификации пользователя;
- `registerButton` – для отображения кнопки регистрации нового пользователя;
- `errorMessage` – для отображения сообщения об ошибке;
- `getAds` – для получения списка всех объявлений, отсортированных указанным образом;
- `adListing` – для отображения списка всех объявлений в виде таблицы.

Страница детального просмотра объявления `viewAd.jsp` (открываемая при переходе WV1, см. рисунок 3.1) использует тег `getAd` для получения описания одного объявления по указанному идентификатору.

Страница регистрации нового пользователя `register.jsp` (WU1, см. рисунок 3.1) использует файловые теги одноколоночного представления содержания `layout1Column` и вывода сообщения об ошибке `errorMessage`.

Страница личного кабинета пользователя `cabinet.jsp` (WU4, см. рисунок 3.1) использует следующие теги:

- `layout1Column` – для задания HTML-компоновки, содержащей одну колонку содержания;
- `getAds` – для получения списка объявлений пользователя, отсортированных указанным образом;
- `adListing` – для отображения списка объявлений в виде таблицы;
- `newButton` – для отображения кнопки создания нового объявления;
- `editButton` – для отображения кнопки редактирования объявления рядом с каждым объявлением из списка;
- `deleteButton` – для отображения кнопки удаления объявления рядом с каждым объявлением из списка.

Страница редактирования объявления `updateAd.jsp` (WU5 и WU8, см. рисунок 3.1) применяется как при создании нового, так и при изменении существующего объявления, и использует файловые теги одноколоночного представления содержания `layout1Column` и вывода сообщения об ошибке `errorMessage`.

3.1.3 Диаграмма компонентов «Обработчики приложения»

На рисунке 3.9 показаны страницы JSP, являющиеся непосредственным исполнителями различных операций (аутентификации, регистрации и т.п.), а также используемые ими теги (выделены жёлтым фоном).

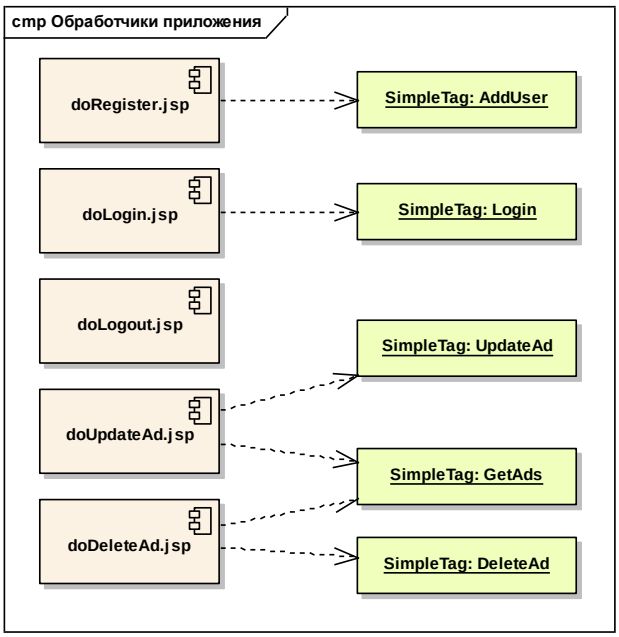


Рисунок 3.9 – Диаграмма компонентов «Обработчики приложения»

Обработчик регистрации нового пользователя `doRegister.jsp` (WU2, см. рисунок 3.1) использует тег `AddUser` для добавления нового пользователя во множество зарегистрированных пользователей приложения.

Обработчик аутентификации пользователя `doLogin.jsp` (WU3 и WU14, см. рисунок 3.1) использует тег `Login` для проверки корректности введенных логина и пароля.

Обработчик выхода из системы `doLogout.jsp` (WU12, см. рисунок 3.1) не использует дополнительных тегов.

Обработчик обновления объявления `doUpdateAd.jsp` (WU6, см. рисунок 3.1) использует тег `GetAds` для загрузки исходной версии объявления, а затем тег `UpdateAd` для сохранения изменений.

Обработчик удаления объявления `doDeleteAd.jsp` (WU10, см. рисунок 3.1) использует тег `GetAds` для загрузки исходной версии объявления, а затем тег `DeleteAd` для его удаления.

3.2 Подготовительный этап

Этап подготовки веб-приложения выполняется аналогично лабораторной работе №7, но в качестве имени контекста приложения задаётся *ad*.

4 Задания

4.1 Вариант B

Модифицировать приложение (по вариантам).

№ п/п	Цель модификаций
1	
2	
3	
4	
5	
6	
7	
8	
9	

4.2 Вариант C

Модифицировать приложение (по вариантам).

№ п/п	Цель модификаций
1	
2	
3	
4	
5	
6	

Приложение 1. Основные классы приложения

Интерфейс Identifiable

```
package bsu.rfe.java.teacher.entity;

public interface Identifiable {
	int getId();
}
```

Базовый класс ListOfIdentifiables

```
package bsu.rfe.java.teacher.entity;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.HashSet;

public class ListOfIdentifiables<T extends Identifiable & Serializable>
implements Serializable {

	private static final long serialVersionUID = -4621472982618921772L;
	protected HashSet<T> items = new HashSet<T>();
	private transient Integer nextId;

	public ListOfIdentifiables() {
		// Исходное значение следующего незанятого идентификатора - 1
		nextId = 1;
	}

	// Читает из объектного потока набор элементов и помещает их в хэш-
массив.
	// При этом происходит вычисление максимального идентификационного
номера.
	// Данный метод автоматически вызывается в ходе процедуры
десериализации.
	@SuppressWarnings("unchecked")
	private void readObject(final ObjectInputStream in) throws IOException,
ClassNotFoundException {
		// Читаем сохраненное в файле хэш-множество
		items = (HashSet<T>)in.readObject();
		// Ищем максимальное значение идентификатора пользователя
		nextId = 0;
		for(T item : items) {
			final Integer itemId = item.getId();
			if(itemId > nextId)
				nextId = itemId;
		}
		// Доступное значение идентификатора будет на 1 больше
		// максимального имеющегося
		nextId++;
	}

	// Записывает в объектный поток содержимое хэш-массива.
	// Данный метод автоматически вызывается в ходе процедуры
десериализации.
	private void writeObject(final ObjectOutputStream out) throws
IOException {
```

```
            // Записываем весь хэш-массив в объектный поток
            out.writeObject(items);
      }

      // Возвращает следующее значение идентификатора.
      protected int getNextId() {
            return nextId++;
      }
}
```

Класс User

```
package bsu.rfe.java.teacher.entity;

import java.io.Serializable;

public class User implements Serializable, Identifiable {

      private static final long serialVersionUID = -5363773994153628499L;

      // Идентификатор пользователя
      private int id;
      // Логин пользователя
      private String login = "";
      // Имя пользователя
      private String name = "";
      // Пароль пользователя
      private String password = "";
      // Email пользователя
      private String email = "";

      public int getId() {
            return id;
      }

      public void setId(int id) {
            this.id = id;
      }

      public String getLogin() {
            return login;
      }

      public void setLogin(String login) {
            this.login = login;
      }

      public String getName() {
            return name;
      }

      public void setName(String name) {
            this.name = name;
      }

      public String getPassword() {
            return password;
      }

      public void setPassword(String password) {
            this.password = password;
      }
```

```
	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public int hashCode() {
		return id;
	}

	public boolean equals(Object obj) {
		// Если obj - ссылка на другой объект, равна this, то это один и
тот же объект
		if (this == obj)
			return true;
		// Если ссылка на другой объект - null, то объекты не равны

		if (obj == null)
			return false;
		// Если классы объектных ссылок не совпадают, объекты не равны

		if (getClass() != obj.getClass())
			return false;
		User other = (User) obj;
		// Результат сравнения решается равенством идентификаторов
		if (id != other.id)
			return false;
		return true;
	}
}
```

