ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στα Δίκτυα Υπηρεσιών

Assisting Lecture 7 (Java Restful WS in Google App Engine)

Μύρων Παπαδάκης
Τμήμα Επιστήμης Υπολογιστών

CS-592: Introduction Service Networks Spring 2015

Assisting Lecture: Java Restful Web Services in the Google App Engine

Myron Papadakis (myrpap@gmail.com)

Creating and Deploying Web applications in Google App Engine

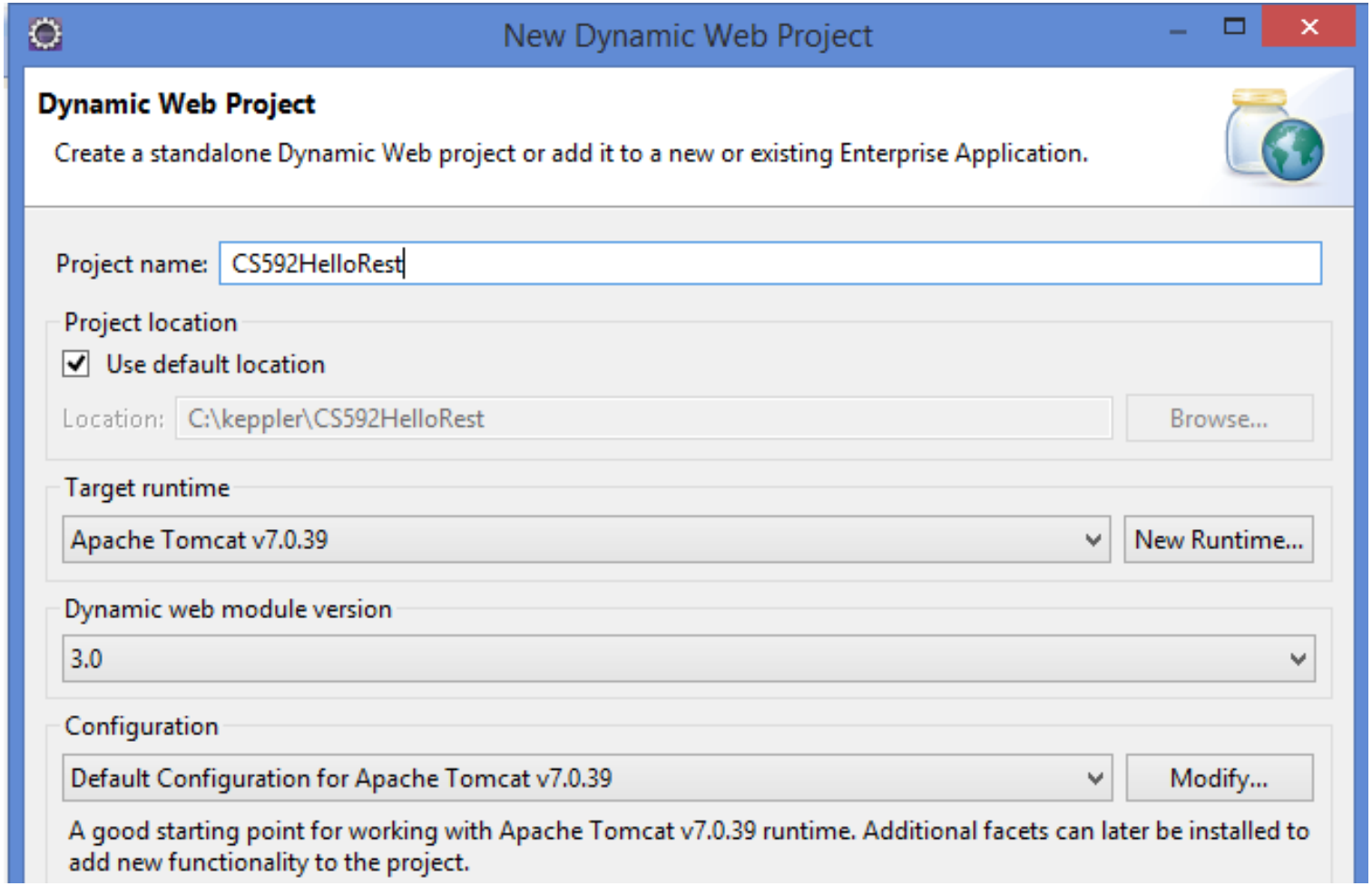
Example 1: Hello World Example using JAX-RS

