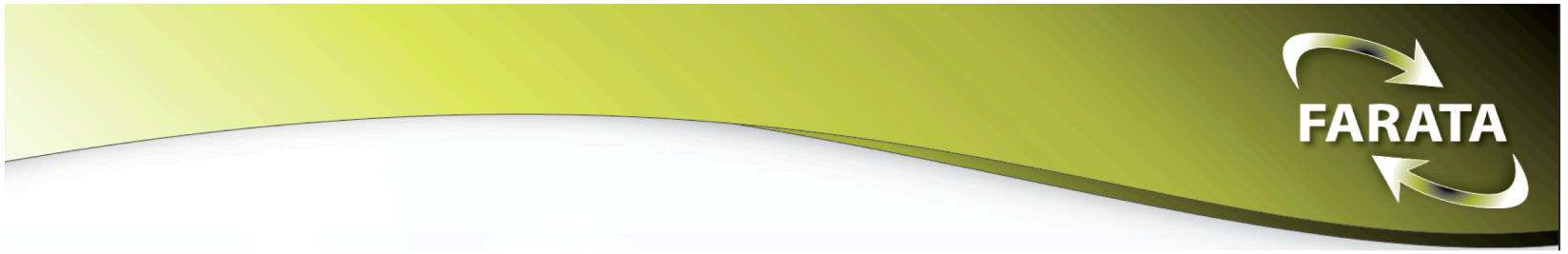


Java Programming
Unit 14

Java EE Overview
Servlets

Java EE, JCP, JSR

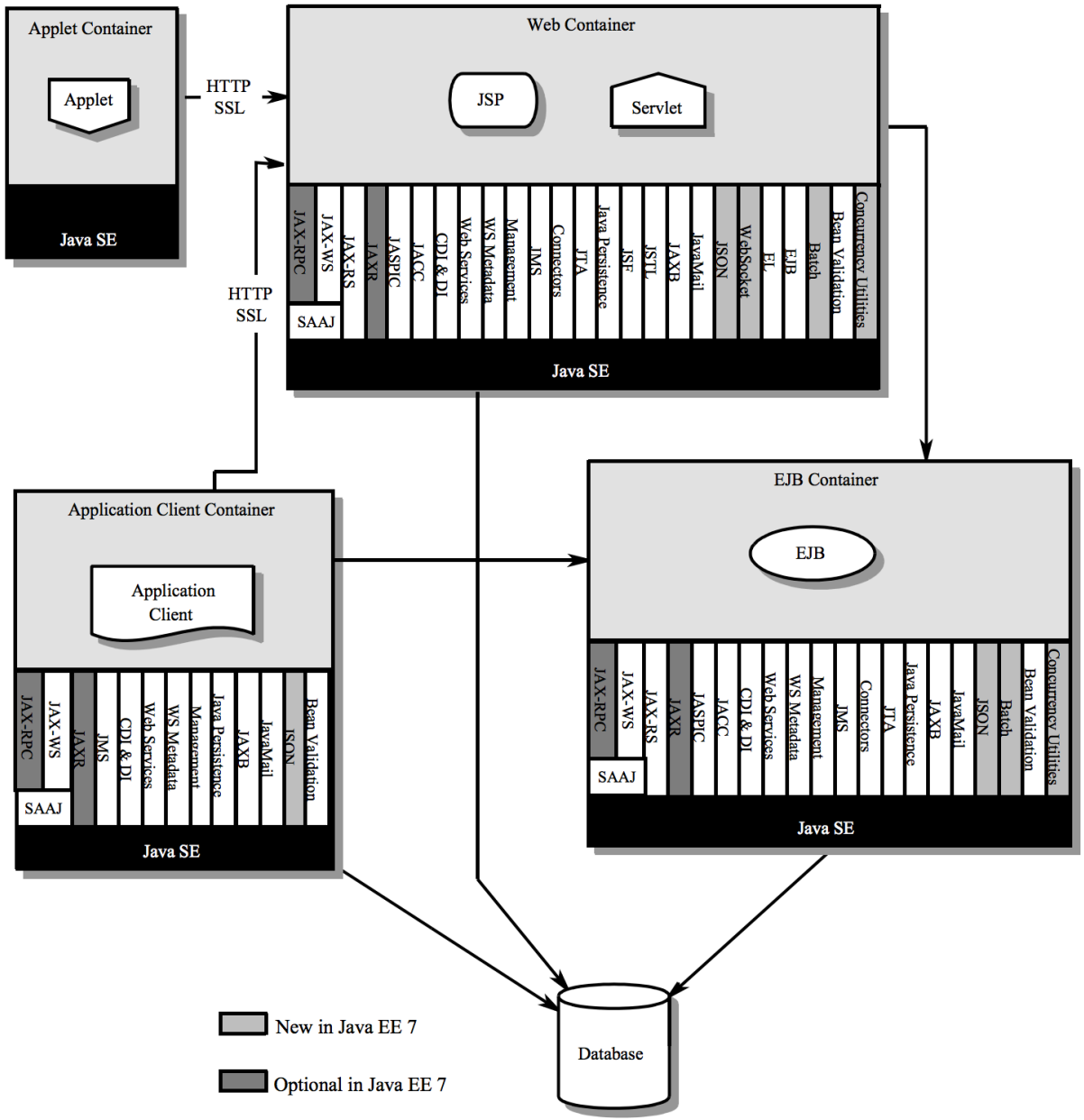
- Java Community Process – a mechanism for developing standard specs: https://www.jcp.org/en/home/index