Класс Ad

```
package bsu.rfe.java.teacher.entity;

import java.io.Serializable;
import java.util.Calendar;
import java.util.Date;

public class Ad implements Serializable, Identifiable {

	private static final long serialVersionUID = -1777984074044025486L;
	// Идентификатор сообщения
	private int id = 0;
	// Заголовок сообщения
	private String subject = "";
	// Текст сообщения
	private String body = "";
	// Автор сообщения (id)
	private int authorId;
	// Автор сообщения (ссылка, не сериализуется)
	transient private User author;
	// Последнее время модификации сообщения
	private Long lastModified;
	// Последнее время модификации сообщения как объект Date
	transient private Date lastModifiedDate;

	public Ad() {
		lastModified = Calendar.getInstance().getTimeInMillis();
	}
```

```
	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getSubject() {
		return subject;
	}

	public void setSubject(String subject) {
		this.subject = subject;
	}
	public String getBody() {
		return body;
	}

	public void setBody(String body) {
		this.body = body;
	}

	public int getAuthorId() {
		return authorId;
	}
	public void setAuthorId(int authorId) {
		this.authorId = authorId;
	}

	public User getAuthor() {
		return author;
	}

	public void setAuthor(User author) {
		this.author = author;
	}

	public Long getLastModified() {
		return lastModified;
	}

	public void setLastModified(Long lastModified) {
		// При установке последнего времени изменения в секундах
		// одновременно изменяется и время последнего изменения как дата
		this.lastModified = lastModified;
		lastModifiedDate = new Date(lastModified);
	}

	public Date getLastModifiedDate() {
		return lastModifiedDate;
	}

	public int hashCode() {
		return id;
	}

	public boolean equals(Object obj) {
		// Если obj - ссылка на другой объект, равна this, то это один и
тот же объект
		if (this == obj)
			return true;
		// Если ссылка на другой объект - null, то объекты не равны
		if (obj == null)
```

```
                return false;
            // Если классы объектных ссылок не совпадают, объекты не равны
            if (getClass() != obj.getClass())
                return false;
            Ad other = (Ad) obj;
            // Результат сравнения решается равенством идентификаторов
            if (id != other.id)
                return false;
            return true;
      }
}
```

Класс UserList

```
package bsu.rfe.java.teacher.entity;

public class UserList extends ListOfIdentifiables<User> {

      private static final long serialVersionUID = 7115985836992230188L;

      public synchronized User findUser(String login) {
            // Ищем пользователя с заданным логином
            for(User user : items) {
                  if(user.getLogin().equals(login)) {
                        return user;
                  }
            }
            return null;
      }

      public synchronized User findUser(Integer id) {
            // Ищем пользователя с заданным идентификатором
            for(User user : items) {
                  if(user.getId() == id) {
                        return user;
                  }
            }
            return null;
      }

      public synchronized User addUser(User user)
                  throws UserExistsException {
            // Если пользователь с данным логином уже зарегистрирован,
генерируем исключение
            if(findUser(user.getLogin()) != null)
                  throw new UserExistsException();
            // Выбрать следующий незанятый id для автора
            user.setId(getNextId());
            // Добавить автора в список
            items.add(user);
            return user;
      }

      // Класс искючения, указывающего при попытке добавления пользователя,
      // что пользователь с заданным логином уже присутствует в системе
      public static class UserExistsException extends Exception {
            private static final long serialVersionUID = 584737643480913385L;
      }
}
```

Класс AdList

```
package bsu.rfe.java.teacher.entity;

import java.util.HashSet;

public class AdList extends ListOfIdentifiables<Ad> {

	private static final long serialVersionUID = 882150501461356499L;

	// Выполняет добавление нового объявления с автоматическим присвоением
	// идентификационного номера. Метод синхронизирован.
	public synchronized Ad addAd(User author, Ad ad) {
		// Связать автора с объявлением
		ad.setAuthorId(author.getId());
		ad.setAuthor(author);
		// Выбрать следующий незанятый id для объявления
		ad.setId(getNextId());
		// Добавить сообщение в список
		items.add(ad);
		return ad;
	}

	// Обновляет объявление
	public synchronized void updateAd(Ad ad) {
		items.add(ad);
	}

	// Удаляет объявление
	public synchronized void deleteAd(Ad ad) {
		items.remove(ad);
	}

	// Вовзращает копию содержимого списка объявлений. Метод
синхронизирован.
	@SuppressWarnings("unchecked")
	public synchronized HashSet<Ad> getAds() {
		// Клонируем объект в целях обеспечения синхронизации
		return (HashSet<Ad>)items.clone();
	}
}
```

Вспомогательный класс UserListHelper

```
package bsu.rfe.java.teacher.helper;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.servlet.ServletContext;
import bsu.rfe.java.teacher.entity.UserList;

public abstract class UserListHelper {

	// Относительный путь к файлу, в котором хранятся данные о пользователях
	private static final String USERS_FILENAME = "WEB-INF/users.dat";
	// Полный путь к файлу, в котором хранятся данные о пользователях
	private static String USERS_PATH = null;

	// Читает данные пользователей из файла хранилища и формирует на их
основе объект UserList.
	public static UserList readUserList(ServletContext context) {
```

```
		try {
			// Определяем физический путь к файлу
			USERS_PATH = context.getRealPath(USERS_FILENAME);
			// Создаем объектный поток ввода на основе файлового потока
				ObjectInputStream in = new ObjectInputStream(new
FileInputStream(USERS_PATH));
				return (UserList)in.readObject();
				//return (UserList)in.readObject();
			} catch (Exception e) {
				// Если возникли проблемы с чтением из файла, возвращаем
пустой список
				return new UserList();
			}
		}

		// Сохраняет в файле хранилища содержимое списка пользователей
		public static void saveUserList(UserList users) {
			// Путь к файлу с данными уже находится в переменной USERS_PATH
			// Она была инициализирована при загрузке данных в процессе
инициализации приложения
			synchronized (users) {
				try {
					// Создаем объектный поток вывода на основе файлового
потока
					ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(USERS_PATH));
					// Записываем содержимое объекта в поток
					out.writeObject(users);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		}
}
```

Вспомогательный класс AdListHelper

```
package bsu.rfe.java.teacher.helper;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import javax.servlet.ServletContext;

import bsu.rfe.java.teacher.entity.AdList;

public abstract class AdListHelper {

		// Логический путь к файлу, в котором хранятся данные об объявлениях
		private static final String ADS_FILENAME = "WEB-INF/ads.dat";
		// Полный путь к файлу, в котором хранятся данные об объявлениях
		private static String ADS_PATH = null;

		// Читает данные объявлениях из файла хранилища и формирует на их основе
объект AdList.
		public static AdList readAdList(ServletContext context) {
		try {
			// Определяем физический путь к файлу
			ADS_PATH = context.getRealPath(ADS_FILENAME);
			// Создаем объектный поток ввода на основе файлового потока
				ObjectInputStream in = new ObjectInputStream(new
FileInputStream(ADS_PATH));
```

```
                return (AdList)in.readObject();
            } catch (Exception e) {
                // Если возникли проблемы с чтением из файла, возвращаем
пустой список
                return new AdList();
            }
      }

      // Сохраняет в файле хранилища содержимое списка объявлений
      public static void saveAdList(AdList ads) {
            // Путь к файлу с данными уже находится в переменной ADS_PATH
            // Она была инициализирована при загрузке данных в процессе
инициализации приложения
            synchronized (ads) {
                try {
                    // Создаем объектный поток вывода на основе файлового
потока
                    ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(ADS_PATH));
                    // Записываем содержимое объекта в поток
                    out.writeObject(ads);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
      }
}
```