Download Jersey

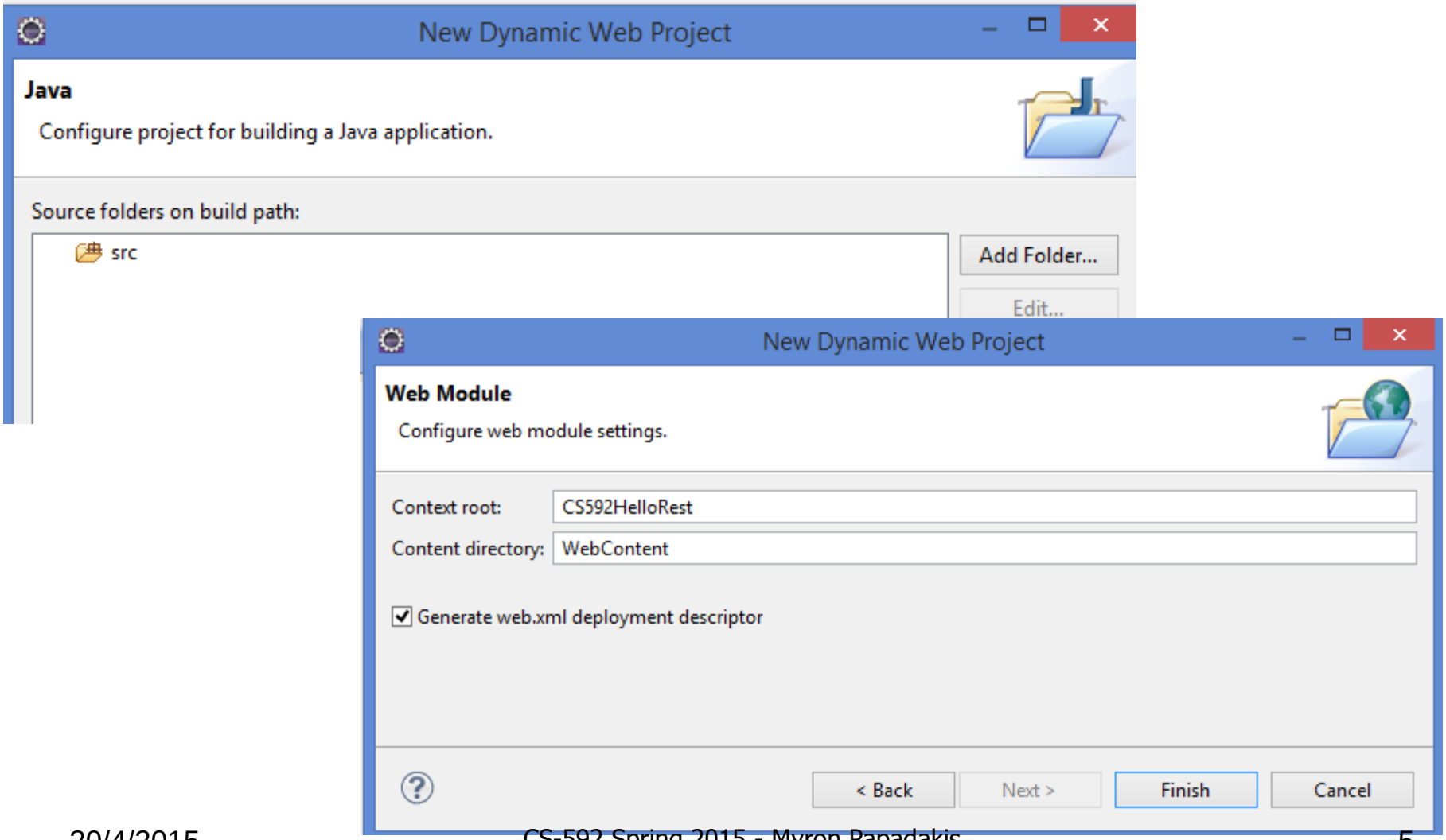
- Download the Jersey distribution as zip file from the Jersey download site.
  - jaxrs-ri-2.16 or jaxrs-ri-2.17.zip
  - It is a collection of jars..

- The zip contains the Jersey implementation JAR and its core dependencies.
  - It does not provide dependencies for third party JARs beyond those for JSON support and JavaDoc.
  - This means that in many cases we have to add some jars from external parties….(i.e. jaxb and gson)

