

JAVA FOR STUDENTS

ФИЛИАЛ КАФЕДРЫ ПОИТ БГУИР В ЕРАМ
SYSTEMS

КУРС: ВЕБ-ТЕХНОЛОГИИ (JAVA)
SERVLETS FUNDAMENTALS

Olga Smolyakova , PhD

Oracle Certified Java 6 Programmer

Olga_Smolyakova@epam.com

Содержание

1. JSP. Основные определения
2. Простая JSP-страница
3. JSP Life Circle
4. JSP API
5. JSP and web.xml
6. Структура web-проекта
7. Servlet Life Circle
8. ServletRequest, ServletResponse
9. web.xml
10. .war
11. Контейнер сервлетов Tomcat

JSP. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

JSP (Java Server Pages) — технология, позволяющая веб-разработчикам динамически генерировать **HTML, XML** и другие веб-страницы.



Является составной частью единой технологии создания бизнес-приложений Java Platform, Enterprise Edition.

Технология позволяет внедрять **Java**-код, а также **EL (expression language)** в статичное содержимое страницы.

Достоинства:

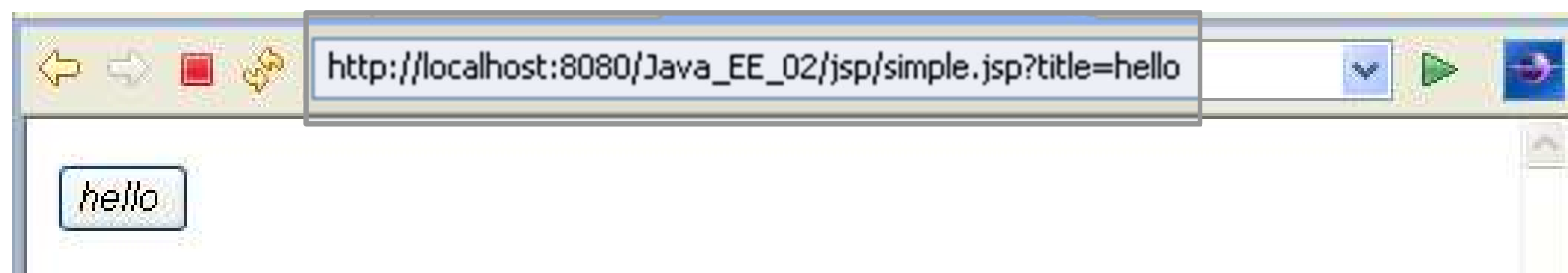
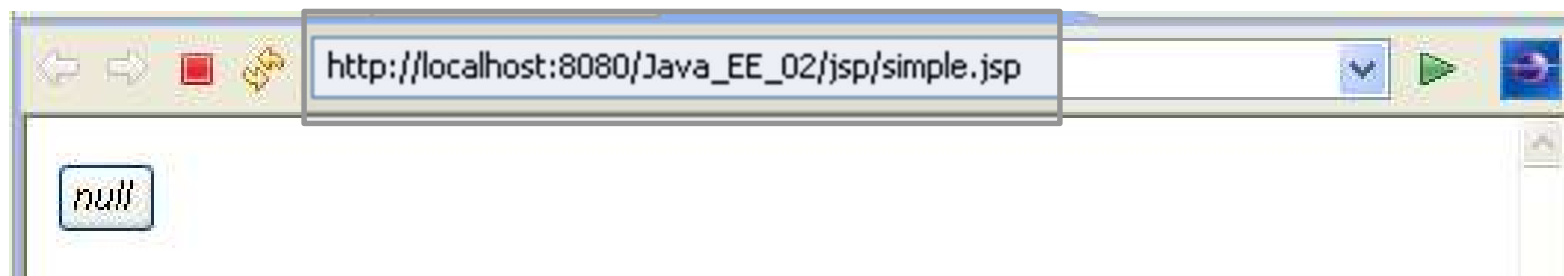
- *Разделение динамического и статического содержания.*
- *Независимость от платформы.*
- *Многократное использование компонентов.*
- *Возможность использования скриптов и тегов.*

ПРОСТАЯ JSP-СТРАНИЦА

simple.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Insert title here</title>
</head>
<body>
    <button>
        <i><%=request.getParameter("title") %></i>
    </button>
</body>
</html>
```

Результаты выполнения запросов к странице simple.jsp

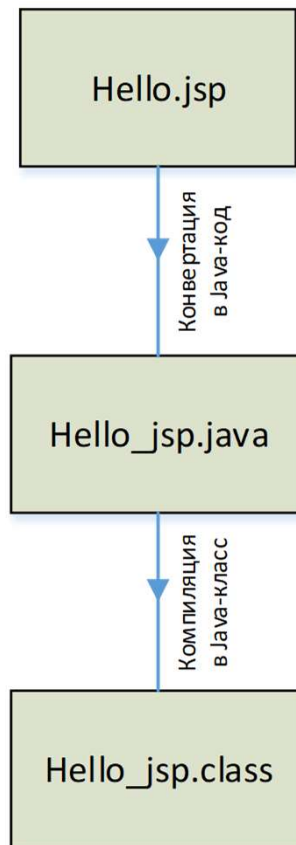


JSP LIFE CIRCLE

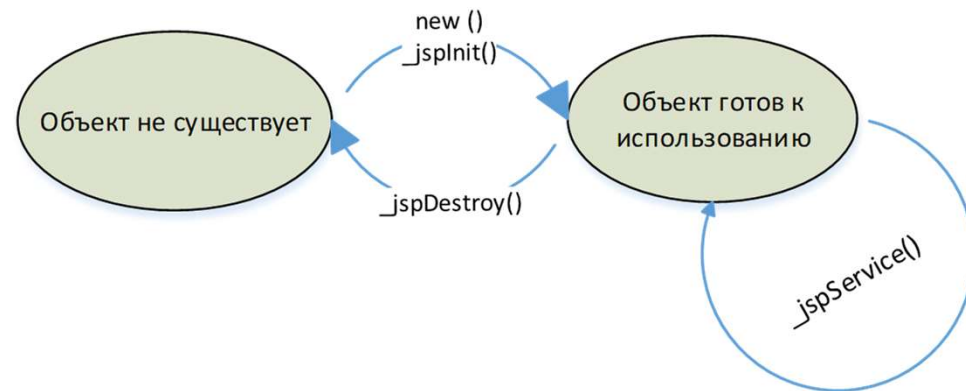
Жизненный цикл JSP включает следующие шаги (фазы):

- **Трансляция** страницы JSP – создание исходного кода сервлета. Трансляция происходит один раз, когда jsp-страница выполняет запрос впервые. Повторная трансляция не должна произойти до тех пор, пока страница не будет обновлена.
- **Компиляция** страницы JSP – компилятор переведет полученный код в байт-код.
- **Загрузка класса**
- **Создание экземпляра**
- **Вызов метода `jspInit`**
- **Вызов метода `_jspService` для каждого запроса**
- **Вызов метода `jspDestroy`**

1. Жизненный цикл класса страницы



2. Жизненный цикл объекта страницы



↑ Name				Ext	Size	Date
↑ [..]					<DIR>	07/10/2014
simple_jsp				class	4,264	07/10/2014
simple_jsp				java	3,561	07/10/2014

```

public final class simple_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {
...
    public void _jspInit() {...
    }

    public void _jspDestroy() { }

    public void _jspService(final javax.servlet.http.HttpServletRequest request,
        final javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException {}
}

```

JSP API

JSP API

Интерфейс `javax.servlet.jsp.JspPage` содержит два метода:

1. **`public void _jspInit()`** – метод вызывается для инициализации JSP. На jsp-странице можно перегрузить этот метод, используя декларации.
2. **`public void _jspDestroy()`** – метод вызывается когда JSP разрушается контейнером.

Интерфейс `javax.servlet.jsp.HttpJspPage` содержит один метод

- **`public void _jspService(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException`** – это метод вызывается контейнером каждый раз, когда приходит запрос к JSP.

JSP AND WEB.XML

Пользователю нет необходимости регистрировать jsp-страницы в дескрипторе развертывания. Достаточно разместить их в области, доступной из структуры каталогов web-приложения, и они станут доступны пользователю.

Хотя, можно зарегистрировать jsp-страницу подобно сервлету.