Инициализационный сервлет StartupServlet

```
package bsu.rfe.java.teacher.servlet;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import bsu.rfe.java.teacher.entity.Ad;
import bsu.rfe.java.teacher.entity.AdList;
import bsu.rfe.java.teacher.entity.UserList;
import bsu.rfe.java.teacher.helper.AdListHelper;
import bsu.rfe.java.teacher.helper.UserListHelper;

// Сервлет загружается автоматически при запуске приложения
public class StartupServlet extends HttpServlet {

      private static final long serialVersionUID = 1L;

      // В методе инициализации будут создаваться общие структуры данных
      public void init(ServletConfig config) throws ServletException {
            super.init(config);
            // Загрузить список пользователей
            UserList userList =
UserListHelper.readUserList(getServletContext());
            // Сохранить список пользователей в контексте сервлета
            // (для JSP это тождественно равно applicationContext)
            getServletContext().setAttribute("users", userList);
            // Загрузить список сообщений
            AdList adList = AdListHelper.readAdList(getServletContext());
            // Сохранить список объявлений в контексте сервлета
            // (для JSP это тождественно равно applicationContext)
            getServletContext().setAttribute("ads", adList);
            for (Ad ad: adList.getAds()) {
                // Т.к. в сообщениях изначально присутствует только id
автора, для удобства установим ссылки
```

```
                ad.setAuthor(userList.findUser(ad.getAuthorId()));
                // Инициализировать значения свойства lastModifiedDate
                ad.setLastModified(ad.getLastModified());
            }
        }
}
```

Приложение 2. Custom-теги приложения

Тег добавления пользователя AddUser

```
package bsu.rfe.java.teacher.tag;

import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import bsu.rfe.java.teacher.entity.User;
import bsu.rfe.java.teacher.entity.UserList;
import bsu.rfe.java.teacher.entity.UserList.UserExistsException;
import bsu.rfe.java.teacher.helper.UserListHelper;

public class AddUser extends SimpleTagSupport {
      // Поле данных для атрибута user
      private User user;

      // Метод-сеттер для установки атрибута (вызывается контейнером)
      public void setUser(User user) {
            this.user = user;
      }

      public void doTag() throws JspException, IOException {
            // Изначально описание ошибки = null (т.е. ошибки нет)
            String errorMessage = null;
            // Извлечь из контекста приложения общий список пользователей
            UserList userList = (UserList)
getJspContext().getAttribute("users", PageContext.APPLICATION_SCOPE);
            // Проверить, что логин не пустой
            if (user.getLogin()==null || user.getLogin().equals("")) {
                  errorMessage = "Логин не может быть пустым!";
            } else {
                  // Проверить, что имя не пустое
                  if (user.getName()==null || user.getName().equals("")) {
                        errorMessage = "Имя пользователя не может быть
пустым!";
                  }
            }
            // Если ошибки не было - добавить пользователя
            if (errorMessage==null) {
                  try {
                        // Непосредственное добавление пользователя делает
UserList
                        userList.addUser(user);
                        // Записать обновлённый список пользователей в файл
                        UserListHelper.saveUserList(userList);

                  } catch (UserExistsException e) {
                        // Ошибка - пользователь с таким логином уже
существует
                        errorMessage = "Пользователь с таким логином уже
существует!";
```

```
                }
            }
            // Сохранить описание ошибки (текст или null) в сессии
            getJspContext().setAttribute("errorMessage", errorMessage,
PageContext.SESSION_SCOPE);
      }
}
```

Тег удаления объявления DeleteAd

```
package bsu.rfe.java.teacher.tag;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import bsu.rfe.java.teacher.entity.Ad;
import bsu.rfe.java.teacher.entity.AdList;
import bsu.rfe.java.teacher.entity.User;
import bsu.rfe.java.teacher.helper.AdListHelper;

public class DeleteAd extends SimpleTagSupport {
      // Поле данных для атрибута
      private Ad ad;

      // Метод-сеттер для установки атрибута (вызывается контейнером)
      public void setAd(Ad ad) {
            this.ad = ad;
      }

      public void doTag() throws JspException, IOException {
            // Изначально описание ошибки = null (т.е. ошибки нет)
            String errorMessage = null;
            // Извлечь из контекста приложения общий список объявлений
            AdList adList = (AdList) getJspContext().getAttribute("ads",
PageContext.APPLICATION_SCOPE);
            // Извлечь из сессии описание текущего пользователя
            User currentUser = (User) getJspContext().getAttribute("authUser",
PageContext.SESSION_SCOPE);
            // Проверить, что объявление изменяется его автором, а не чужаком
            if (currentUser==null || (ad.getId()>0 &&
ad.getAuthorId()!=currentUser.getId())) {
                  // Произвол! Чужой, а не автор, меняет объявление!
                  errorMessage = "Вы пытаетесь изменить сообщение, к которому
не обладаете правами доступа!";
            }
            if (errorMessage==null) {
                  // Непосредственное удаление объявления делает AdList

                  adList.deleteAd(ad);
                  // Записать обновлённый список объявлений в файл
                  AdListHelper.saveAdList(adList);
            }
            // Сохранить описание ошибки (текст или null) в сессии
            getJspContext().setAttribute("errorMessage", errorMessage,
PageContext.SESSION_SCOPE);
      }
}
```