Eclipse > Create new Dynamic Web Project

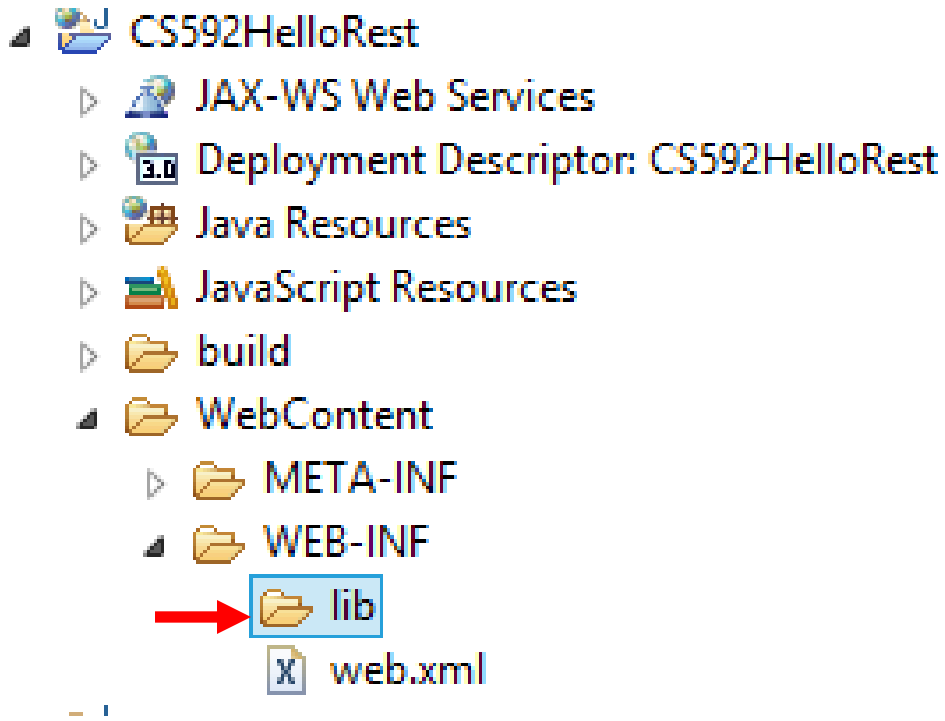


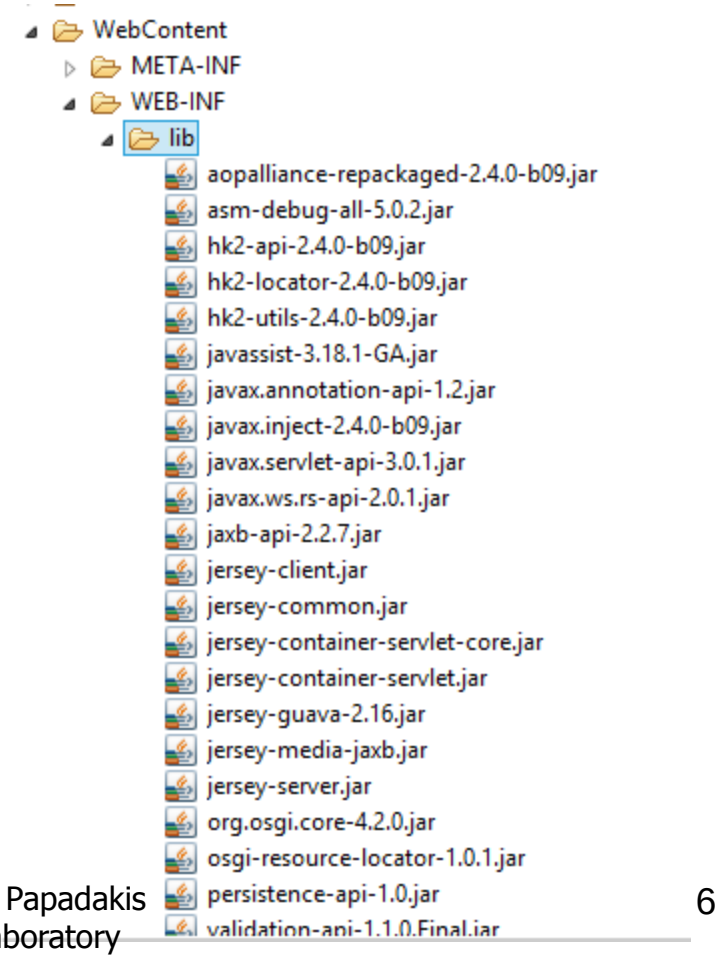
Eclipse > Create new Dynamic Web Project



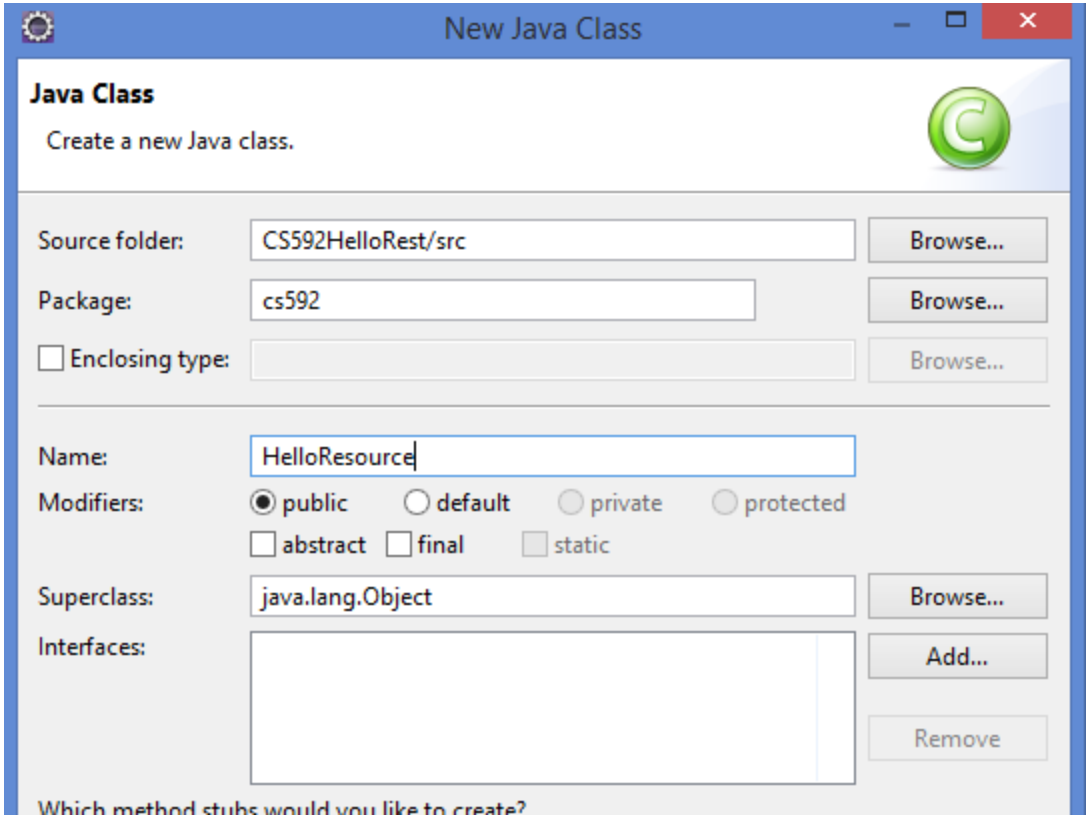
Jersey Jar Copy

- Copy **all JARs** from your Jersey download into the WEB-INF/lib folder.