- Java Specification Request – is a proposed and potentially approved specification for a concrete technology, e.g. JSR 340 – is a specification for Servlets 3.1

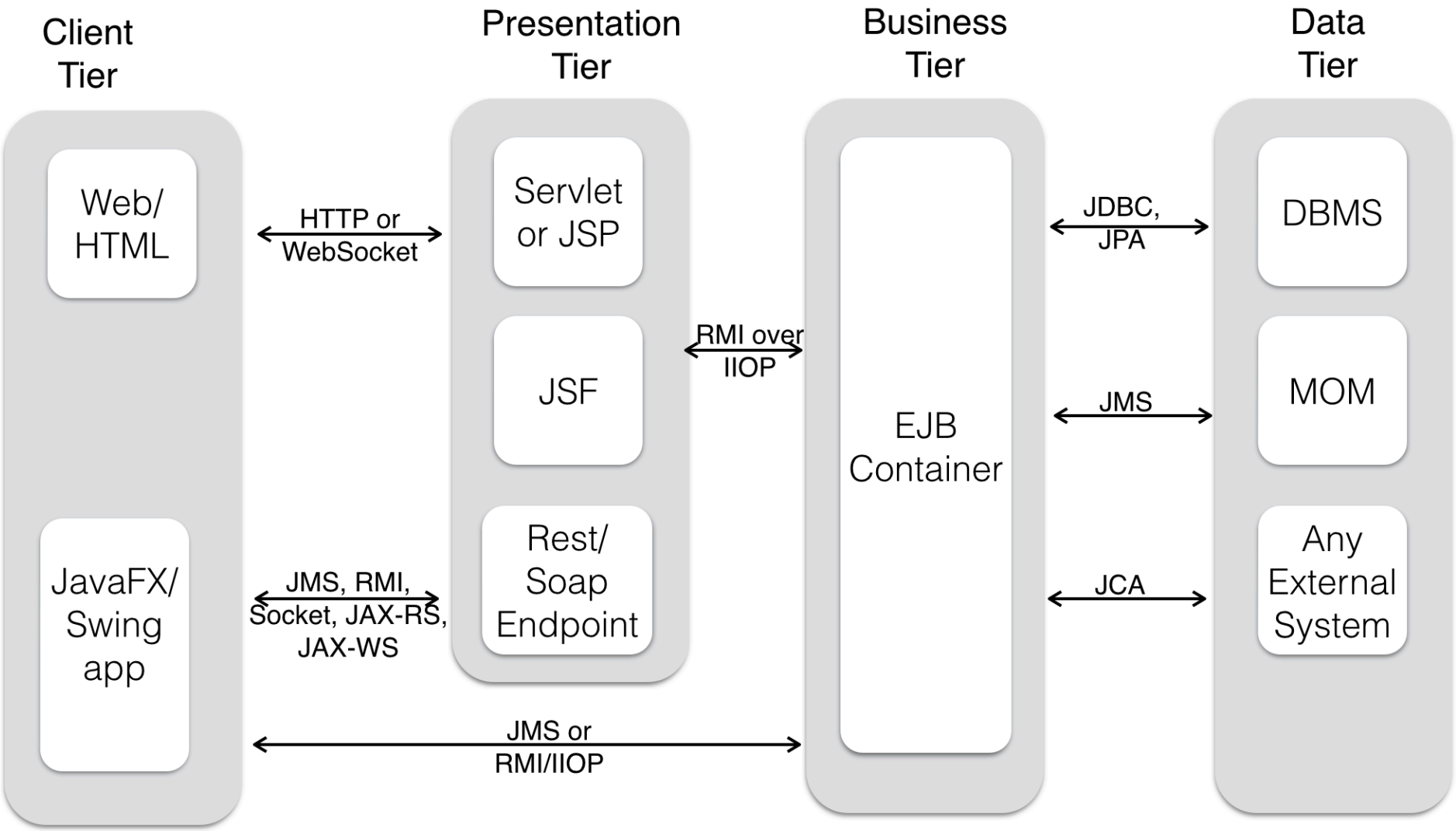
- Java Enterprise Edition is a collection of approved and released JSRs