Тег доступа к объявлениям GetAds

```
package bsu.rfe.java.teacher.tag;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import bsu.rfe.java.teacher.entity.Ad;
import bsu.rfe.java.teacher.entity.AdList;
import bsu.rfe.java.teacher.entity.User;

public class GetAds extends SimpleTagSupport {

	private int id = 0;
	// Поле данных для атрибута range (диапазон объявлений)
	private String range;
	// Поле данных для атрибута sort (основание сортировки)
	private String sort;
	// Поле данных для атрибута dir (направление сортировки)
	private char dir;
	// Поле данных для атрибута var (контейнер результата)
	private String var;

	// Метод-сеттер для установки атрибута (вызывается контейнером)
	public void setId(int id) {
		this.id = id;
	}

	// Метод-сеттер для установки атрибута (вызывается контейнером)
	public void setRange(String range) {
		this.range = range.toLowerCase();
	}

	// Метод-сеттер для установки атрибута (вызывается контейнером)
	public void setSort(String sort) {
		this.sort = sort.toLowerCase();
	}

	// Метод-сеттер для установки атрибута (вызывается контейнером)
	public void setDir(char dir) {
		this.dir = Character.toLowerCase(dir);
	}

	// Метод-сеттер для установки атрибута (вызывается контейнером)
	public void setVar(String var) {
		this.var = var;
	}

	public void doTag() throws JspException, IOException {
		// Извлечь из контекста приложения общий список объявлений
		final AdList adList = (AdList) getJspContext().getAttribute("ads",
PageContext.APPLICATION_SCOPE);
		if (id>0) {
			// Если требуется извлечь данные только 1 объявления

			for (Ad ad: adList.getAds()) {
				if (ad.getId()==id) {
					// Сохранить найденное объявление в переменную
varName
```

```
                                    getJspContext().setAttribute(GetAds.this.var,
ad, PageContext.PAGE_SCOPE);
                              }
                        }
                  } else {
                        // Необходимо построение выборки объявлений
                        // Извлечь из сессии bean аутентифицированного пользователя
                        final User authUser = (User)
getJspContext().getAttribute("authUser", PageContext.SESSION_SCOPE);
                        // В этом списке будут содержаться только отобранные
объявления
                        ArrayList<Ad> sortedList = new ArrayList<Ad>();
                        // Проанализировать все объявления на доске объявлений
                        for (Ad ad: adList.getAds()) {
                              // Если режим фильтрации собственные сообщений
выключен
                              // или текущее объявление принадлежит пользователю
                              if (!"my".equals(range) || (authUser!=null &&
authUser.getId()==ad.getAuthorId())) {
                                    // Добавить объявление в список
                                    sortedList.add(ad);
                              }
                        }
                        // Как анонимный класс определить и создать экземпляр
компаратора
                        // Именно компаратор будет отвечать за сортировку объявлений
заданным способом
                        Comparator<Ad> comparator = new Comparator<Ad>() {
                              public int compare(Ad ad1, Ad ad2) {
                                    int result;
                                    // Если выбрана сортировка по дате последнего
изменения объявления
                                    if (GetAds.this.sort!=null &&
GetAds.this.sort.equals("date")) {
                                          // То результат определяется по значениям
поля lastModified
                                          result =
ad1.getLastModified()<ad2.getLastModified()?-
1:(ad1.getLastModified()>ad2.getLastModified()?1:0);
                                          // Если порядок сортировки обратный -
инвертируем результат сравнения
                                          if (GetAds.this.dir=='d') {
                                                result = -result;
                                          }
                                    } else
                                    // Если выбрана сортировка по теме объявления

                                    if (GetAds.this.sort!=null &&
GetAds.this.sort.equals("subject")) {
                                          // То результат определяется по значениям
поля subject
                                          result =
ad1.getSubject().compareTo(ad2.getSubject());
                                          // Если порядок сортировки обратный -
инвертируем результат сравнения
                                          if (GetAds.this.dir=='d') {
                                                result = -result;
                                          }
                                    } else {
                                          // Иначе сортируем по автору объявления по
значениям поля name автора
                                          result =
ad1.getAuthor().getName().compareTo(ad2.getAuthor().getName());
```

```
                                    // Если порядок сортировки обратный -
инвертируем результат сравнения
                                    if (GetAds.this.dir=='d') {
                                          result = -result;
                                    }
                              }
                              return result;
                        }
                  };
                  if (sortedList.size()==0) {
                        // Если не найдено ни одной записи, то возвратить null

                        sortedList = null;
                  } else {
                        // Отсортировать список
                        Collections.sort(sortedList, comparator);
                  }
                  // Сохранить отсортированный список в переменной с именем
varName
                  // В контексте страницы
                  getJspContext().setAttribute(GetAds.this.var, sortedList,
PageContext.PAGE_SCOPE);
            }
      }
}
```

Тег аутентификации пользователя в системе Login

```
package bsu.rfe.java.teacher.tag;

import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import bsu.rfe.java.teacher.entity.User;
import bsu.rfe.java.teacher.entity.UserList;

public class Login extends SimpleTagSupport {
      // Поле данных для атрибута login
      private String login;
      // Поле данных для атрибута password
      private String password;

      // Метод-сеттер для установки атрибута (вызывается контейнером)
      public void setLogin(String login) {
            this.login = login;
      }

      // Метод-сеттер для установки атрибута (вызывается контейнером)
      public void setPassword(String password) {
            this.password = password;
      }

      public void doTag() throws JspException, IOException {
            // Изначально описание ошибки = null (т.е. ошибки нет)
            String errorMessage = null;
            // Извлечь из контекста приложения общий список пользователей
            UserList userList = (UserList)
getJspContext().getAttribute("users", PageContext.APPLICATION_SCOPE);
            if (login==null || login.equals("")) {
                  errorMessage = "Логин не может быть пустым!";
            } else {
                  // Найти пользователя с таким логином
```

```
                  User user = userList.findUser(login);
                  // Проанализировать результат поиска
                  if (user==null || !user.getPassword().equals(password)) {
                        // Пользователь с таким логином не найден или
неправильный пароль
                        errorMessage = "Такой пользователь не существует или
указанная комбинация логин/пароль неверна!";
                  } else {
                        // Логин и пароль верны, аутентифицировать
пользователя сохранив user в сессии
                        getJspContext().setAttribute("authUser", user,
PageContext.SESSION_SCOPE);
                  }
            }
            // Сохранить описание ошибки (текст или null) в сессии
            getJspContext().setAttribute("errorMessage", errorMessage,
PageContext.SESSION_SCOPE);
      }
}
```

Тег создания и обновления объявлений UpdateAd

```
package bsu.rfe.java.teacher.tag;

import java.io.IOException;
import java.util.Calendar;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.PageContext;
import javax.servlet.jsp.tagext.SimpleTagSupport;
import bsu.rfe.java.teacher.entity.Ad;
import bsu.rfe.java.teacher.entity.AdList;
import bsu.rfe.java.teacher.entity.User;
import bsu.rfe.java.teacher.helper.AdListHelper;

public class UpdateAd extends SimpleTagSupport {
      // Поле данных для атрибута ad
      private Ad ad;

      // Метод-сеттер для установки атрибута (вызывается контейнером)
      public void setAd(Ad ad) {
            this.ad = ad;
      }

      public void doTag() throws JspException, IOException {
            // Изначально описание ошибки = null (т.е. ошибки нет)
            String errorMessage = null;
            // Извлечь из контекста приложения общий список объявлений
            AdList adList = (AdList) getJspContext().getAttribute("ads",
PageContext.APPLICATION_SCOPE);
            // Извлечь из сессии описание текущего пользователя
            User currentUser = (User) getJspContext().getAttribute("authUser",
PageContext.SESSION_SCOPE);
            // Проверить, что заголовок не пустой
            if (ad.getSubject()==null || ad.getSubject().equals("")) {
                  errorMessage = "Заголовок не может быть пустым!";
            } else {
                  // Проверить авторство пользователя - имеет ли он право на
редактирование
                  if (currentUser==null || (ad.getId()>0 &&
ad.getAuthorId()!=currentUser.getId())) {
                        // Произвол! Чужой, а не автор, меняет объявление!
                        errorMessage = "Вы пытаетесь изменить сообщение, к
которому не обладаете правами доступа!";
```

```
                }
            }
            // Если ошибки не было - обновить объявление
            if (errorMessage==null) {
                // Обновить последнюю дату модификации объявления

      ad.setLastModified(Calendar.getInstance().getTimeInMillis());

                // Если id объявлений пустой, то оно создаётся
                if (ad.getId()==0) {
                    adList.addAd(currentUser, ad);
                } else {
                    // Объявление просто обновляется
                    adList.updateAd(ad);
                }
                // Записать обновлённый список объявлений в файл
                AdListHelper.saveAdList(adList);
            }
            // Сохранить описание ошибки (текст или null) в сессии
            getJspContext().setAttribute("errorMessage", errorMessage,
PageContext.SESSION_SCOPE);
      }
}
```