Create New Class (Resource)



Create New Class (Resource)

```
package cs592;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Response;
import javax.ws.rs.PathParam;
import javax.ws.rs.QueryParam;
import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
@Path("/hello")
public class HelloResource {
```

Create New Class (Resource)

```
@GET
@Path("/query")
@Produces("text/html")
public String getGreetingQueryParam(@QueryParam("name")
  String name) {
        return "<html><body><h1>Hello " + name +
  "!</h1></body></html>";
}

@GET
@Path("{name}")
@Produces("text/html")
public String getGreetingParam(@PathParam("name") String
  name) {
      return "<html><body><h1>Hello " + name +
  "!</h1></body></html>";
}
```

Create New Class (Resource)

```
/*
   * Html File should be located to the WebContent
folder
   * Invoke: http://localhost:8080/HelloRest/form.html
   */
  @POST
  @Path("/form")
  public Response sayHiToUser(
          @FormParam("fname") String fname,
          @FormParam("lname") String lname) {

       String output = "<html><body><h1>Hello " + fname +
" "+lname+"!</h1></body></html>";

      return Response.status(200).entity(output).build();

  }
}
```

Create New HTML File

Define Jersey Servlet dispatcher > web.xml

- Default generated web.xml should look like the following
- We have to modify it to deploy it to Tomcat

HelloResource.java | form.html | web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchemc
3    <display-name>CS592HelloRest</display-name>