- Java EE 7 is JSR-242: https://www.jcp.org/aboutJava/communityprocess/final/jsr342

A fragment from JSR-342



Architecting Java EE Applications



Java EE 7 major additions

- Released in 2013
- Simplified JMS and Restful APIs
- Added Java API for JSON
- Added WebSockets support
- Allows using multi-threading in Java EE containers
- Java EE 7 Tutorial is here: http://docs.oracle.com/javaee/7/tutorial/doc/

Java EE Application Servers

- 18 Java application servers support Java EE 6

- As of May 1, 2014 two application servers fully support Java EE 7:

  - GlassFish from Oracle
  - WildFly from Red Hat

For current Java EE compatibility see http://goo.gl/qjqa3

Walkthrough 1 (Installing GlassFish 4)

- Download and unzip GlassFish-4.0.zip (quick start) from https://glassfish.java.net/download.html
- In the Command (or Terminal) Window switch to the directory *glassfish4/bin* and run *./asadmin start-domain domain1*. Windows users should run asadmin.bat start-domain domain1.

```
Yakov:bin yfain11$ ./asadmin start-domain domain1
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain  Location: /Users/yfain11/glassfish4/glassfish/domains/domain1
Log File: /Users/yfain11/glassfish4/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
Yakov:bin yfain11$
```

You must have JDK 7 or later installed

Walkthrough 1 (end)