```
<servlet>
    <servlet-name>JspName1</servlet-name>
    <jsp-file>/instanceCheck.jsp</jsp-file>
</servlet>

<servlet-mapping>
    <servlet-name>JspName1</servlet-name>
    <url-pattern>/jspName2</url-pattern>
</servlet-mapping>
```


Теперь к **jsp**-странице можно получить доступ двумя способами:

<http://localhost:8080/myapp/instanceCheck.jsp>

или

<http://localhost:8080/myapp/jspName2>

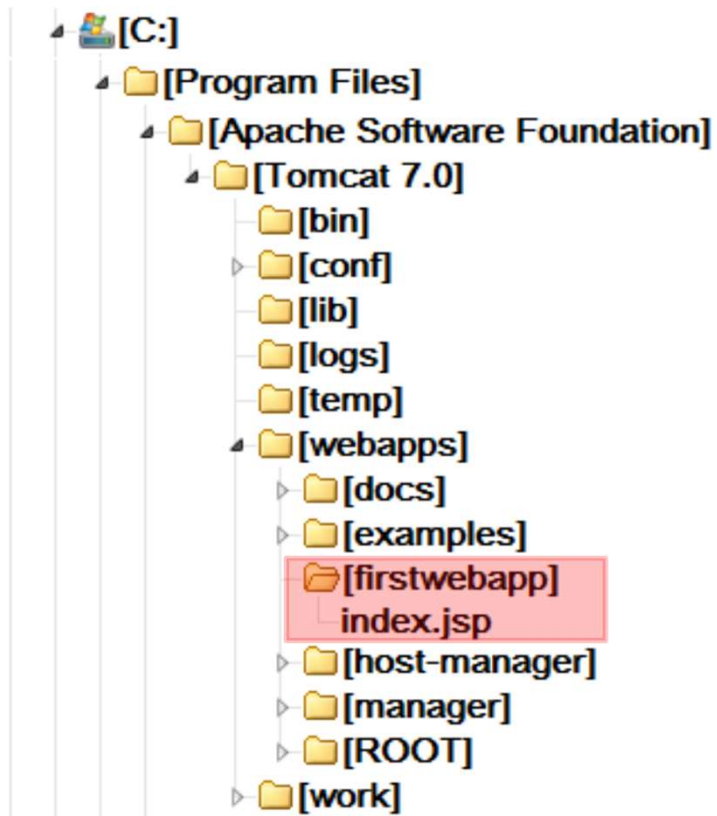


Если вы хотите предотвратить прямой доступ к jsp, размещайте свои jsp-страницы в директории WEB-INF.

СТРУКТУРА WEB-ПРОЕКТА

Простое веб-приложение. Пример 1.

Создаем каталог firstwebapp



Создаем в каталоге firstwebapp файл index.jsp со следующим содержанием

```
1 <html>
2 <head>
3   <meta http-equiv="Content-Type"
4         content="text/html; charset=utf-8">
5   <title>Insert title here</title>
6 </head>
7 <body>
8   <h1>
9     Hello, world!!!
10  </h1>
11 </body>
12 </html>
```

A screenshot of a text editor window titled 'index.jsp'. The editor displays the following HTML code: `<html>`, `<head>`, `<meta http-equiv="Content-Type" content="text/html; charset=utf-8">`, `<title>Insert title here</title>`, `</head>`, `<body>`, `<h1>`, `Hello, world!!!`, `</h1>`, `</body>`, and `</html>`. Line numbers 1 through 12 are visible on the left side of the editor.



Результат:

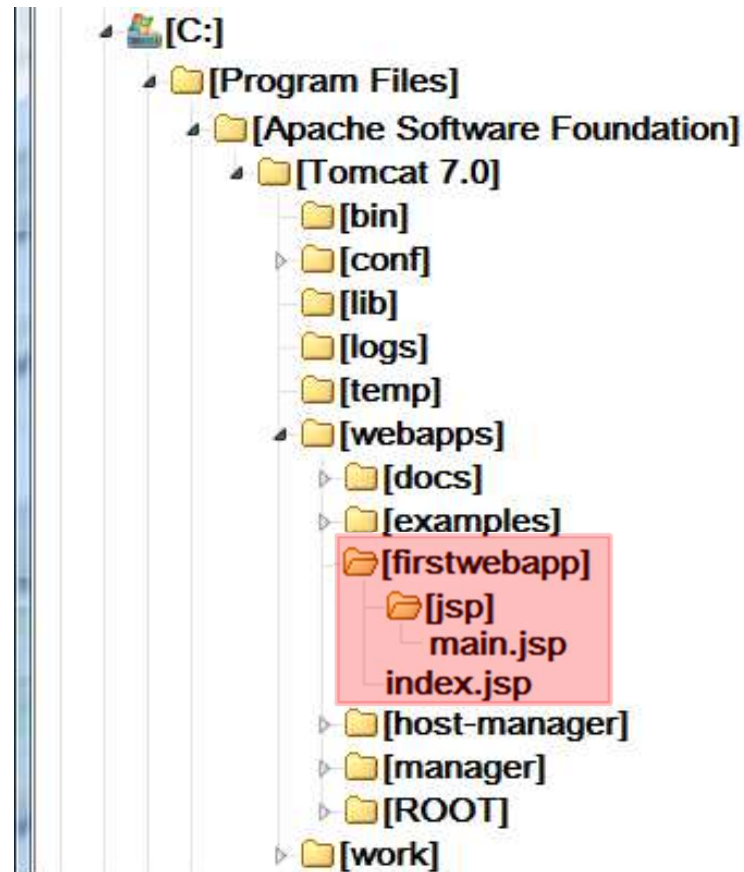
`http://127.0.0.1:8080/firstwebapp`

`http://127.0.0.1:8080/firstwebapp/index.jsp`



Простое веб-приложение. Пример 2.

Создаем в
каталоге
firstwebapp
директорию jsp
с файлом
main.jsp



```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type"
4     content="text/html; charset=utf-8">
5   <title>Insert title here</title>
6 </head>
7 <body>
8   <h1>
9     Go to
10    <a href="jsp/main.jsp" >
11      main.jsp
12    </a>
13  </h1>
14 </body>
15 </html>

```

Содержание файлов index.jsp и main.jsp

```

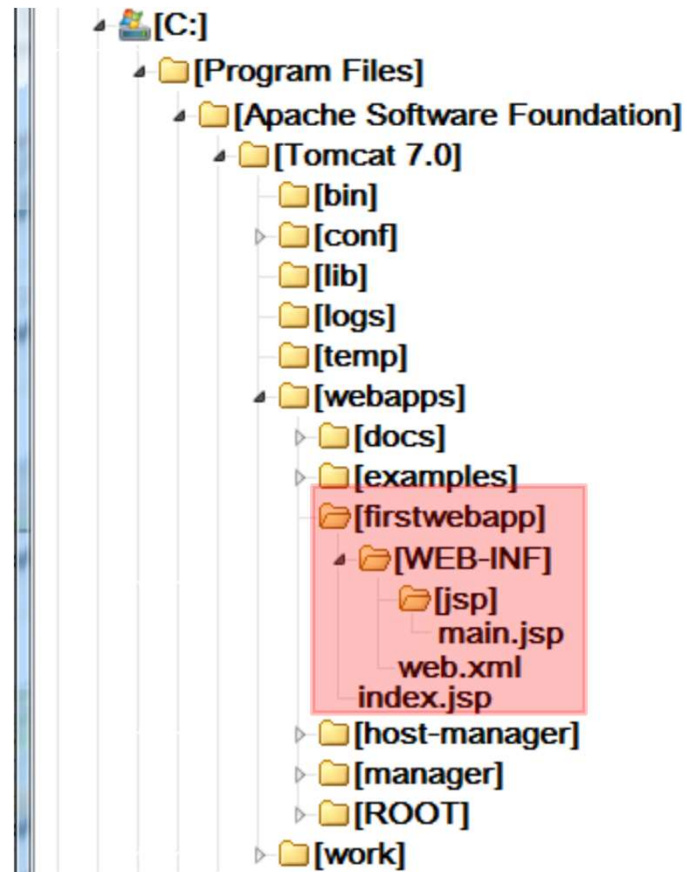
1 <html>
2 <head>
3   <meta http-equiv="Content-Type"
4     content="text/html; charset=utf-8">
5   <title>Insert title here</title>
6 </head>
7 <body>
8   <h2>
9     I am <i>main.jsp</i>
10  </h2>
11  <h1>
12    Go to
13    <a href="../index.jsp"> index.jsp </a>
14  </h1>
15 </body>
16 </html>

```

Результат:



Простое веб-приложение. Пример 3.