4    <welcome-file-list>
5      <welcome-file>index.html</welcome-file>
6      <welcome-file>index.htm</welcome-file>
7      <welcome-file>index.jsp</welcome-file>
8      <welcome-file>default.html</welcome-file>
9      <welcome-file>default.htm</welcome-file>
10     <welcome-file>default.jsp</welcome-file>
11   </welcome-file-list>
12 </web-app>
```

Define Jersey Servlet dispatcher > web.xml

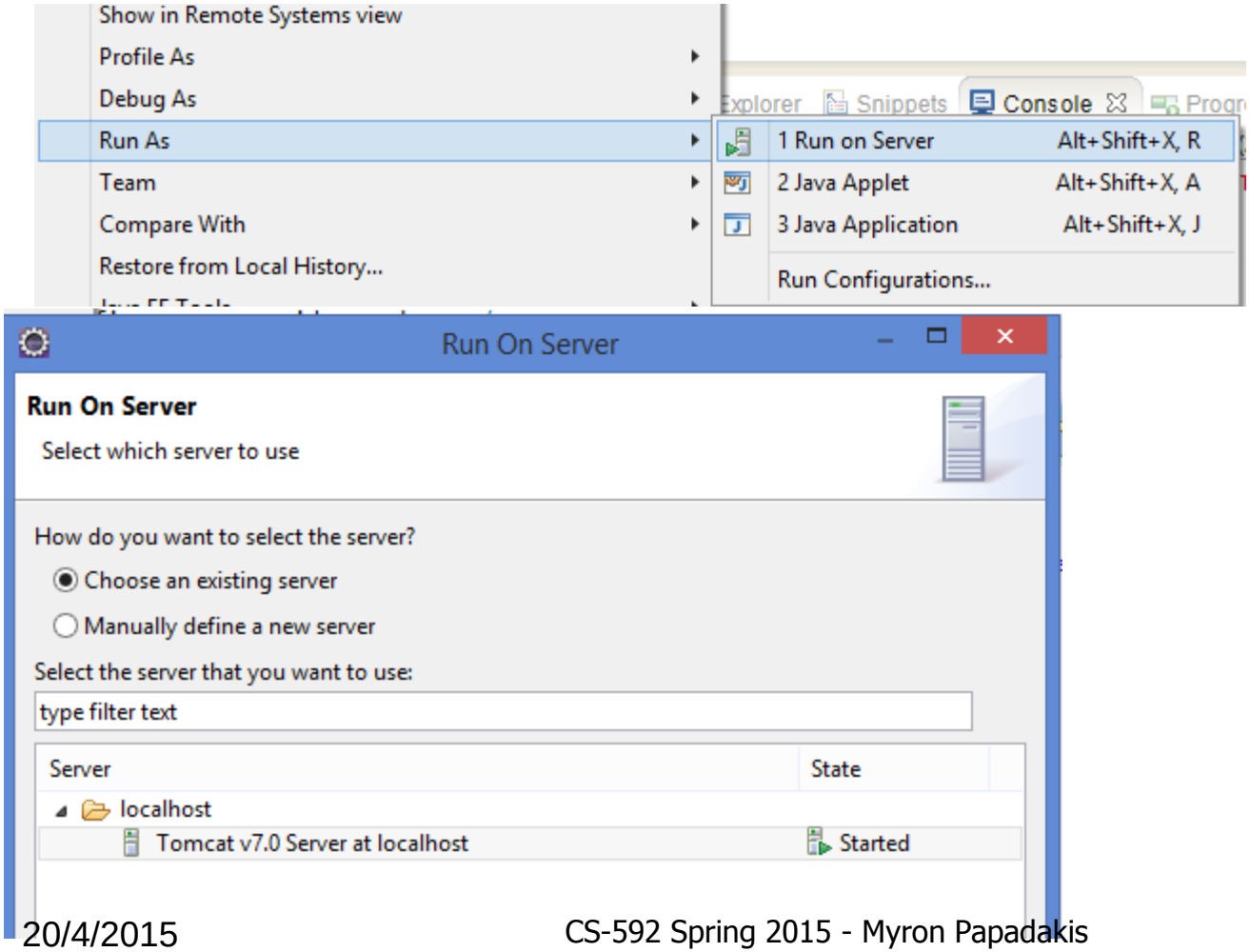
- You need to register Jersey as the servlet dispatcher for REST requests.
- The complete path to a resource is based on the base URL and the @PATh annotation in your class.

- **http://your_domain:port/display-name/url-pattern/path_from_rest_class**

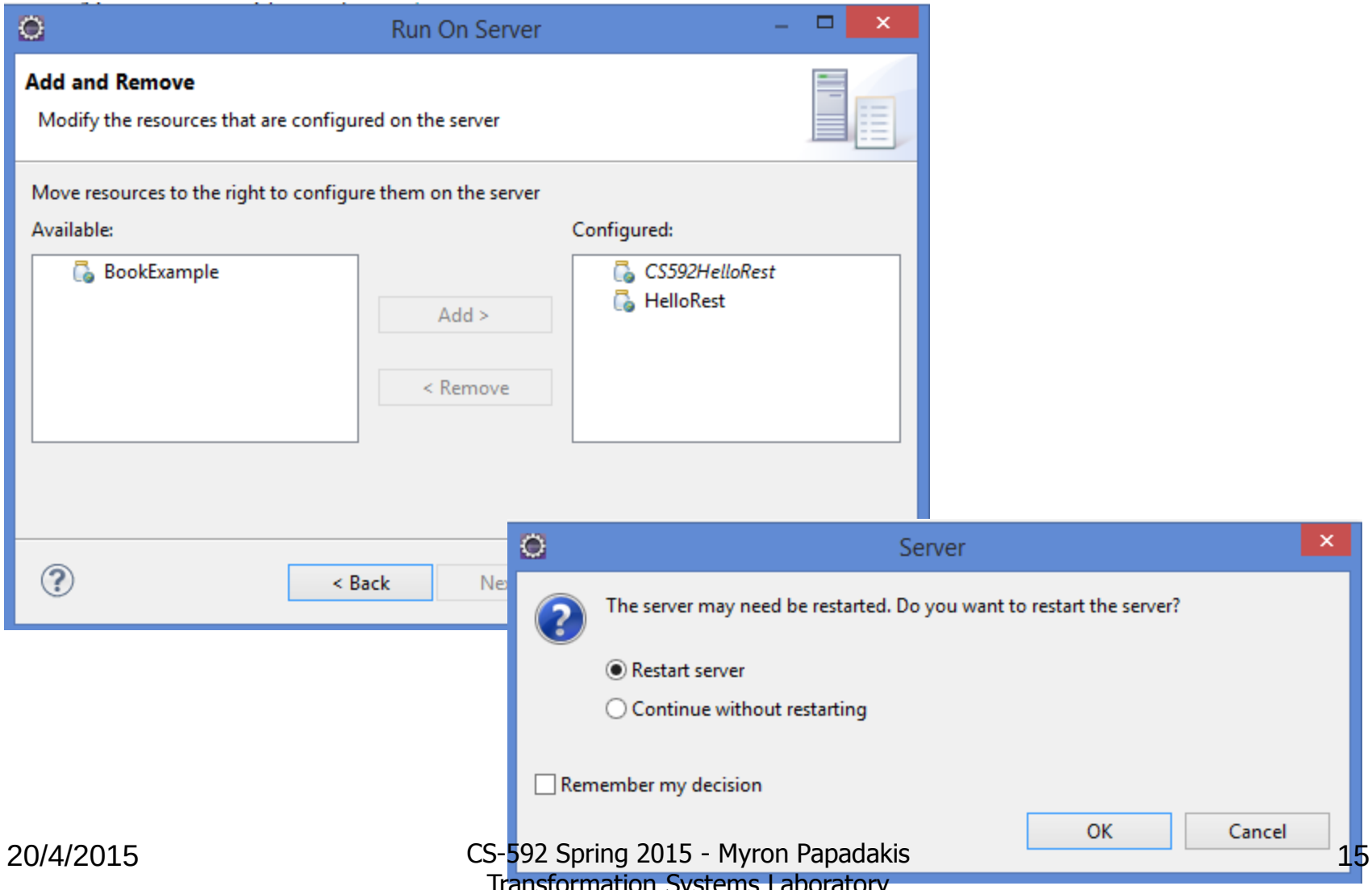
HelloResource.java | form.html | web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/
3    <display-name>CS592HelloRest</display-name>
4    <servlet>
5      <servlet-name>CS592HelloRest</servlet-name>
6      <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
7       <!-- Register resources and providers under com.vogella.jersey.first package. -->
8      <init-param>
9          <param-name>jersey.config.server.provider.packages</param-name>
10         <param-value>cs592</param-value>
11     </init-param>
12     <load-on-startup>1</load-on-startup>
13   </servlet>
14   <servlet-mapping>
15     <servlet-name>CS592HelloRest</servlet-name>
16     <url-pattern>/rest/*</url-pattern>
17   </servlet-mapping>
18 </web-app>
```