Дескриптор библиотеки тегов ad.tld

```
<?xml version="1.0" encoding="UTF-8" ?>

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
    version="2.0">

  <description>Набор тегов для приложения "Доска объявлений"</description>
  <display-name>AD App Tags</display-name>
  <tlib-version>1.0</tlib-version>
  <short-name>ad</short-name>
  <uri>http://bsu.rfe.java.teacher.tag/ad</uri>

  <tag>
    <description>Выполняет добавление нового пользователя</description>
    <name>addUser</name>
    <tag-class>bsu.rfe.java.teacher.tag.AddUser</tag-class>
    <body-content>empty</body-content>
    <attribute>
        <description>Bean пользователя</description>
        <name>user</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
  <tag>
      <description>Аутентифицирует пользователя</description>
      <name>login</name>
      <tag-class>bsu.rfe.java.teacher.tag.Login</tag-class>
      <body-content>empty</body-content>
      <attribute>
            <description>Логин пользователя</description>
            <name>login</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
```

```
      <attribute>
            <description>Пароль пользователя</description>
            <name>password</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
  </tag>
  <tag>
      <description>Извлечение объявлений различными способами</description>
      <name>getAds</name>
      <tag-class>bsu.rfe.java.teacher.tag.GetAds</tag-class>
      <body-content>empty</body-content>
      <attribute>
            <description>Указывает id объявления для извлечения</description>
            <name>id</name>
            <required>false</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
      <attribute>
            <description>Задаёт имя переменной, куда будет помещён
результат</description>
            <name>var</name>
            <required>true</required>
            <rtexprvalue>false</rtexprvalue>
      </attribute>
      <attribute>
            <description>Определяет диапазон объявлений для
извлечения</description>
            <name>range</name>
            <required>false</required>
            <rtexprvalue>false</rtexprvalue>
      </attribute>
      <attribute>
            <description>Задаёт основание для сортировки
объявлений</description>
            <name>sort</name>
            <required>false</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
      <attribute>
            <description>Определяет порядок сортировки</description>
            <name>dir</name>
            <required>false</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
  </tag>
  <tag>
      <description>Обновляет (или создаёт) объявление</description>
      <name>updateAd</name>
      <tag-class>bsu.rfe.java.teacher.tag.UpdateAd</tag-class>
      <body-content>empty</body-content>
      <attribute>
            <description>Bean объявления</description>
            <name>ad</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
  </tag>
  <tag>
      <description>Удаляет объявление из системы</description>
      <name>deleteAd</name>
      <tag-class>bsu.rfe.java.teacher.tag.DeleteAd</tag-class>
      <body-content>empty</body-content>
      <attribute>
```

```
            <description>Bean объявления</description>
            <name>ad</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
      </attribute>
  </tag>
</taglib>
```

Приложение 3. Теговые файлы приложения

Тег таблицы со списком объявлений adListing.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеки --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Коллекция объявлений для показа --%>
<%@attribute name="adListing" required="true" rtexprvalue="true"
type="java.util.AbstractCollection" %>
<%-- Режим редактирования (будут ли показываться кнопки edit, delete --%>
<%@attribute name="editMode" required="false" rtexprvalue="false"
type="java.lang.Boolean" %>

<%-- Таблица с заголовками показывается только если в списке есть хотя бы
одно объявление --%>
<c:if test="${adListing!=null}">
      <table border="0" cellpadding="5" cellspacing="1">
            <tr bgcolor="#cccccc" align="center">
                  <td>
                        Тема объявления<br>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="subject"/>
                              <c:param name="dir" value="asc"/>

                        </c:url>"><img src="<c:url value="/resources/asc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="subject"/>
                              <c:param name="dir" value="desc"/>

                        </c:url>"><img src="<c:url value="/resources/desc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                  </td>
                  <td>
                        Автор<br>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="author"/>
                              <c:param name="dir" value="asc"/>

                        </c:url>"><img src="<c:url value="/resources/asc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="author"/>
                              <c:param name="dir" value="desc"/>
```

```
                        </c:url>"><img src="<c:url value="/resources/desc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                  </td>
                  <td>
                        Дата последнего изменения<br>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="date"/>
                              <c:param name="dir" value="asc"/>

                        </c:url>"><img src="<c:url value="/resources/asc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                        <a href="<c:url
value="${pageContext.request.requestURL}">
                              <c:param name="sort" value="date"/>
                              <c:param name="dir" value="desc"/>

                        </c:url>"><img src="<c:url value="/resources/desc.png"
/>" width="20" height="19" border="0" align="absmiddle"></a>
                  </td>
            </tr>
            <%-- Организовать цикл по всем объявлениям из коллекции adListing
--%>
            <c:forEach items="${adListing}" var="ad">
                  <tr valign="top">
                        <td>
                              <%-- Вывести тему объявления, являющуюся
гиперссылкой на страницу детального просмотра объявления --%>
                              <a href="<c:url value="/viewAd.jsp"><c:param
name="id" value="${ad.id}" /></c:url>"><c:out value="${ad.subject}"/></a>
                              <%-- Кнопки редактирования / удаления объявлений
показываются только в случае включенного режима редактирования --%>
                              <c:if test="${editMode==true}">
                                    <my:editButton ad="${ad}" />
                                    <my:deleteButton ad="${ad}" />
                              </c:if>
                        </td>
                        <%-- Вывести автора объявления --%>
                        <td><c:out value="${ad.author.name}"/></td>
                        <%-- Вывести дату последней модификации объявления --
%>
                        <td><fmt:formatDate pattern="hh:mm:ss dd-MM-yyyy"
value="${ad.lastModifiedDate}" /></td>
                  </tr>
            </c:forEach>
      </table>
</c:if>
```

Тег кнопки удаления объявления deleteButton.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@attribute name="ad" required="true" rtexprvalue="true"
type="bsu.rfe.java.teacher.entity.Ad" %>
<%-- Кнопка удаления показывается только если:
      1) пользователь аутентифицирован (authUser!=null);
      2) передано текущее объявление (ad!=null);
      3) id автора объявленния и id аутентифицированного пользователя
совпадают --%>
<c:if test="${sessionScope.authUser!=null && ad!=null &&
                  ad.authorId==sessionScope.authUser.id}">
```

```
      <div style="float: left; margin: 10px; margin-top: 20px; padding: 5px
0px; border: 1px solid black; background-color: #ccc; width: 150px; text-
align: center">
            <a href="<c:url value="/doDeleteAd.jsp">
                  <c:param name="id" value="${ad.id}" />
            </c:url>">Удалить</a>
      </div>
</c:if>
```

Тег кнопки редактирования объявления editButton.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ attribute name="ad" required="true" rtexprvalue="true"
type="bsu.rfe.java.teacher.entity.Ad" %>
<%-- Кнопка редактирования показывается только если:
      1) пользователь аутентифицирован (authUser!=null);
      2) передано текущее объявление (ad!=null);
      3) id автора объявленния и id аутентифицированного пользователя
совпадают --%>
<c:if test="${sessionScope.authUser!=null && ad!=null &&
                  ad.authorId==sessionScope.authUser.id}">
      <div style="float: left; margin: 10px; margin-top: 20px; padding: 5px
0px; border: 1px solid black; background-color: #ccc; width: 150px; text-
align: center">
            <a href="<c:url value="/updateAd.jsp">
                  <c:param name="id" value="${ad.id}" />
            </c:url>">Редактировать</a>
      </div>
</c:if>
```

Тег вывода сообщения об ошибке errorMessage.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%-- Проанализировать, сохранено ли в сессии сообщение об ошибке --%>
<c:if test="${sessionScope.errorMessage!=null}">
      <%-- Если да, то показать его --%>
      <div style="padding: 10px;">
            <span style="background-color: yellow;">
                  <c:out value="${sessionScope.errorMessage}" />
            </span>
      </div>
      <%-- А потом удалить --%>
      <c:remove var="errorMessage" scope="session" />
</c:if>
```