```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type"
4     content="text/html; charset=utf-8">
5   <title>Insert title here</title>
6 </head>
7 <body>
8   <h1>
9     Go to
10    <a href="main" >
11      main.jsp
12    </a>
13  </h1>
14 </body>
15 </html>

```

Содержание файлов index.jsp и main.jsp

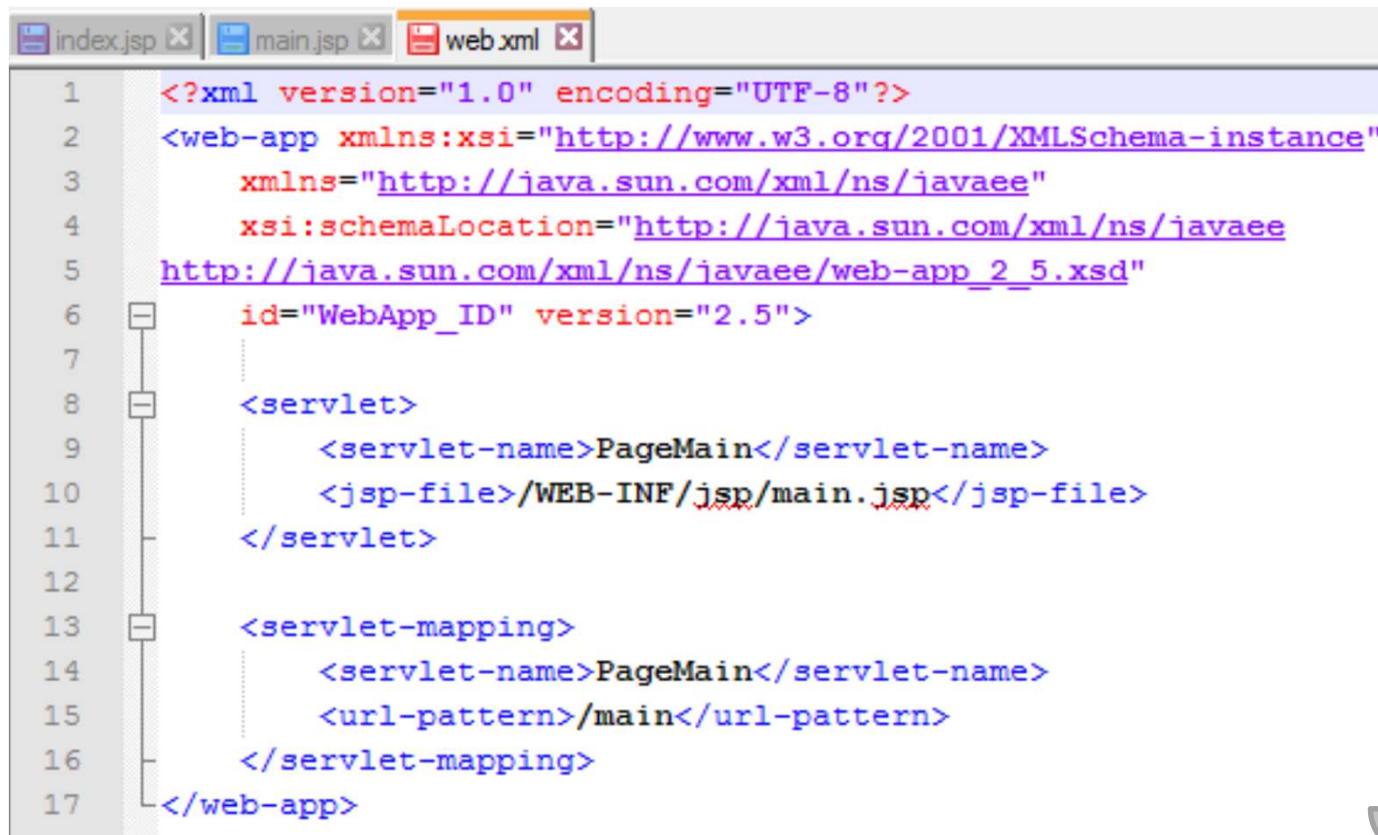
```

1 <html>
2 <head>
3   <meta http-equiv="Content-Type"
4     content="text/html; charset=utf-8">
5   <title>Insert title here</title>
6 </head>
7 <body>
8   <h2>
9     I am the <i>main.jsp</i>
10  </h2>
11  <h1>
12    Go to
13    <a href="index.jsp" >
14      index.jsp
15    </a>
16  </h1>
17 </body>
18 </html>

```



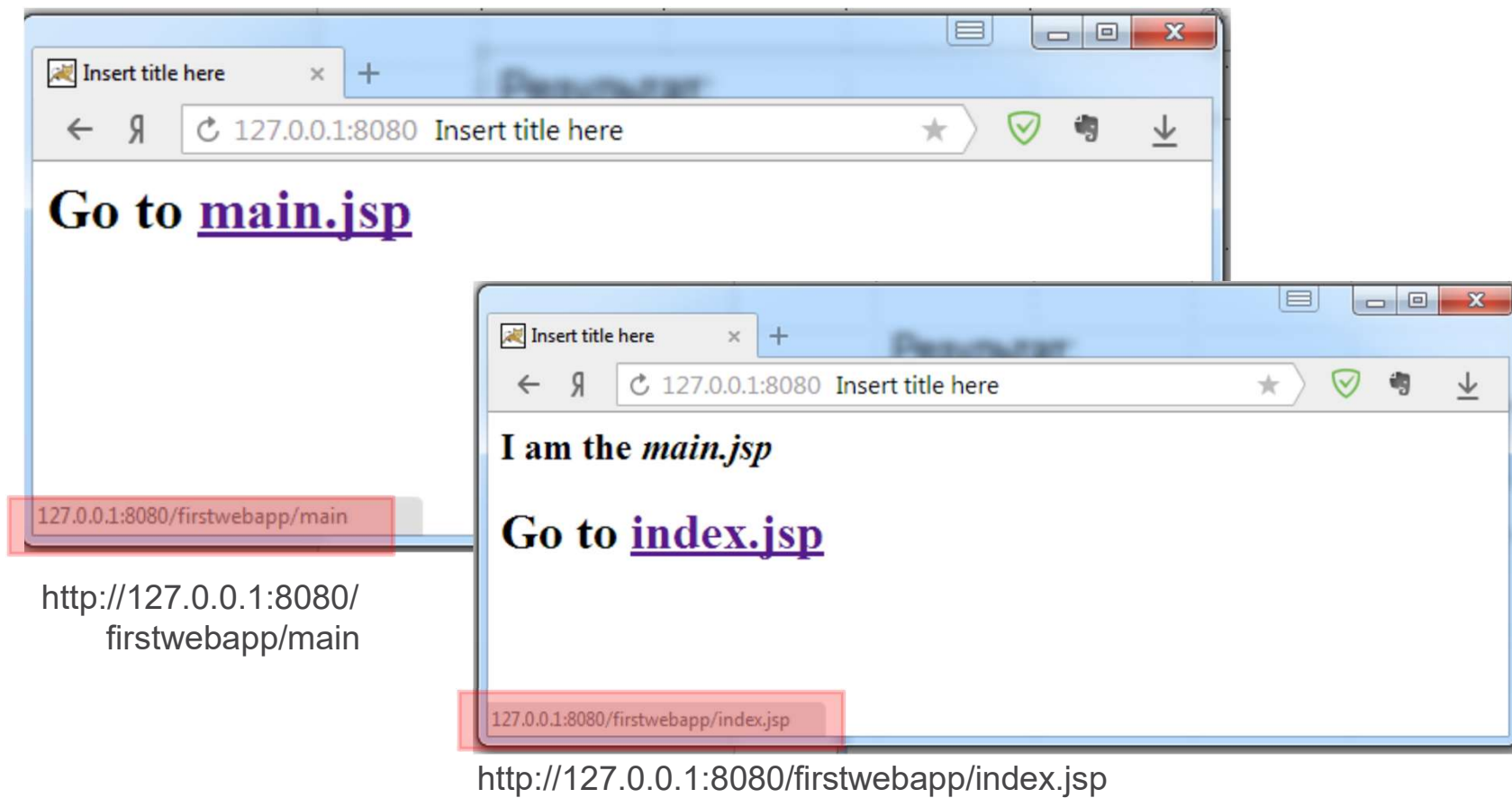
Содержание файла web.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3         xmlns="http://java.sun.com/xml/ns/javaee"
4         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5                             http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
6         id="WebApp_ID" version="2.5">
7     ...
8     <servlet>
9         <servlet-name>PageMain</servlet-name>
10        <jsp-file>/WEB-INF/jsp/main.jsp</jsp-file>
11    </servlet>
12
13    <servlet-mapping>
14        <servlet-name>PageMain</servlet-name>
15        <url-pattern>/main</url-pattern>
16    </servlet-mapping>
17 </web-app>
```



Результат:



Базовая структура веб-приложения должна включать **корневую директорию, WEB-INF директорию и дескриптор развертывания web.xml.**

Название корневого каталога будет частью URL, указывающего на один из содержащихся ресурсов, и используемый в качестве имени приложения.