Running in Tomcat



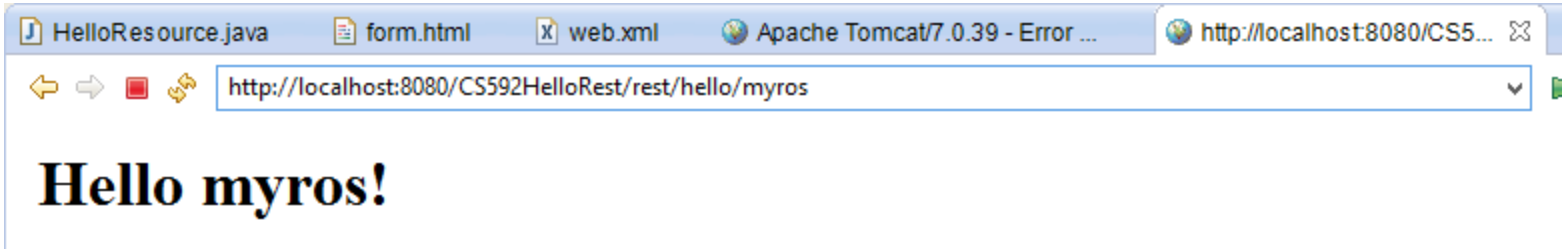
Running in Tomcat



Running in Tomcat

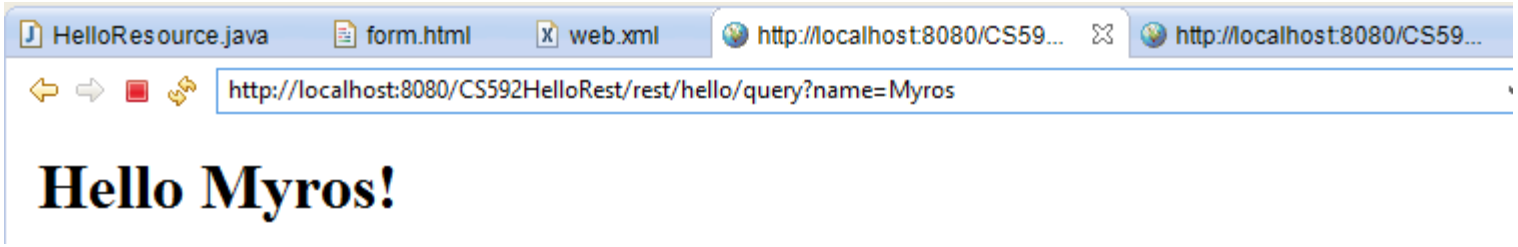
- Don’t worry if you see the following page
- Remember
- **http://your_domain:port/display-name/url-pattern/path_from_rest_class**

Running in Tomcat > PathParam



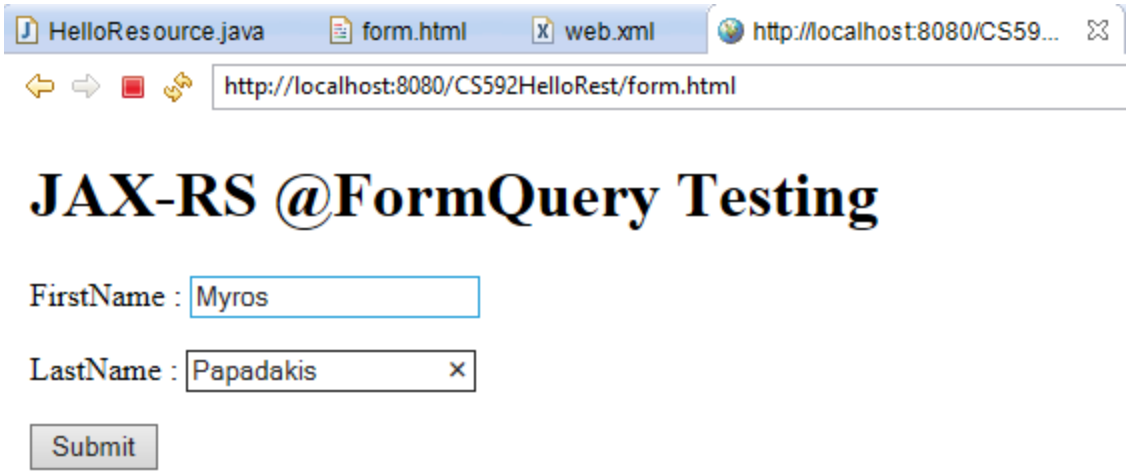

```
@GET
@Path("{name}")
@Produces("text/html")
public String getGreetingParam(@PathParam("name") String name) {
    return "<html><body><h1>Hello " + name + "!</h1></body></html>";
}
```

Running in Tomcat > QueryParam




```
@GET
@Path("/query")
@Produces("text/html")
public String getGreetingQueryParam(@QueryParam("name") String name) {
    return "<html><body><h1>Hello " + name + "!</h1></body></html>";
}
```

Running in Tomcat > FormParam




```
/*
 * Html File should be located to the WebContent folder
 * Invoke: http://localhost:8080/HelloRest/form.html
 */
@POST
@Path("/form")
public Response sayHiToUser(
        @FormParam("fname") String fname,
        @FormParam("lname") String lname) {

    String output = "<html><body><h1>Hello " + fname + " "+lname+"!</h1></body></html>";

   return Response.status(200).entity(output).build();
```