Тег одноколоночной компоновки layout1Column.tag

```
<%@ tag pageEncoding="UTF-8" %>
<div style="padding: 15px; clear: both;">
      <%-- Тело тега подставляется в этом месте --%>
      <jsp:doBody />
</div>
```

Тег двухколоночной компоновки layout2Columns.tag

```
<%@ tag pageEncoding="UTF-8" %>
```

```
<%-- Атрибуты тега - содержание левой колонки, содержание правой колонки,
ширина левой колонки, ширина правой колонки --%>
<%@ attribute name="leftColumnBody" required="false" rtexprvalue="true" %>
<%@ attribute name="leftColumnWidth" required="true" rtexprvalue="false" %>
<%@ attribute name="rightColumnBody" required="false" rtexprvalue="true" %>
<%@ attribute name="rightColumnWidth" required="true" rtexprvalue="false" %>
<div style="clear: both; width: 100%;">
      <div style="float: left; width: ${leftColumnWidth}">
            <div style="padding: 15px">
                  ${leftColumnBody}
            </div>
      </div>
      <div style="float: right; width: ${rightColumnWidth};">
            <div style="padding: 15px">
                  ${rightColumnBody}
            </div>
      </div>
      <div style="clear: both"></div>
</div>
```

Тег формы входа loginForm.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Входным атрибутом тега является processor - адрес страницы-обработчика
запроса на аутентификацию --%>
<%@ attribute name="processor" required="true" rtexprvalue="true" %>
<%-- Форма входа показывается только если в настоящий момент пользователь не
аутентифицирован --%>
<c:if test="${sessionScope.authUser==null}">
      <form action="${processor}" method="post">
            <table border="0" cellspacing="0" cellpadding="5">
                  <tr>
                        <td>Логин:</td>
                        <td><input type="text" name="login" value=""></td>
                  </tr>
                  <tr>
                        <td>Пароль:</td>
                        <td><input type="password" name="password"
value=""></td>
                  </tr>
                  <tr>
                        <td> </td>
                        <td><input type="submit" value="Войти"></td>
                  </tr>
            </table>
      </form>
</c:if>
```

Тег кнопки создания нового объявления newButton.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Кнопка нового объявления показывается только если пользователь
аутентифицирован (authUser!=null) --%>
<c:if test="${sessionScope.authUser!=null}">
      <div style="background-color: #ccc; border: 1px solid black; float:
right; margin: 10px; margin-top: 20px; padding: 5px 0px; text-align: center;
width: 150px; ">
            <a href="<c:url value="/updateAd.jsp" />">Создать</a>
```

```
      </div>
</c:if>
```

Тег кнопки регистрации нового пользователя registerButton.tag

```
<%@ tag pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Атрибут processor содержит адрес страницы, которой будет адресован
запрос на регистрацию пользователя --%>
<%@ attribute name="processor" required="true" rtexprvalue="true" %>
<%-- Ссылка на регистрацию показывается только если в настоящий момент
пользователь не аутентифицирован,
       т.е. в переменной authUser в области сессии не сохранён JavaBean
пользователя  --%>
<c:if test="${sessionScope.authUser==null}">
      <a href="${processor}">Зарегистрироваться</a>
</c:if>
```

Приложение 4. JSP-страницы приложения

Страница верхнего заголовка (/WebContent /static/header.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%-- Обработать параметр сортировки --%>
<c:if test="${param.sort!=null}">
      <c:set var="sort" scope="session" value="${param.sort}"/>
</c:if>
<%-- Обработать параметр направления сортировки --%>
<c:if test="${param.dir!=null}">
      <c:set var="dir" scope="session" value="${param.dir}"/>
</c:if>
<%-- Общая декоративная "шапка" для всех страниц --%>
<div style="background-color: #a0c8ff; padding: 10px; ">
      <img src="/ad/resources/ad.png" width="50" height="53" border="0"
align="left">
      <div style="font-family: 'Trebuchet'; font-size: 30px; height: 53px;
margin-left: 80px;">
            Доска объявлений "Фиговый листок" v.1.0.0
      </div>
</div>
<%-- Панель отображается если пользователь аутентифицирован --%>
<c:if test="${sessionScope.authUser!=null}">
      <div style="background-color: #ccc; padding: 5px">
            <div style="float: right; margin-right: 5px">
                  [ <a href="<c:url value="/doLogout.jsp" />">Выйти</a> ]
            </div>
            Вы вошли как:
            <c:out value="${sessionScope.authUser.name}" />
            ( <c:out value="${sessionScope.authUser.login}" /> )
      </div>
</c:if>
```

Страница нижнего заголовка (/WebContent/static/footer.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<div style="background-color: #a0c8ff; height: 40px;">
	<div style="font-family: 'Trebuchet'; font-size: 12px; padding-top:
11px; text-align: center;">
		&copy; Лабораторная работа №8. Курс "Прикладное программирование".
2010
	</div>
</div>
```

Главная страница приложения (/WebContent/index.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Импортировать собственную библиотеку теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>
<html>
	<head>
		<title>Главная страница</title>
		<meta http-equiv='Content-Type' content='text/html; charset=UTF-
8'/>
	</head>
	<body>
		<%-- Подключить заголовок страницы --%>
		<jsp:include page="/static/header.jsp"></jsp:include>
		<h1>Главная страница</h1>
		<%-- Вставить разметку 2-колоночной страницы --%>
		<my:layout2Columns leftColumnWidth="68%" rightColumnWidth="28%">
			<jsp:attribute name="leftColumnBody">
				<%-- Содержание левой колонки передаётся как атрибут
leftColumnBody --%>
				<%-- Извлечь список всех объявлений --%>
				<ad:getAds range="all" var="adListing"
sort="${sessionScope.sort}" dir="${sessionScope.dir}" />
				<%-- Показать объявления без возможности
редактирования --%>
				<my:adListing adListing="${adListing}"
editMode="false" />
			</jsp:attribute>
			<jsp:attribute name="rightColumnBody">
				<%-- Содержание правой колонки передаётся как атрибут
rightColumnBody --%>
				<%-- Вставить тег отображения сообщения об ошибке --%>
				<my:errorMessage />
				<%-- Вставить форму входа --%>
				<my:loginForm method="post">

					<jsp:attribute name="processor">
						<%-- Адрес страницы-обработчика задаётся
как атрибут processor
						Т.к. необходимо "склеить" имя страницы с
именем контекста,
						используется JSTL тег c:url --%>
						<c:url value="/doLogin.jsp" />
					</jsp:attribute>
				</my:loginForm>
				<%-- Вставить ссылку регистрации --%>
```