Например, вызов *index.html* файла, расположенного в корне каталога '*mysite*', может быть сделан как:

<http://localhost:8080/mysite> или
<http://localhost:8080/mysite/index.html>

Web.xml конфигурационный файл используется для:

- Объявление Servlets и JSPs.
- Отображения Servlets и JSPs в URL шаблоны
- Определения welcom-страниц
- Установления безопасности содержимого, ролей и методов аутентификации.

Простое веб-приложение. Пример 4.



Содержание файла web.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5 http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
6     id="WebApp_ID" version="2.5">
7     ...
8     <servlet>
9         <servlet-name>Controller</servlet-name>
10        <servlet-class>myapp.Controller</servlet-class>
11    </servlet>
12
13    <servlet-mapping>
14        <servlet-name>Controller</servlet-name>
15        <url-pattern>/Controller</url-pattern>
16    </servlet-mapping>
17 </web-app>
```



Controller.java

```
package myapp;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Controller extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Controller() { super(); }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        processRequest();
    }

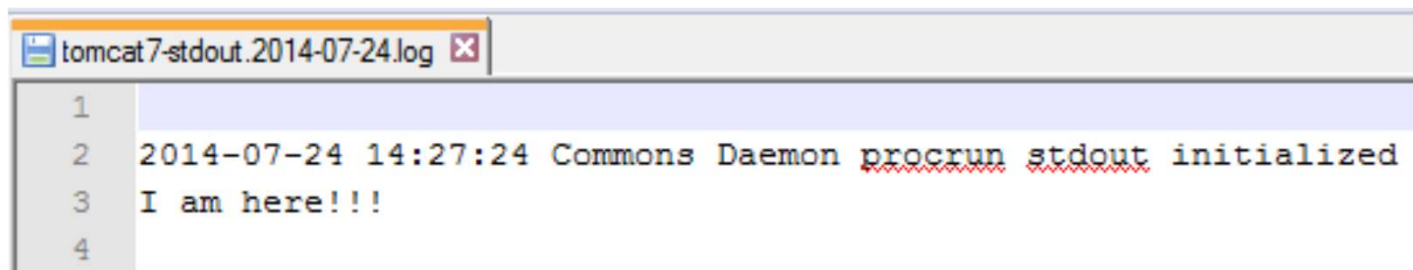
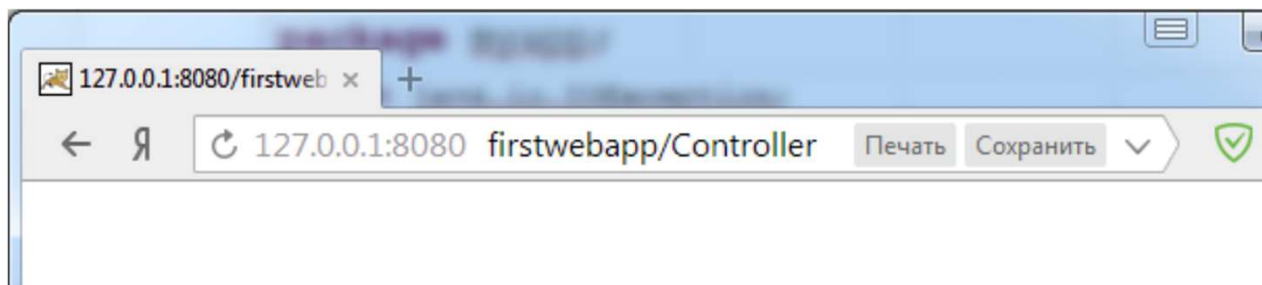
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        processRequest();
    }

    private void processRequest() {
        System.out.println("I am here!!!");
    }
}
```



Результат:

<http://127.0.0.1:8080/firstwebapp/Controller>



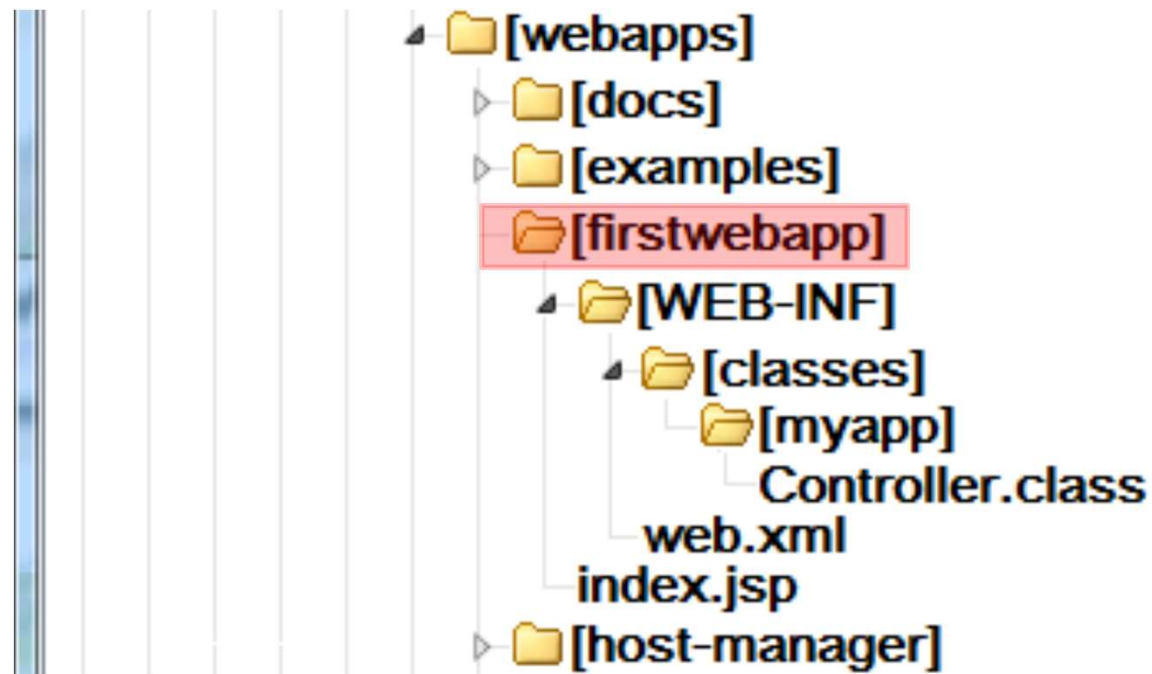
Любой класс, который загружен и выполнен в веб-контейнерах, должен быть расположен в **WEB-INF\classes**.

Это могут быть:

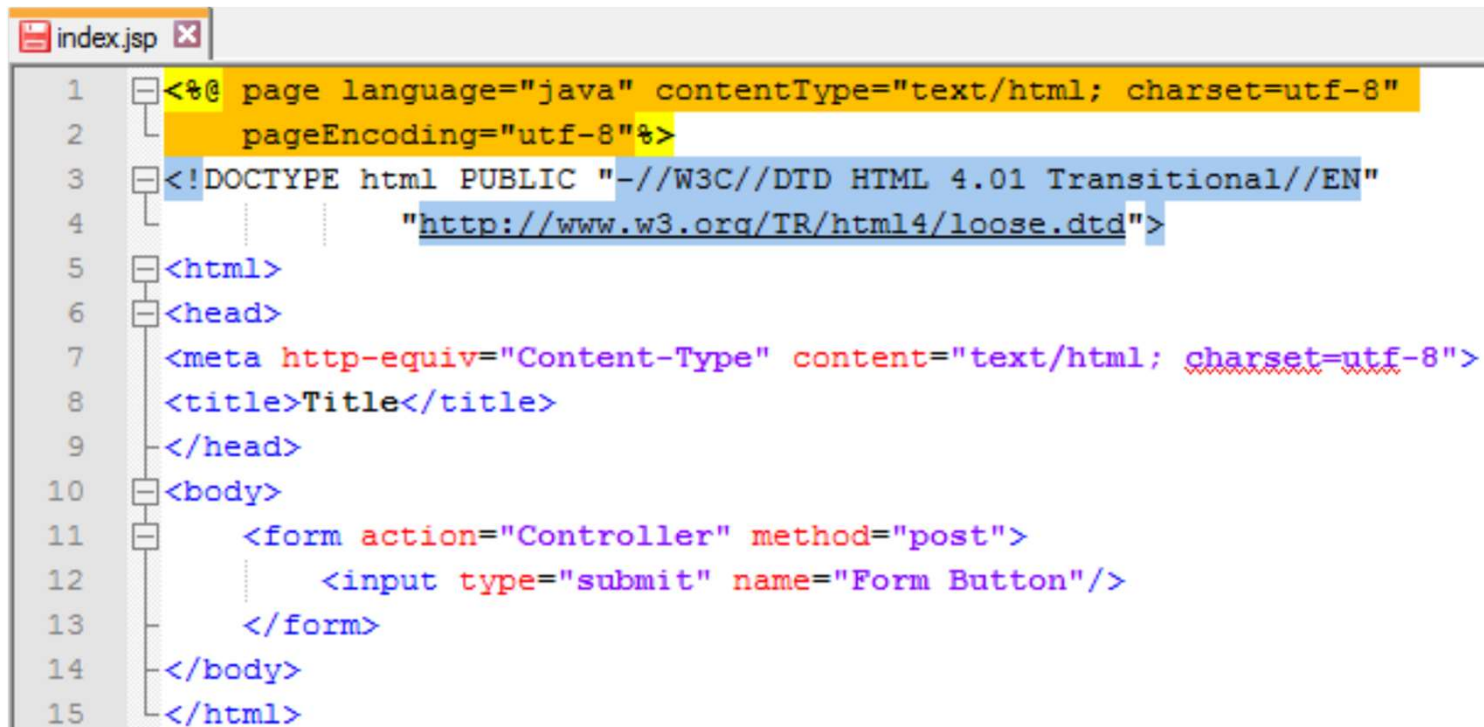
- Servlets
- Java Beans (used in JSP)
- Tag libraries classes (used in JSP)
- Helper classes

Другие файлы как JSPs и статическое содержание могут быть расположены где угодно в соответствии с корневой директорией.

Простое веб-приложение. Пример 5.



Содержание файла index.jsp