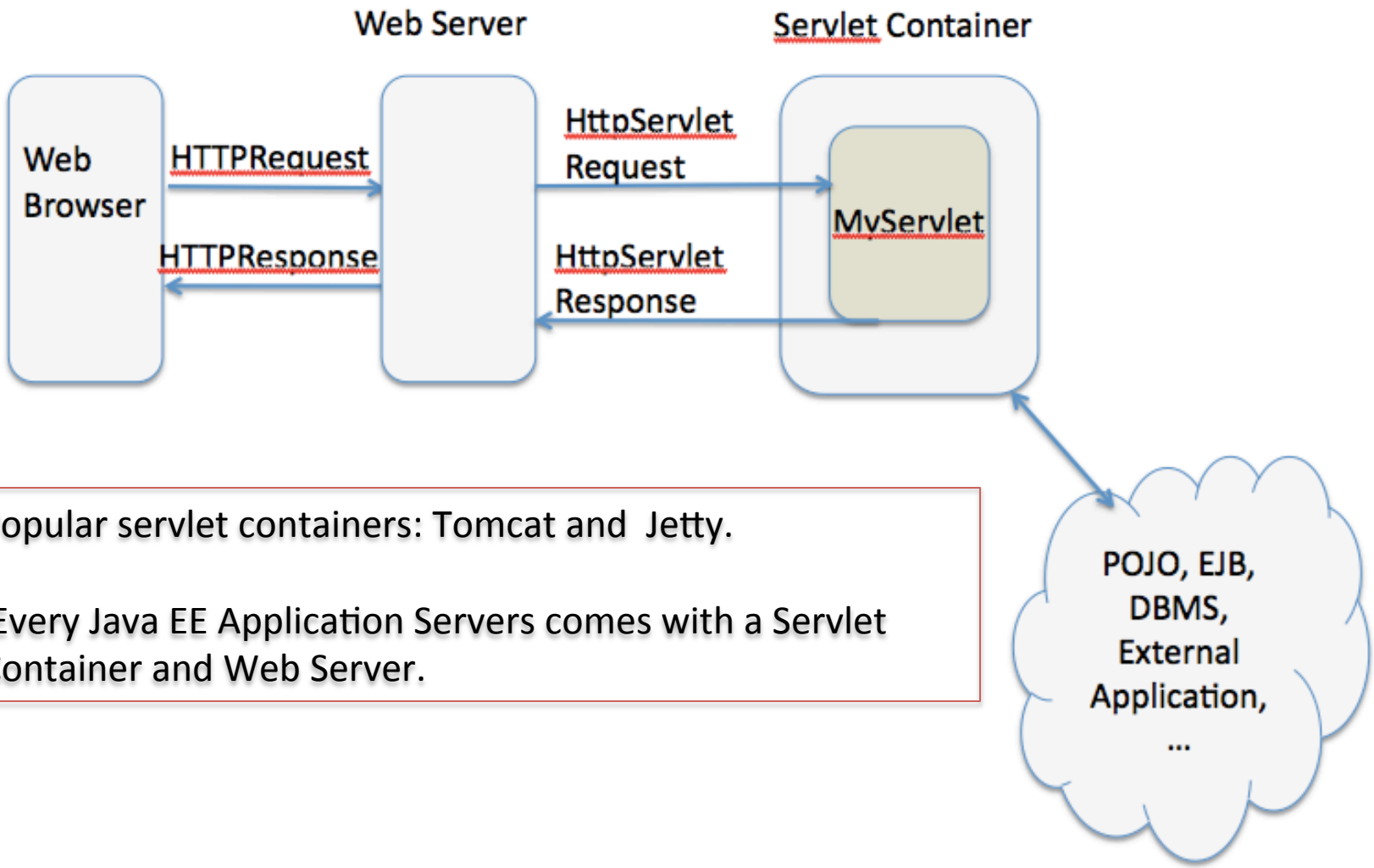
- Test your install by entering http://localhost:8080 - you'll see a Web page stating that GlassFish server is running.

- Open the admin console by visiting http://localhost:4848

- For start/stop instructions refer to Quick Start Guide, section Starting and Stopping the Default Domain:

https://glassfish.java.net/docs/4.0/quick-start-guide.pdf

Web applications with Servlets



Popular servlet containers: Tomcat and Jetty.

Every Java EE Application Servers comes with a Servlet Container and Web Server.

Sample Web Page With Servlets: MyBooks.com

1. The client's machine just needs a Web browser. The bookstore will consist of a number of HTML Web pages for getting user's input, send it in a form of HTTPRequest object to MyBooks.com.

2. *The computer that MyBooks.com is mapped to* has to run some Web Server software that *listens to* the users' requests. If a **Web server** receives a simple request of a static HTML content, it'll process the request and will send back HTTPResponse with the requested **static content**.

3. The Web site MyBooks.com will also run a **servlet container** with deployed Java servlet(s). If the Web server receives a user request to find books based on some criteria, it'll create and pass HttpServletRequest to the appropriate servlet deployed in the *servlet container*.