Google App Engine

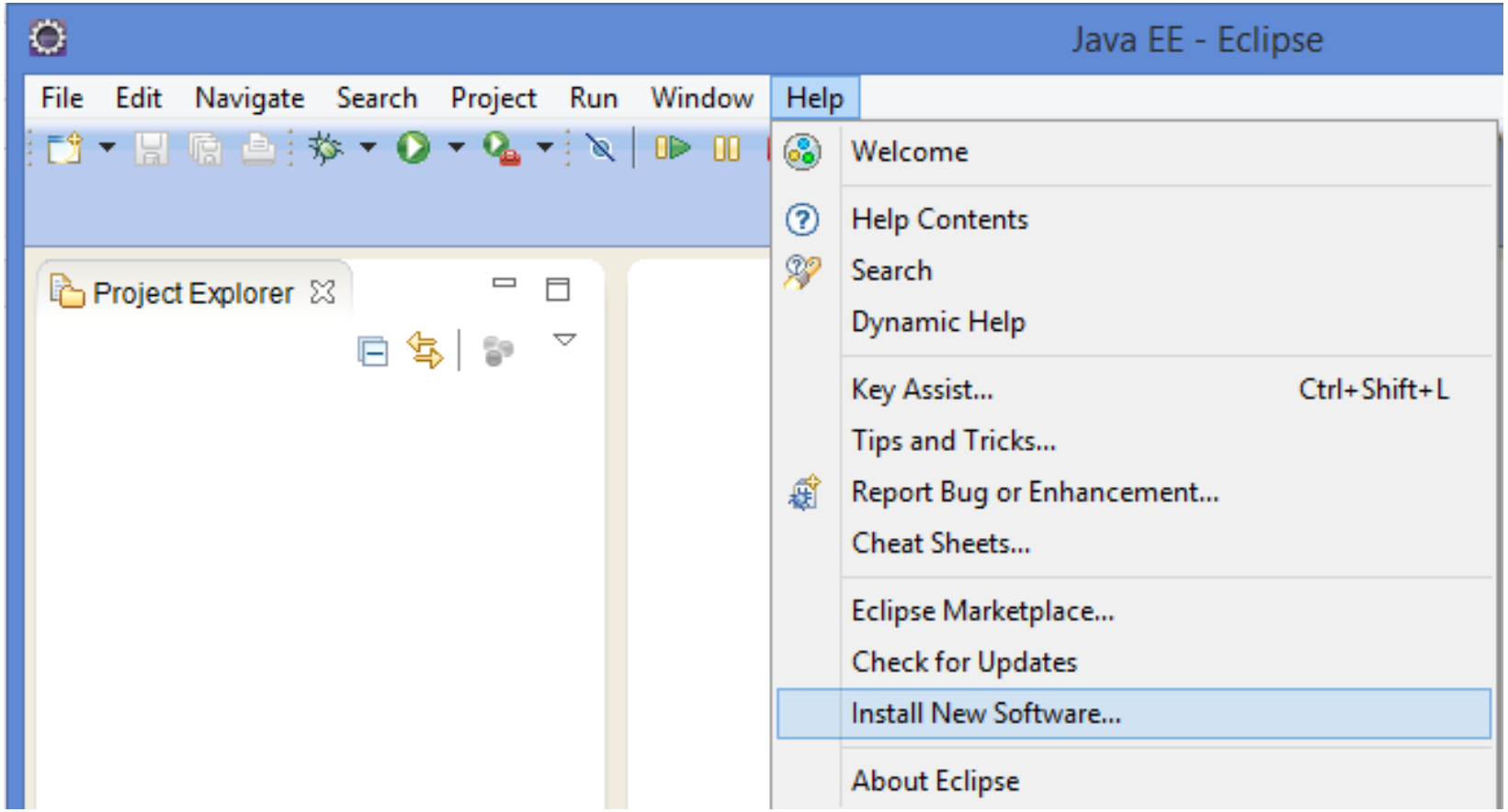
Note

- Unfortunately for Netbeans the google app engine plugin does not work smoothly.
- There are a lot of conflicts with libraries and it is very frustrating
- https://code.google.com/p/nb-gaelyk-plugin/downloads/list