```
1  <%@ page language="java" contentType="text/html; charset=utf-8"
2     pageEncoding="utf-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4     "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
8     <title>Title</title>
9  </head>
10 <body>
11     <form action="Controller" method="post">
12         <input type="submit" name="Form Button"/>
13     </form>
14 </body>
15 </html>
```

Содержание файла web.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
6     id="WebApp_ID" version="2.5">
7     ...
8     <servlet>
9         <servlet-name>Controller</servlet-name>
10        <servlet-class>myapp.Controller</servlet-class>
11    </servlet>
12
13    <servlet-mapping>
14        <servlet-name>Controller</servlet-name>
15        <url-pattern>/Controller</url-pattern>
16    </servlet-mapping>
17 </web-app>
```




Controller.java

```
package myapp;
import ...
public class Controller extends HttpServlet {
    private static final long serialVersionUID = 1L;

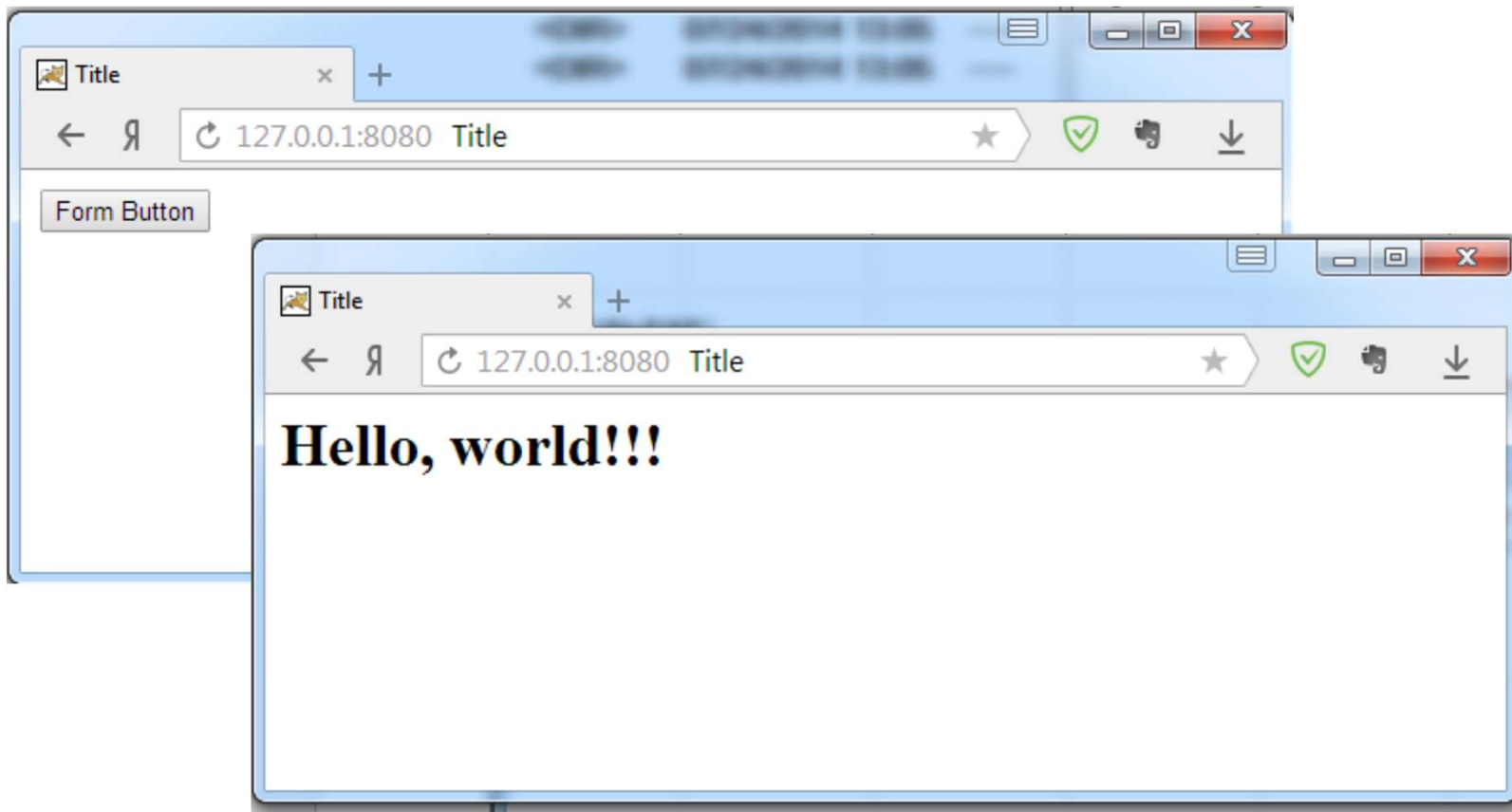
    public Controller() {        super();    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {processRequest(request, response);    }
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {processRequest(request, response);    }

    private void processRequest(HttpServletRequest request,
HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head>");
        out.println("<meta http-equiv=\"Content-Type\"
content=\"text/html; charset=utf-8\">");
        out.println("<title>Title</title>");
        out.println("</head><body>");
        out.println("<h1> Hello, world!!! </h1>");
        out.println("</body></html>");

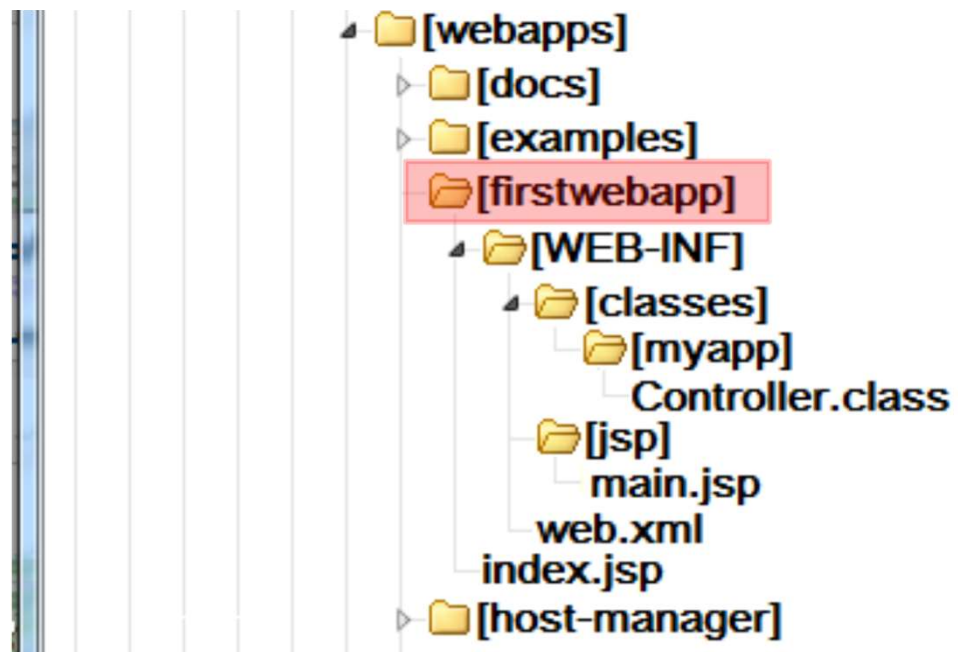
    }
}
```