4. The servlet creates the HTML page with found books that meet search criteria, and sends this **dynamic content** to the Web server in HttpServletResponse, which wraps it inside HttpResponse object and sends it back to the user's Web browser.

5. The user's browser displays the received HTML document.

The Thin HTML Client

```
<HTML>
   <Head>
    <Title>Find a book</Title>
   </Head>

   <Body>
     Enter a word from the book title:

    <Form action=http://www.MyBooks.com/servlet/FindBooks method=Get>
       <input type=Text name=booktitle>
       <input type=Submit value="Search">
    </Form>

   </Body>
</HTML>
```

Walkthrough 2

- Create a plain text file *BookSearch.html* with the content from the previous slide.

- Open this file in a web browser using the menu File | Open, and enter any text in the input field and press the button Search.

- You'll get the error message because there is neither server, nor servlet FindBooks at this address.

How to Write a Java Servlet

- To create a servlet, write a Java class that extends from `HTTPServlet` and annotate it with `@WebServlet` annotation.

- The class HTTPServlet extends `GenericServlet`, which defines the method `service()`.

- The method `service()` receives the client's response and directs it to one of the methods of HTTPServlet descendent *that you have to override* such as `doGet()`, `doPost()` et al.

Your First Servlet

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;

@WebServlet(urlPatterns="/books", name="FindBooks" )
public class FindBooks extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException{

      // The code processing request goes here
      // The resulting Web page will be sent back via the
      // I/O stream that response variable contains

        PrintWriter out = response.getWriter();
        out.println("Hello from FindBooks");
  }
}
```

You must have the jar with javax.servlet.* in classpath to compile and run.
Java EE SDK includes it, or you can use the one
that comes with your application server.

Deploying a Servlet

Specify servlet deployment parameters in the annotation `@WebServlet`:

```
@WebServlet(urlPatterns="/books", name="FindBooks”)
```

Every application server or servlet container has a directory known as *document root*.

For example, if you put the HTML file TermAndConditions.html in a subfolder *legal* of document root in the server MyBooks.com, the users would need to direct their Web browser to http://www.mybooks.com/legal/TermAndConditions.html.

In GlassFish application server, the default document root is directory /glassfish/domains/domain1/docroot.

In Apache Tomcat it's the directory webapps.
The servlets deployment directory will also be located in document root, but it will contain the subdirectories WEB-INF and (maybe) META-INF.

Web Module Structure

document root dir
- HTML pages
- WEB-INF
  - classes
    - com
      - practicaljava
        - lesson27
          - FindBooks.class
  - lib
  - web.xml (optional)
- META-INF (optional)
  - manifest.mf

Typically, a Web application with servlets is deployed in a Web Archive File (WAR).

Such file has a .war extension, and maintains the same directory structure.

You can create a .war file either using your IDE or using build scripts (prefferable).

Walkthrough 3: Eclipse + Glassfish

- Shut down the GlassFish server if it's running () - run ./asadmin stop-domain located in *bin* folder.
- In Eclipse switch to Java EE perspective.
- Right-click inside the Servers view. Select New | Server. You should see GlassFish 4.0 in the list. If not – click on Download Additional Server adapters and install GlassFish Tools
- Select GlassFish 4.0 and press Next.
- One the next Window, Select JDK 7. Press Browse and select the folder glassfish **located inside** glassfish4. Press Next.
- Press Finish on the next window – no password required.
- You'll see GlassFish in Eclispe Servers View. Right-click and start it.

Creating a Servlet Project in Eclipse

- In Eclipse for Java EE Developers switch to Java EE perspective and create Dynamic Web Project using the File | New menu.

- You can also find see this menu under File | New | Other | Web.

Walkthrough 4 (start)

1. Create a dynamic Web project by selecting Eclipse menu File | New | Other | Web | Dynamic Web Project. Name it *lesson27*.

2. In the dropdown Target runtime is GlassFish 4.0. Press Next, Next, and Finish.

3. Observe the folder WebContent in your project. This is your server-side deployment part.

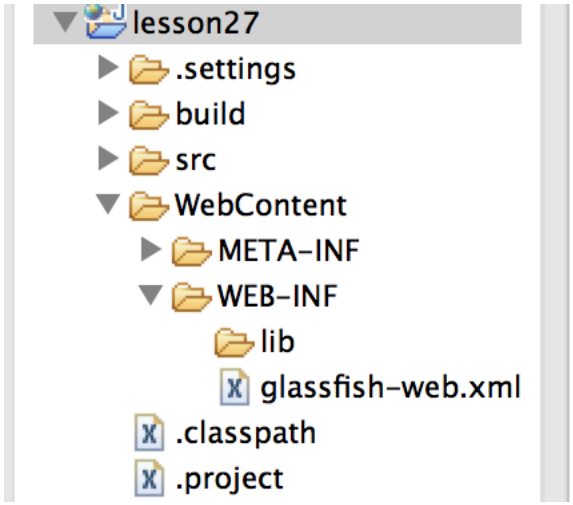
4. Create new servlet: right-click on the project name and select New | Servlet. Specify com.practicaljava.lesson27 as the name of the Java package and FindBooks as the class name. Press Next.

5. In the URL Mappings box select FindBooks, press Edit, and enter /book in the Patterns field. Press OK and Finish.

Walkthrough 4 (continue)

5. In the next window press Finish.

6. In the generated code note the annotated class declaration and methods `doGet()` and `doPost()`.