```
                        <my:registerButton>
                              <jsp:attribute name="processor">
                                    <%-- Адрес страницы с формой регистрации
передаётся как и
                                    для страницы-обработчика формы регистрации
--%>
                                    <c:url value="/register.jsp" />
                              </jsp:attribute>
                        </my:registerButton>
                  </jsp:attribute>
            </my:layout2Columns>
            <%-- Вставить нижний заголовок страницы --%>
            <%@ include file="/static/footer.jsp" %>
      </body>
</html>
```

Страница детального просмотра объявления (/WebContent/viewAd.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеки --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%-- Импортировать собственную библиотеку теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>

<%-- Получить bean требуемого объявления по переданному параметру id --%>
<ad:getAds id="${param.id}" var="ad" />
<html>
      <head>
            <%-- Вывести тему объявления как заголовок страницы --%>
            <title><c:out value="${ad.subject}" /></title>
            <meta http-equiv='Content-Type' content='text/html; charset=UTF-
8'/>
      </head>
      <body>
            <%-- Подключить заголовок страницы --%>
            <jsp:include page="/static/header.jsp"></jsp:include>
            <%-- Вставить разметку 1-колоночной страницы --%>
            <my:layout1Column>
                  <%-- Вывести тему объявления крупными буквами как заголовок
--%>
                  <h1><c:out value="${ad.subject}" /></h1>
                  <%-- Вывести служебную информацию об объявлении - кто автор?
когда изменено? --%>
                  <div style="text-decoration: italic; font-size: 10px">
                        Автор: <c:out value="${ad.author.name}" />,
                        последняя дата редактирования:
                        <fmt:formatDate pattern="hh:mm:ss dd-MM-yyyy"
value="${ad.lastModifiedDate}" />
                  </div>
                  <%-- Отобразить текст объявления в отдельной рамке --%>

                  <div style="border: 1px solid black; padding: 20px;">
                        <c:out value="${ad.body}" />
                  </div>
            </my:layout1Column>
            <%-- Вставить нижний заголовок страницы --%>
            <%@ include file="/static/footer.jsp" %>
      </body>
</html>
```

Страница регистрации пользователя (/WebContent/register.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Импортировать собственную библиотеку теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>
<html>
	<head>
		<title>Главная страница</title>
		<meta http-equiv='Content-Type' content='text/html; charset=UTF-
8'/>
	</head>
	<body>
		<%-- Подключить заголовок страницы --%>
		<jsp:include page="/static/header.jsp"></jsp:include>
		<h1>Главная страница</h1>
		<%-- Вставить разметку 2-колоночной страницы --%>
		<my:layout2Columns leftColumnWidth="68%" rightColumnWidth="28%">
			<jsp:attribute name="leftColumnBody">
				<%-- Содержание левой колонки передаётся как атрибут
leftColumnBody --%>
				<%-- Извлечь список всех объявлений --%>
				<ad:getAds range="all" var="adListing"
sort="${sessionScope.sort}" dir="${sessionScope.dir}" />
				<%-- Показать объявления без возможности
редактирования --%>
				<my:adListing adListing="${adListing}"
editMode="false" />
			</jsp:attribute>
			<jsp:attribute name="rightColumnBody">
				<%-- Содержание правой колонки передаётся как атрибут
rightColumnBody --%>
				<%-- Вставить тег отображения сообщения об ошибке --%>
				<my:errorMessage />
				<%-- Вставить форму входа --%>
				<my:loginForm method="post">

					<jsp:attribute name="processor">
						<%-- Адрес страницы-обработчика задаётся
как атрибут processor
						Т.к. необходимо "склеить" имя страницы с
именем контекста,
						используется JSTL тег c:url --%>
						<c:url value="/doLogin.jsp" />
					</jsp:attribute>
				</my:loginForm>
				<%-- Вставить ссылку регистрации --%>
				<my:registerButton>
					<jsp:attribute name="processor">
						<%-- Адрес страницы с формой регистрации
передаётся как и
						для страницы-обработчика формы регистрации
--%>
						<c:url value="/register.jsp" />
					</jsp:attribute>
				</my:registerButton>
			</jsp:attribute>
		</my:layout2Columns>
```

```
            <%-- Вставить нижний заголовок страницы --%>
            <%@ include file="/static/footer.jsp" %>
      </body>
</html>
```

Страница-обработчик регистрации пользователя (/WebContent/doRegister.jsp)

```
<%@page language="java" pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>

<%-- Указать, что мы ожидаем данные в кодировке UTF-8 --%>
<fmt:requestEncoding value="UTF-8" />
<%-- Удалить из сессии старые данные --%>
<c:remove var="userData" />
<%-- Сконструировать новый JavaBean в области видимости сессии --%>
<jsp:useBean id="userData" class="bsu.rfe.java.teacher.entity.User"
scope="session" />
<%-- Скопировать в bean все параметры из HTTP-запроса --%>
<jsp:setProperty name="userData" property="*" />
<%-- Обратиться к собственному тегу для сохранения пользователя --%>
<ad:addUser user="${userData}" />
<%-- Проанализировать переменную errorMessage в области видимости session --
%>
<c:choose>
      <c:when test="${sessionScope.errorMessage==null}">
            <%-- Ошибок не возникло, удалить из сессии сохранённые данные
пользователя --%>
            <c:remove var="userData" scope="session" />
            <%-- Инициировать процесс аутентификации --%>
            <jsp:forward page="/doLogin.jsp" />
      </c:when>
      <c:otherwise>
            <%-- Переадресовать на форму регистрации --%>
            <c:redirect url="/register.jsp" />
      </c:otherwise>
</c:choose>
```

Страница-обработчик входа пользователя (/WebContent/doLogin.jsp)

```
<%@page language="java" pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеки --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>

<%-- Указать, что мы ожидаем данные в кодировке UTF-8 --%>
<fmt:requestEncoding value="UTF-8" />
<%-- Обратиться к собственному тегу для аутентификации пользователя на основе
указанных им логина и пароля --%>
<ad:login login="${param.login}" password="${param.password}" />
<%-- Проверить сообщение об ошибке, чтобы узнать результат аутентификации --
%>
<c:choose>
      <c:when test="${sessionScope.errorMessage==null}">
            <%-- Ошибок не возникло, переадресовать на страницу личного
кабинета --%>
            <c:redirect url="/cabinet.jsp" />
```

```
	</c:when>
	<c:otherwise>
		<%-- Переадресовать на начальную страницу --%>
		<c:redirect url="/index.jsp" />
	</c:otherwise>
</c:choose>
```

Страница личного кабинета пользователя (/WebContent/cabinet.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Импортировать собственную библиотеку теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>
<%-- Если пользователь не аутентифицирован, то просмотр страницы невозможен -
-%>
<c:if test="${sessionScope.authUser==null}">
	<c:redirect url="/index.jsp" />
</c:if>
<html>
	<head>
		<title>Личный кабинет</title>
		<meta http-equiv='Content-Type' content='text/html; charset=UTF-
8'/>
		<meta http-equiv="Expires" content="Mon, 11 May 1998 0:00:00 GMT">
		<meta http-equiv="Last-Modified" content="Fri, Jan 25 2099
23:59:59 GMT">
	</head>
	<body>
		<%-- Подключить заголовок страницы --%>
		<jsp:include page="/static/header.jsp"></jsp:include>
		<%-- Вставить кнопку создания нового объявления --%>
		<my:newButton />
		<h1>Личный кабинет</h1>
		<%-- Вставить разметку 1-колоночной страницы --%>
		<my:layout1Column>
			<%-- Извлечь список объявлений пользователя --%>
			<ad:getAds range="my" var="adListing"
sort="${sessionScope.sort}" dir="${sessionScope.dir}" />
			<%-- Показать объявления с возможностью редактирования --%>
			<my:adListing adListing="${adListing}" editMode="true" />

		</my:layout1Column>
		<%-- Вставить нижний заголовок страницы --%>
		<%@ include file="/static/footer.jsp" %>
	</body>
</html>
```