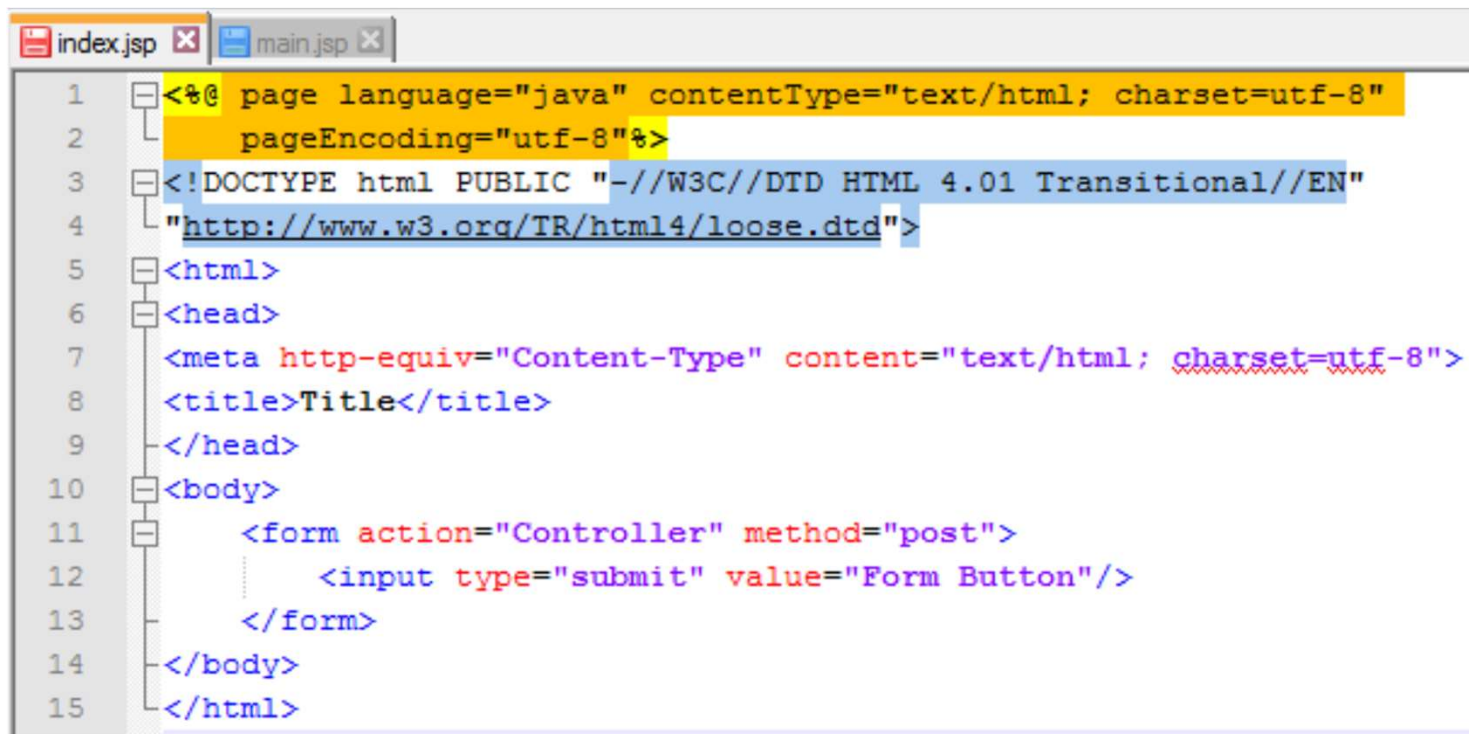
Результат:



Простое веб-приложение. Пример 6.



Содержание файла index.jsp



```
1 <%@ page language="java" contentType="text/html; charset=utf-8"
2   pageEncoding="utf-8"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4   "http://www.w3.org/TR/html4/loose.dtd">
5 <html>
6 <head>
7   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
8   <title>Title</title>
9 </head>
10 <body>
11   <form action="Controller" method="post">
12     <input type="submit" value="Form Button"/>
13   </form>
14 </body>
15 </html>
```



Содержание файла main.jsp

```
index.jsp x main.jsp x
1  <%@ page language="java" contentType="text/html; charset=utf-8"
2     pageEncoding="utf-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4     "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
8     <title>Insert title here</title>
9  </head>
10 <body>
11     <h1>Hello, world!!!</h1>
12     <form action="index.jsp" method="post">
13         <input type="submit" value="Go To Index Page" />
14     </form>
15 </body>
16 </html>
```



Controller.java

```
package myapp;
import ...
public class Controller extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public Controller() {        super();        }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {processRequest(request, response);    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {processRequest(request, response);    }

    private void processRequest(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException {

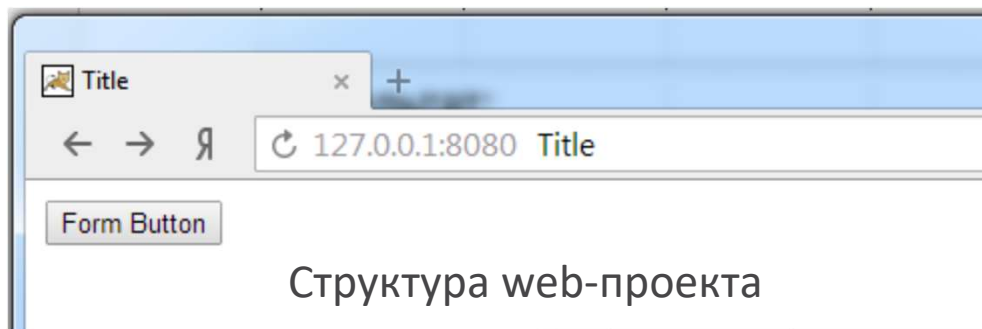
        RequestDispatcher requestDispatcher =
            request.getRequestDispatcher("/WEB-INF/jsp/main.jsp");

            requestDispatcher.forward(request, response);

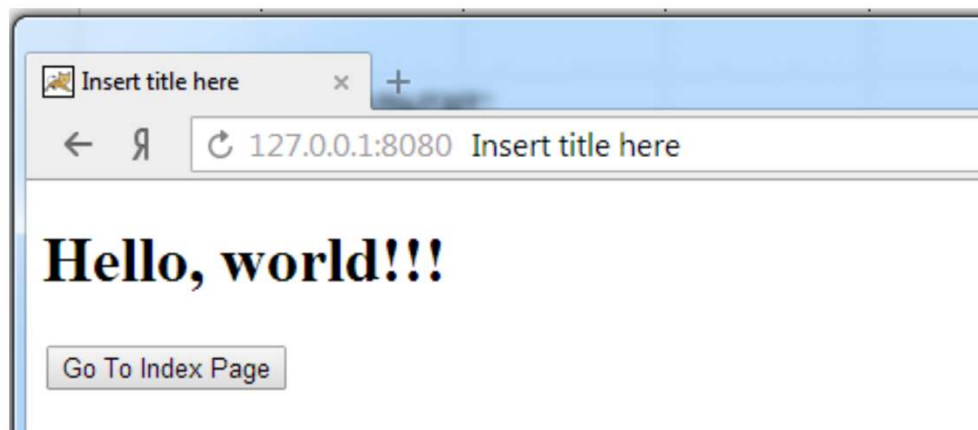
        }
    }
}
```