```
3  import java.io.IOException;
9
10 /**
11  * Servlet implementation class FindBooks
12  */
13 @WebServlet("/book")
14 public class FindBooks extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
```

Walkthrough 4 (end)

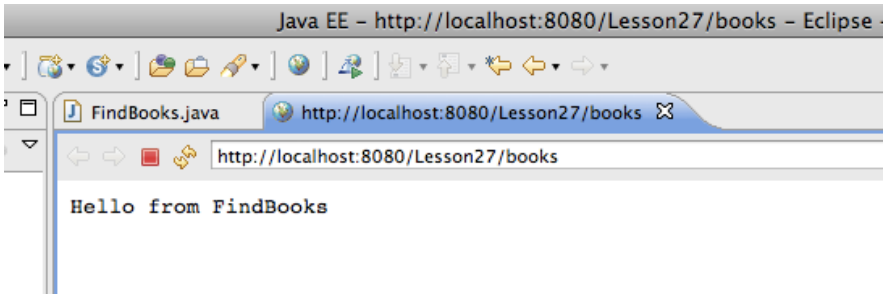
7. Add the following two lines inside the method `doGet()`:

```
PrintWriter out = response.getWriter();
out.println("Hello from FindBooks");
```

8. Correct the errors by importing the `PrintWriter` class.

9. Deploy the servlet in GlassFish: open the Servers view, right-click on the server and select *Add and Remove* from the menu. Select lesson27 in the left panel and add it to the right one. Check the content of the directory, where this app is deployed: *glassfish4/glassfish/domains/domain1/eclipseApps*

10. Run the servlet: right-click on FindBooks and select Run on Server. Confirm deployment under GlassFish. Eclipse will start its internal browser and display the following:



11. Copy the servlet's URL http://localhost:8080/lesson27/book from Eclipse to your Web Browser - you'll see the same output.

Homework

Study all the materials from Lessons 25-27 from the textbook.

1. Study the following HTTP Protocol tutorial from tutsplus.com:
   a) Part 1: http://bit.ly/17mLK87
   b) Part 2: http://bit.ly/11S639i

2. Do the assignment from the Try It section of the lesson 27.

3. After step 1 is complete, stop GlassFish and start it in the Debug mode. Set a breakpoint in the servlet's `doGet()` method.
4. Submit the stock price request from your HTML file and observe the values of the local variables in `doGet()` while stepping through the Java code in the Eclipse debugger.

Additional materials

Watch the video on getting started with GlassFish 4

https://www.youtube.com/watch?v=DQpiuweG7W8

Servlets: http://www.servletworld.com/

Eclipse doc on Web modules and WAR files:
http://goo.gl/7iIkIq

GlassFish server documentation:
http://glassfish.java.net/docs/