Страница редактирования объявления (/WebContent/updateAd.jsp)

```
<%@page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Импортировать собственную библиотеку теговых файлов --%>
<%@taglib prefix="my" tagdir="/WEB-INF/tags" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>
```

```
<%-- Проанализировать значение параметра id. Если он пустой или нулевой - то
объявление создаётся --%>
<c:choose>
	<c:when test="${param.id==null || param.id==0}">
		<%-- Установить значение переменной title равным "Создание" --%>
		<c:set var="title" scope="page" value="Создание" />
	</c:when>
	<c:otherwise>
		<%-- Установить значение переменной title равным "Редактирование"
--%>
		<c:set var="title" scope="page" value="Редактирование" />
		<%-- Если в сессии не сохранилось данных с предыдущей неудачной
попытки --%>
		<c:if test="${sessionScope.errorMessage==null}">
			<%-- Получить bean требуемого объявления --%>
			<ad:getAds id="${param.id}" var="ad" />
			<%-- Сохранить его в контексте сессии --%>
			<c:set var="adData" scope="session" value="${ad}" />

		</c:if>
	</c:otherwise>
</c:choose>
<html>
	<head>
		<title><c:out value="${title}" /> объявления</title>
		<meta http-equiv='Content-Type' content='text/html; charset=UTF-
8'/>
	</head>
	<body>
		<%-- Подключить заголовок страницы --%>
		<jsp:include page="/static/header.jsp"></jsp:include>
		<%-- Вставить разметку 1-колоночной страницы --%>
		<my:layout1Column>
			<h1><c:out value="${title}" /> объявления</h1>
			<%-- Вставить тег отображения сообщения об ошибке --%>
			<my:errorMessage />
			<%-- Отобразить форму редактирования объявления (с
подстановкой переданных данных) --%>
			<form action="/ad/doUpdateAd.jsp" method="post">
				<%-- Если переданный параметр id больше нуля (т.е. мы
редактируем объявление),
				то сохранить его в невидимом поле --%>
				<c:if test="${param.id>0}">
					<input type="hidden" name="id"
value="${param.id}">
				</c:if>
				<table body="0" cellspacing="0" cellpadding="10"
width="90%">
					<tr>
						<td>Заголовок:</td>
						<%-- Начальное значение поля ввода равно
текущей теме объявления --%>
						<td><input type="text" name="subject"
value="${sessionScope.adData.subject}" style="width: 90%"></td>
					</tr>
					<tr>
						<td>Текст:</td>
						<%-- Начальное значение области текста
равно текущему тексту объявления --%>
						<td><textarea name="body" rows="10" "
style="width: 90%"><c:out value="${sessionScope.adData.body}"
/></textarea></td>
					</tr>
					<tr>
```

```
                                        <td> </td>
                                        <td><input type="submit"
value="Сохранить">
                                                <%-- При нажатии на кнопку
"Отменить" возвращаемся на страницу кабинета --%>
                                                <input type="button"
value="Отменить" onclick="window.location='<c:url value="/cabinet.jsp" />';">
                                        </td>
                                    </tr>
                                </table>
                            </form>
                        </my:layout1Column>
                        <%-- Вставить нижний заголовок страницы --%>
                        <%@ include file="/static/footer.jsp" %>
                </body>
</html>
```

Страница-обработчик изменения объявления (/WebContent/doUpdateAd.jsp)

```
<%@page language="java" pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>

<%-- Указать, что мы ожидаем данные в кодировке UTF-8 --%>
<fmt:requestEncoding value="UTF-8" />
<%-- Очистить переменную перед использованием --%>
<c:remove var="adData" />
<%-- Проанализировать режим работы - изменение существующего или создание
нового объявлений --%>
<c:choose>
      <%-- Если параметр id больше нуля (т.е. модифицируется существующее
объявление) --%>
      <c:when test="${param.id>0}">
            <%-- то сначала извлечь его --%>
            <ad:getAds id="${param.id}" var="ad" />
            <%-- и переместить из page в session --%>
            <c:set var="adData" scope="session" value="${ad}"/>
      </c:when>
      <c:otherwise>
            <%-- В противном случае создать новое объявление в области
видимости сессии --%>
            <jsp:useBean id="adData" class="bsu.rfe.java.teacher.entity.Ad"
scope="session" />
      </c:otherwise>
</c:choose>
<%-- Скопировать в bean параметры subject и body из HTTP-запроса --%>
<jsp:setProperty name="adData" property="subject" />
<jsp:setProperty name="adData" property="body" />
<%-- Вызвать собственный тег обновления объявления --%>
<ad:updateAd ad="${adData}"/>
<%-- Проверить сообщение об ошибке, чтобы узнать результат операции --%>
<c:choose>
      <c:when test="${sessionScope.errorMessage==null}">
            <%-- Ошибок не возникло, очистить временные данные --%>
            <c:remove var="adData" scope="session" />
            <%-- Переадресовать на страницу личного кабинета --%>
            <c:redirect url="/cabinet.jsp" />
      </c:when>
      <c:otherwise>
            <%-- Переадресовать на страницу редактирования объявления --%>
```

```
            <c:redirect url="/updateAd.jsp">
                  <%-- В качестве параметра при переадресации передать id
объявления --%>
                  <c:param name="id" value="${adData.id}" />
            </c:redirect>
      </c:otherwise>
</c:choose>
```

Страница-обработчик удаления объявления (/WebContent/doDeleteAd.jsp)

```
<%@page language="java" pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Импортировать собственную библиотеку тегов --%>
<%@taglib prefix="ad" uri="http://bsu.rfe.java.teacher.tag/ad" %>

<%-- Извлечь JavaBean требуемого объявления --%>
<ad:getAds id="${param.id}" var="ad" />
<%-- Удалить его из системы --%>
<ad:deleteAd ad="${ad}"/>
<%-- Переадресовать на страницу кабинета --%>
<c:redirect url="/cabinet.jsp" />
```

Страница-обработчик выхода пользователя (/WebContent/doLogout.jsp)

```
<%@page language="java" pageEncoding="UTF-8" %>
<%-- Импортировать JSTL-библиотеку --%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%-- Удалить из сессии JavaBean с данными пользователя --%>
<c:remove var="authUser" scope="session" />
<%-- Переадресовать на начальную страницу --%>
<c:redirect url="/index.jsp" />
```

Дескриптор веб-приложения (/WebContent/WEB-INF/web.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
  <display-name>Задание 8 - Доска объявлений</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <display-name>StartupServlet</display-name>
    <servlet-name>StartupServlet</servlet-name>
    <servlet-class>bsu.rfe.java.teacher.servlet.StartupServlet</servlet-
class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <jsp-config>
      <taglib>
            <taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
            <taglib-location>/WEB-INF/tlds/c.tld</taglib-location>
      </taglib>
      <taglib>
            <taglib-uri>http://java.sun.com/jsp/jstl/fmt</taglib-uri>
            <taglib-location>/WEB-INF/tlds/fmt.tld</taglib-location>
```

```
      </taglib>
      <taglib>
            <taglib-uri>http://java.sun.com/jsp/jstl/functions</taglib-uri>
            <taglib-location>/WEB-INF/tlds/fn.tld</taglib-location>
      </taglib>
      <taglib>
            <taglib-uri>http://java.sun.com/jsp/jstl/sql</taglib-uri>
            <taglib-location>/WEB-INF/tlds/sql.tld</taglib-location>
      </taglib>
      <taglib>
            <taglib-uri>http://java.sun.com/jsp/jstl/xml</taglib-uri>
            <taglib-location>/WEB-INF/tlds/x.tld</taglib-location>
      </taglib>
  </jsp-config>
</web-app>
```