Результат:



Структура web-проекта



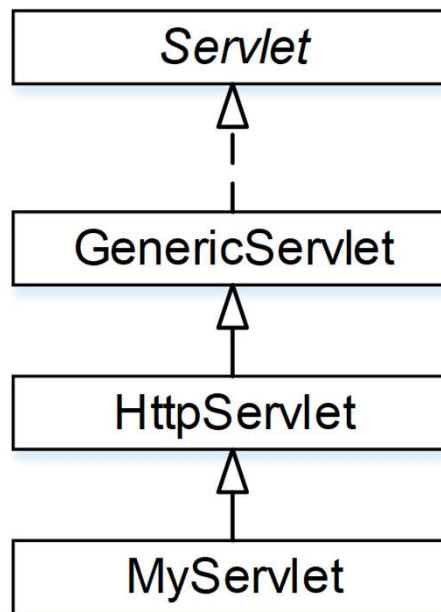
JAVAX.SERVLET

Сервлеты – это компоненты приложений Java Enterprise Edition, выполняющиеся на стороне сервера, способные обрабатывать клиентские запросы и динамически генерировать ответы на них.

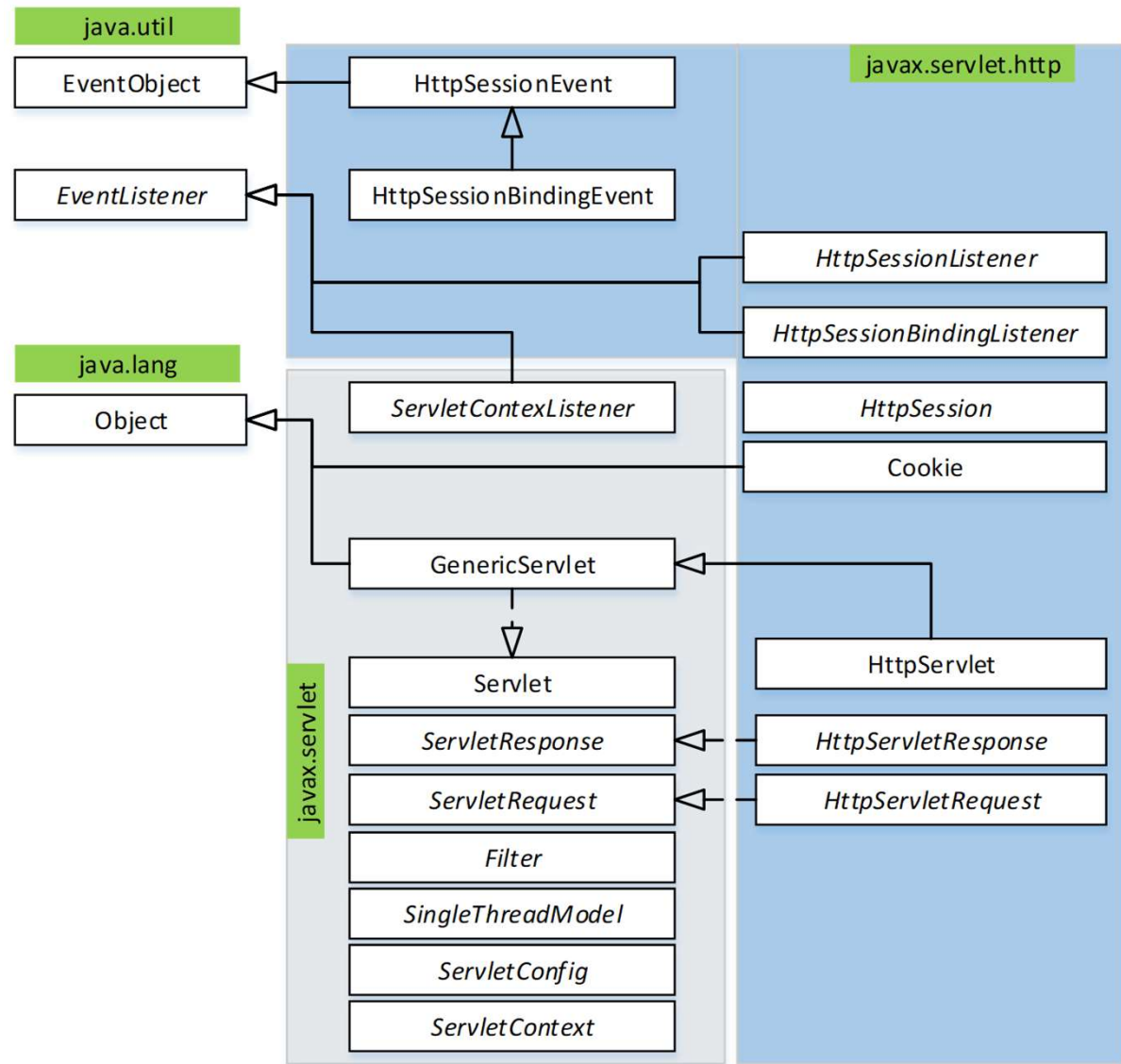
Наибольшее распространение получили сервлеты, обрабатывающие клиентские запросы по протоколу HTTP.

SERVLET

Пакет `javax.servlet` обеспечивает интерфейсы и классы для написания сервлетов.



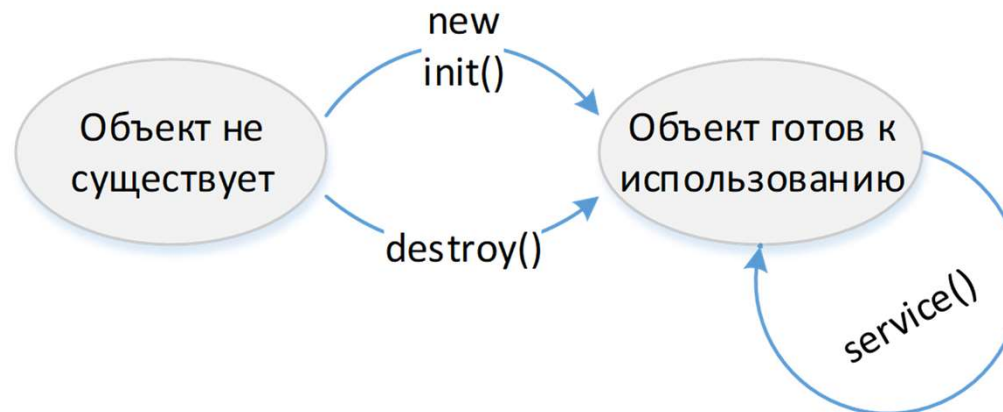
Servlets API



SERVLET LIFE CIRCLE

Жизненный цикл сервлета включает следующие шаги (фазы):

- Загрузка класса
- Создание экземпляра
- Вызов метода `init`
- Вызов метода `service` для каждого запроса
- Вызов метода `destroy`



SERVLETREQUEST, SERVLETRESPONSE

ServletRequest и **ServletResponse** –интерфейсы, определенные в пакете **javax.servlet**.

ServletRequest	связь клиента с сервером	Предоставляет доступ к именам параметров, переданных клиентом, именам удаленного хоста, создавшего запрос и сервера который их получает и др. Входному потоку ServletInputStream для получения данных от клиентов.
ServletResponse	обратная связь сервлета с клиентом	Позволяет сервлету устанавливать длину содержания и тип MIME ответа. Обеспечивает исходящий поток ServletOutputStream и Writer , через которые сервлет может отправлять ответные данные.

