Queen Mary University of London

School of Electronic Engineering and Computer Science

Lab 4 – *JavaBeans* and JSPs

Note: This lab follows the work you have done in the previous lab. Therefore, before starting this lab, please ensure that you have done as much as possible of "Lab 3 – JavaScript and *Servlets*".

1. *Servlet* forwarding and introduction to *JavaBeans*

a) Rewrite the *servlet* **`HoroscopeServlet`**, so that no output is generated and sent back to the client browser. Instead, it finds the horoscope from the information provided by the client, and then forwards that to another *servlet* called **`HoroscopeToAnotherServlet`** providing the name and the horoscope in the request object, which then outputs the information back to the client browser.

b) Write a *JavaBean* called **`PersonalDetailsBean`** that holds the name, gender (female, male) and Zodiac sign of a person. The person's gender is held as a **`boolean`**, the sign of the Zodiac is held as an **`int`** between **`1`** and **`12`** inclusive, and the first name is held as a **`String`**.

c) **Questions**:

- Describe the notion of *servlet redirection* and indicate how it differs from *servlet forwarding*.
- When writing *JavaBeans*, it must be ensured that they can be transmitted across a network. How is this requirement satisfied when writing *JavaBeans*?

2. Session Handling

a) Write HTML corresponding to the screenshot in **Figure 1** that invokes the *servlet* called **`SimpleHoroscopeServlet`**.

- **`SimpleHoroscopeServlet`** reads the client's input. If this is the client's first visit to the site, the *servlet* remembers the name and Zodiac sign in a new instance of **`PersonalDetailsBean`** that is stored in the *session* object and responds with a request for the gender of the user.
- When the client responds with the user's gender (again to the same *servlet*), then the *bean* is retrieved and the horoscope is given according to the rules used previously.

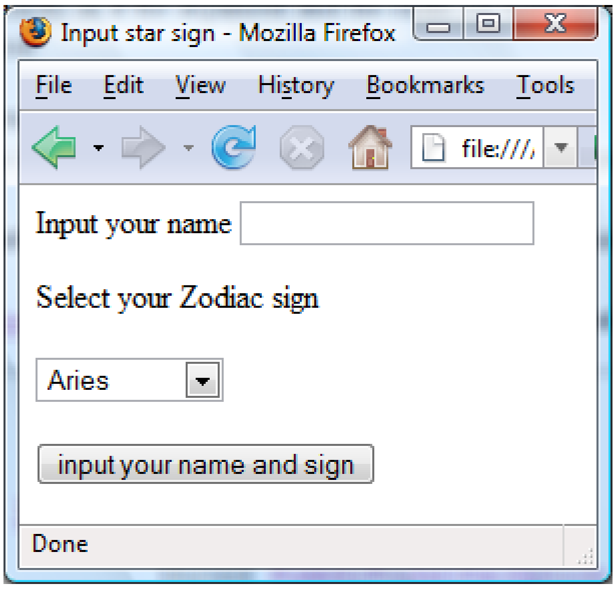

Figure 1

b) **Question**:

- What is a *session*, in the context of web applications?

3. JSPs

a) Instead of outputting the horoscope directly (as in **Part 2** of this lab), the *servlet* forwards to a JSP titled **`output.jsp`**. The JSP extracts the person's details from the **`PersonalDetailsBean`** and outputs the horoscope according to the same rules. Show the changes to the *servlet* i.e. write **`SimpleHoroscopeServlet2`** and write this JSP.

b) **Question**:

- What is the main difference between JSPs and *servlets*?

Note: The next page contains a *list of typical errors* you might encounter when developing web applications, together with some suggestions as to the cause of the errors and thus, how to fix them.

END of Lab 4: "*JavaBeans* and JSPs"

Appendix: Typical Error Messages

Error 1: `error: cannot read: src\com\example\web\PhoneSelect.java`

Possible causes of error:

a) Your java file has a different name (e.g. **`Phone.java`**). This is common in web apps where you have one name for the HTML files, plus concealed names due to *servlet* mapping.
b) You are compiling in the wrong directory. The compiler will look in the current directory for the directory **`src`**. In the standard deployment model the **`src`** directory will be inside the **`WEB-INF`** directory.

Error 2:

```
cannot find symbol
symbol: class Iterator/List
```

Your class does not **`import java.util.*;`**.

Error 3:

```
cannot find symbol
symbol: class NameOfBusinessClass (e.g., PhoneExpert.java)
```

Your class does not import the package, e.g., **`import com.example.model.*;`**.

Error 4:

```
cannot find symbol
symbol: class HttpServlet
```

This class is inside the **`servlet-api.jar`** provided with Tomcat. Possible causes of error:

a) You have not installed Tomcat.
b) There is a typo (typing/spelling error) in your **`classpath`** instructions.
c) You have not provided the complete path in your **`classpath`** instructions. For example, on a home computer this may look like:
   `C:\Program Files\Apache Software Foundation\Tomcat 6.0\lib\servlet-api.jar`
   or
   `C:\Program Files\Apache Software Foundation\Tomcat 6.0\lib\servlet-api.jar;classes;.`

Error 5: `Your browser cannot find your Servlet.do file`

Possible causes of error:

a) Incorrect class path provided in the HTML file. It should be the full local path e.g., **`localhost:8080/Phone/SelectPhone.do`**.
b) There is a typo in your deployment descriptor.

To test the validity of your deployment descriptor, double click on the XML file. This should open in a browser and tell you if there are any syntax errors.