Napravljena je web aplikacija prema maven archetype za web aplikacije

D:\MAVEN\hf>**mvn archetype:generate -DarchetypeArtifactId=maven-archetype-webapp**

sa sledecim podesavanjima

groupId: **com.hf**

artifactId: **ch8-jsp**

version: **1.0-SNAPSHOT**

package: **war**

**pom.xml**

**ubacen tomcat plugin**

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

</plugin>

</plugins>

**ubacen dependency za servlete**

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

**web.xml**

Ubačeni su sledeći elementi:

servlet i servlet-mapping za klasu koja je servlet a to je ListenerTester

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

<web-app version="2.5"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd">

<display-name>Archetype Created Web Application</display-name>

<servlet>

<servlet-name>**servlet**</servlet-name>

<servlet-class>**Servlet**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**servlet**</servlet-name>

<url-pattern>**/usernameTest.do**</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

**Napravljen je folder src/main/java/. Tu je stavljena klasa Servlet.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class **Servlet** extends HttpServlet {

protected void doPost(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException {

**String name = request.getParameter("username");**

**request.setAttribute("name", name);**

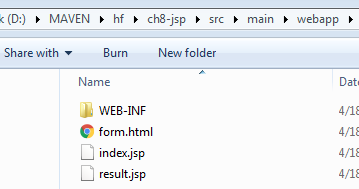
RequestDispatcher view = request.getRequestDispatcher("**/result.jsp**");

view.forward(request, response);

}

}

Servelt uzima parametar username iz forme, postavlja ga kao atribut u request i salje request view-u result.jsp. Result.jsp ce izvaditi taj atribut iz requesta i prikazati ga.



**form.html**

<html><body>

<form method="POST" action="**usernameTest.do**">

Username: <input type="text" **name="username"**><br/>

<input type="submit" value="posalji">

</form>

</body></html>

**result.jsp**

<html><body>

Hello

<%= **request.getAttribute("name")** %>

</body></html>

**index.jsp**

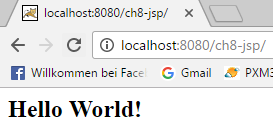
<html>

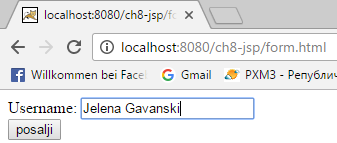
<body>

<h2>Hello World!</h2>

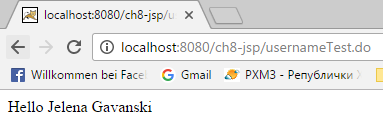
</body>

</html>





Klikom na dugme posalji dobija se



## But what if the attribute is not a String, but an instance of Person?

Sad sam premuljala projekat tako sto sam Person klasu ubacila u paket foo

**Person.java**

**package foo;**

public class **Person** {

private String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**U Servletu ubaceno**

**Person p** = new Person();

p.setName("Evan");

**request.setAttribute("person", p);**

**result.jsp**

<!--

<html><body>

<% foo.Person p = (foo.Person)**request.getAttribute("person")**; %>

Person is: <%= p.getName() %>

Hello

<%= request.getAttribute("name") %>

</body></html>

-->

isto sto i

<html><body>

<%@ page import="foo.\*" %>

<% Person p = (Person)**request.getAttribute("person");** %>

Person is: <%= p.getName() %>

Hello

<%= request.getAttribute("name") %>

</body></html>

## Person is java bean so we will use the bean-related standard actions

Zelimo da eliminisemo scripting.

Objekat person je atribut u request scopu.

Objekat person ima property name.

There is only one syntax for the Action element, as it conforms to the XML standard:

<jsp:action\_name attribute="value" />

Action elements are basically predefined functions and there are following JSP actions available:

<https://www.tutorialspoint.com/jsp/jsp_actions.htm>

<html><body>

**<jsp:useBean id="person" class="foo.Person" scope="request" />**

**Person created by servlet: <jsp:getProperty name="person" property="name" />**

Hello

<%= request.getAttribute("name") %>

</body></html>

**<jsp:useBean id="person" class="foo.Person" scope="request" />**

**jsp:useBean** standardna akcija

**id="person"** ono sto je setovano u servletu sa request.setAttribute("person", p);

**class="foo.Person"** klasa beana

**scope="request"** scope atributa je request

**<jsp:getProperty name="person" property="name" />**

**jsp:getProperty** standardna akcija

**name="person"** identifikuje bean, slaze se sa id u jsp:useBean

**property="name"** identifikuje ime propertija (ime propertija u klasi Person)

**result.jsp**

<html><body>

<jsp:useBean **id="person"** class="foo.Person" scope="request" />

Person created by servlet: <jsp:getProperty **name="person"** property="name" />

<br/>

<jsp:useBean **id="person1"** class="foo.Person" scope="page" >

<jsp:setProperty **name="person1"** property="name" value="Fred" />

</jsp:useBean>

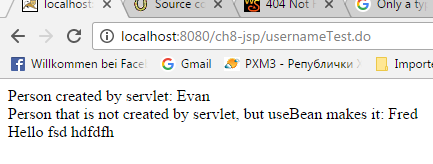
Person that is not created by servlet, but useBean makes it: <jsp:getProperty **name="person1"** property="name" />

<br/>

Hello

<%= request.getAttribute("name") %>

</body></html>



Things you need to know for using beans with JSP and servlets are these few rules (we’re showing only those that apply to what we’re doing with servlets and JSPs):

1) You MUST have a public, no-arg constructor.

2) You MUST name your public getter and setter methods starting with “get” (or “is”, for a boolean) and “set”, followed by the same word. (getFoo(), setFoo()). The property name is derived from stripping off the “get” and “set”, and changing the first character of what’s left to lowercase.

3) The setter argument type and the getter return type MUST be identical. This defines the property type.

int getFoo() void setFoo(int foo)

4) The property name and type are derived from the getters and setters and NOT from a member in the class. For example, just because you have a private int foo variable does NOT mean a thing in terms of properties. You can name your variables whatever you like. The “foo” property name comes from the methods. In other words, you have a property simply because you have a getter and setter. How you implement them is up to you.

5) For use with JSPs, the property type SHOULD be a type that is either a String or a primitive. If it isn’t, it can still be a legal bean, but you won’t be able to rely only on standard actions, and you might have to use scripting.