Объект **HttpServletRequest** предоставляет доступ к данным HTTP заголовка и позволяют получить аргументы, которые клиент направил вместе с запросом.

Метод	Описание
String getParameter(String name)	Возвращает значение именованного параметра
String[] getParameterValues(String name)	Возвращает массив значений именованного параметра
Enumeration getParameterNames()	Возвращает enumeration, содержащий все имена параметров запроса
String getQueryString()	Возвращает строковую величину необработанных данных клиента для HTTP запросов GET
BufferedReader getReader()	возвращает объект BufferedReader (text) для считывания необработанных данных (для HTTP запросов POST, PUT, и DELETE)
ServletInputStream getInputStream()	возвращает объект ServletInputStream (binary) для считывания необработанных данных (для HTTP запросов для POST, PUT, и DELETE)

Методы `HttpServletRequest`.

```
boolean authenticate(HttpServletRequest response)
String getAuthType()
Cookie[] getCookies()
String getHeader(String name)
Enumeration<String> getHeaders(String name)
int getIntHeader(String name)
Part getPart(String name)
String getPathInfo()
String getQueryString()
String getRequestedSessionId()
StringBuffer getRequestURL()
HttpSession getSession()
java.security.Principal getUserPrincipal()
boolean isRequestedSessionIdFromCookie()
boolean isRequestedSessionIdFromURL()
boolean isRequestedSessionIdValid()
boolean isUserInRole(java.lang.String role)
void login(String username, String password)
void logout()

String getContextPath()
long getDateHeader(name)
Enumeration<String> getHeaderNames()
String getMethod()
Collection<Part> getParts()
String getPathTranslated()
String getRemoteUser()
String getRequestURI()
String getServletPath()
HttpSession getSession(boolean create)
```

Объект **HttpServletResponse** позволяет вернуть данные (ответ) пользователю.

Метод	Описание
getWriter	Возвращает объект класса <code>Writer</code> (text) для формирования ответа
getOutputStream	Возвращает объект класса <code>ServletOutputStream</code> (binary) для формирования ответа
void sendError(int sc, String msg)	Сообщение о возникших ошибках, где <code>sc</code> – код ошибки, <code>msg</code> – текстовое сообщение
void setDateHeader(String name, long date)	Добавление даты в заголовок ответа
void setHeader(String name, String value)	Добавление параметров в заголовок ответа. Если параметр с таким именем уже существует, то он будет заменен
setContentType(String type)	Установка тип содержимого (Content-type)

Методы `HttpServletResponse`.

```
void addCookie(Cookie cookie)
void addDateHeader(String name, long date)
void addHeader(String name, String value)
void addIntHeader(String name, int value)
boolean containsHeader(String name)
String encodeRedirectURL(String url)
String encodeURL(String url)
String getHeader(String name)
Collection<String> getHeaderNames()
Collection<String> getHeaders(String name)
int getStatus()
void sendError(int sc)
void sendError(int sc, String msg)
void sendRedirect(String location)
void setDateHeader(String name, long date)
void setHeader(String name, String value)
void setIntHeader(String name, int value)
void setStatus(int sc)
```


index.jsp

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>MyJSP</title>
</head>
<body>
    <form action="MyServlet" method="get">
        <input type="hidden" name="command" value="forward" />
        Enter login:<br/>
        <input type="text" name="login" value="" /><br/>
        Enter password:<br/>
        <input type="password" name="password" value="" /><br/>
        <input type="submit" value="Отправить" /><br/>
    </form>
</body>
</html>
```



Controller.java


```
package _java._ee._01._reqparam;
import ...
public class Controller extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html");

        String login = request.getParameter("login");
        String password = request.getParameter("password");

        PrintWriter out = response.getWriter();
        out.println("Your login: " + login);
        out.println("<br />Your password: " + password);
    }
}
```

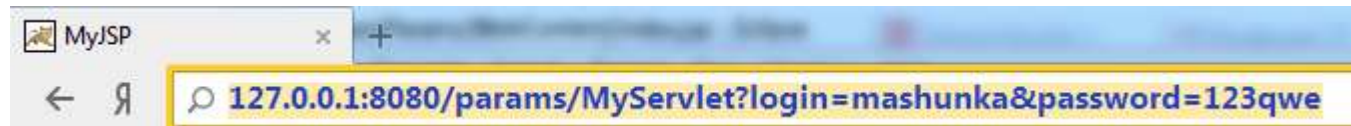


web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ...>
    ...
    <servlet>
        <servlet-name>Controller</servlet-name>
        <servlet-class>_java._ee._01._reqparam.Controller
        </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Controller</servlet-name>
        <url-pattern>/MyServlet</url-pattern>
    </servlet-mapping>
</web-app>
```



Запрос:



ИЛИ

`http://127.0.0.1:8080/params/MyServlet?command=forward&login=mashunka&password=123qwe`



Ответ:

Your login: mashunka
Your password: 123qwe

ИЛИ



Your login: mashunka
Your password: 123qwe

WEB.XML

web.xml - дескриптор развертывания приложения.

Элементы **web.xml**

<servlet> - блок, описывающий сервлеты

<display-name> - название сервлета

<description> - текстовое описание сервлета

<servlet-name> - имя сервлета

<servlet-class> - класс сервлета

<init-param> - блок, описывающий параметры инициализации сервлета

<param-name> - название параметра

<param-value> - значение параметра

<servlet-mapping> - блок, описывающий соответствие url и запускаемого сервлета

<servlet-name> - имя сервлета

<url-pattern> - описывает url-шаблон

<session-config> - блок, описывающий параметры сессии

<session-timeout> - максимальное время жизни сессии

<login-config> - блок, описывающий параметры, как пользователь будет логиниться к серверу

<auth-method> - метод авторизации (BASIC, FORM, DIGEST, CLIENT-CERT)

<welcome-file-list> - блок, описывающий имена файлов, которые будут пытаться открыться при запросе только по имени директории (без названия файла). Сервер будет искать первый существующий файл из списка и загрузит именно его

<welcome-file> - имя файла

<error-page> - блок, описывающий соответствие ошибки и загружаемой при этом страницы

<error-code> - код произошедшей ошибки

<exception-type> - тип произошедшей ошибки

<location> - загружаемый файл

<taglib> - блок, описывающий соответствие JSP Tag library descriptor с URI-шаблоном

<taglib-uri> - название uri-шаблона

<taglib-location> - расположение шаблона

Servlet 3.0 позволяет регистрировать сервлеты, фильтры и слушателей не с помощью дескриптора развертывания web.xml, а с помощью аннотаций.

- @WebServlet,
- @WebFilter
- @WebListener

Сервлет-контейнер отвечает за обработку аннотированных классов расположенных в каталоге WEB-INF/classes, в архивах из каталога WEB-INF/lib, или где-либо еще в classpath.

.WAR

Web-приложение поставляется в виде архива **.war**, содержащего все его файлы.

Содержащаяся в этом архиве структура директорий Web-приложения должна включать директорию WEB-INF, вложенную непосредственно в корневую директорию приложения.

Директория **WEB-INF** содержит две поддиректории —

- **/classes** для .class-файлов сервлетов, классов и интерфейсов EJB-компонентов и других Java-классов, и
- **/lib** для .jar и .zip файлов, содержащих используемые библиотеки.

Файл **web.xml** также должен находиться непосредственно в директории **WEB-INF**.

War-файл необходимо разместить в папке **/webapps** контейнера Tomcat

КОНТЕЙНЕР СЕРВЛЕТОВ ТОМСАТ

Структура Tomcat

/Apache Software Foundation/Tomcat...

Подкаталоги:

- **/bin** – содержит файлы запуска контейнера сервлетов **tomcatX.exe**, **tomcatXw.exe** и некоторые необходимые для этого библиотеки;
- **/common** – содержит библиотеки служебных классов, в частности Servlet API;
- **/conf** – содержит конфигурационные файлы, в частности конфигурационный файл контейнера сервлетов **server.xml**;
- **/logs** – помещаются log-файлы;
- **/lib** – размещены необходимые библиотеки;
- **/webapps** – в этот каталог помещаются приложения (в отдельный каталог)



СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Java for students

Филиал кафедры ПОИТ БГУИР в Eram Systems
курс: Веб-технологии (JAVA)

Author: Olga Smolyakova , PhD
Oracle Certified Java 6 Programmer
Olga_Smolyakova@epam.com