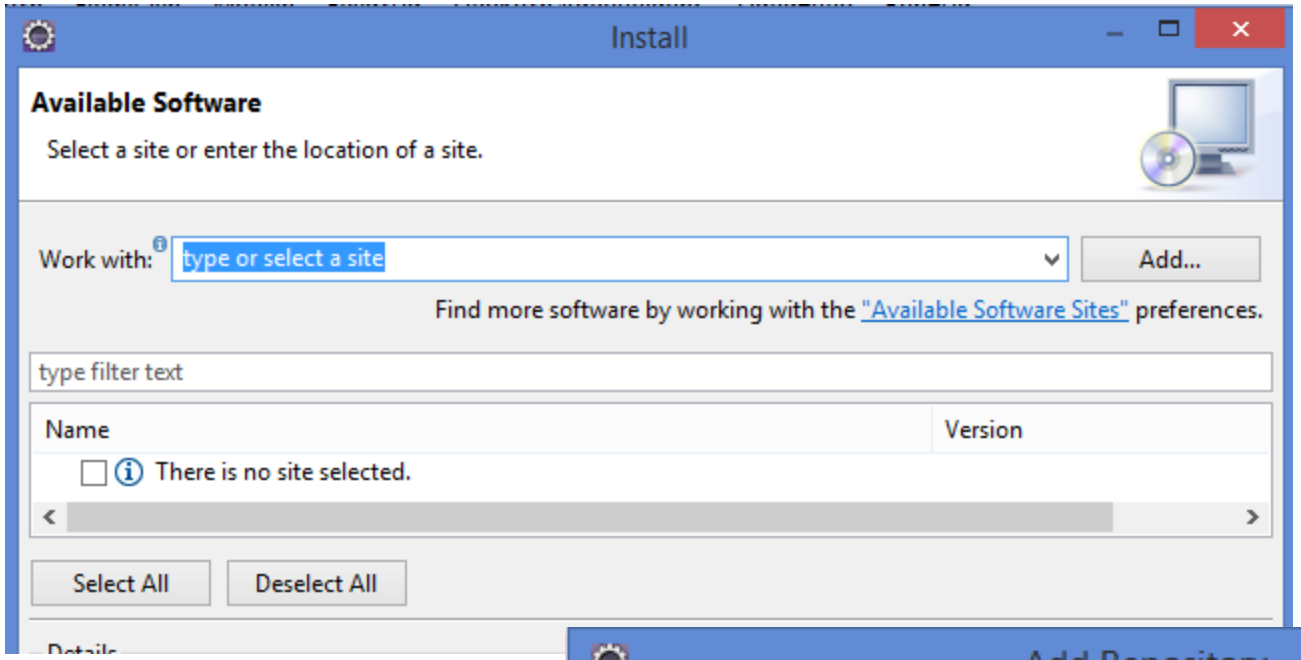
Google Plugin for Eclipse

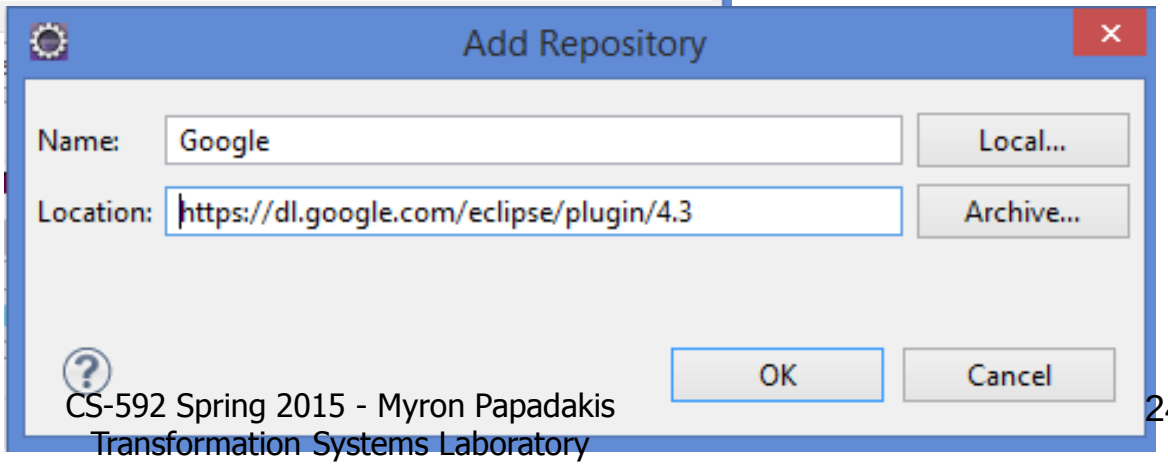
- https://developers.google.com/eclipse/docs/download
- Complete Instructions are given here (for example for Keppler)
  - https://developers.google.com/eclipse/docs/install-eclipse-4.3
- Install the plugin and Restart the Ide
- https://developers.google.com/eclipse/docs/signin

Google Plugin for Eclipse

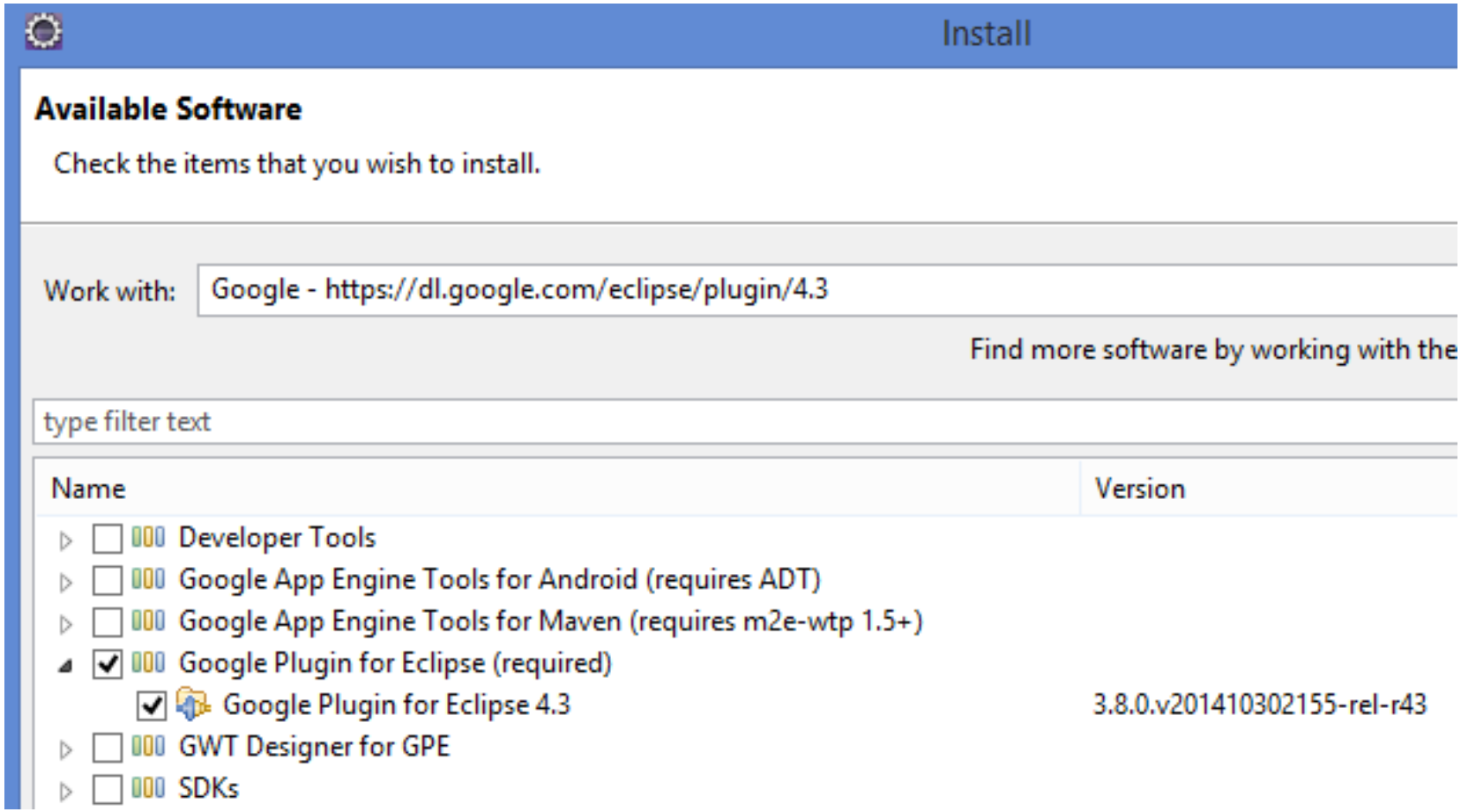


Google Plugin for Eclipse

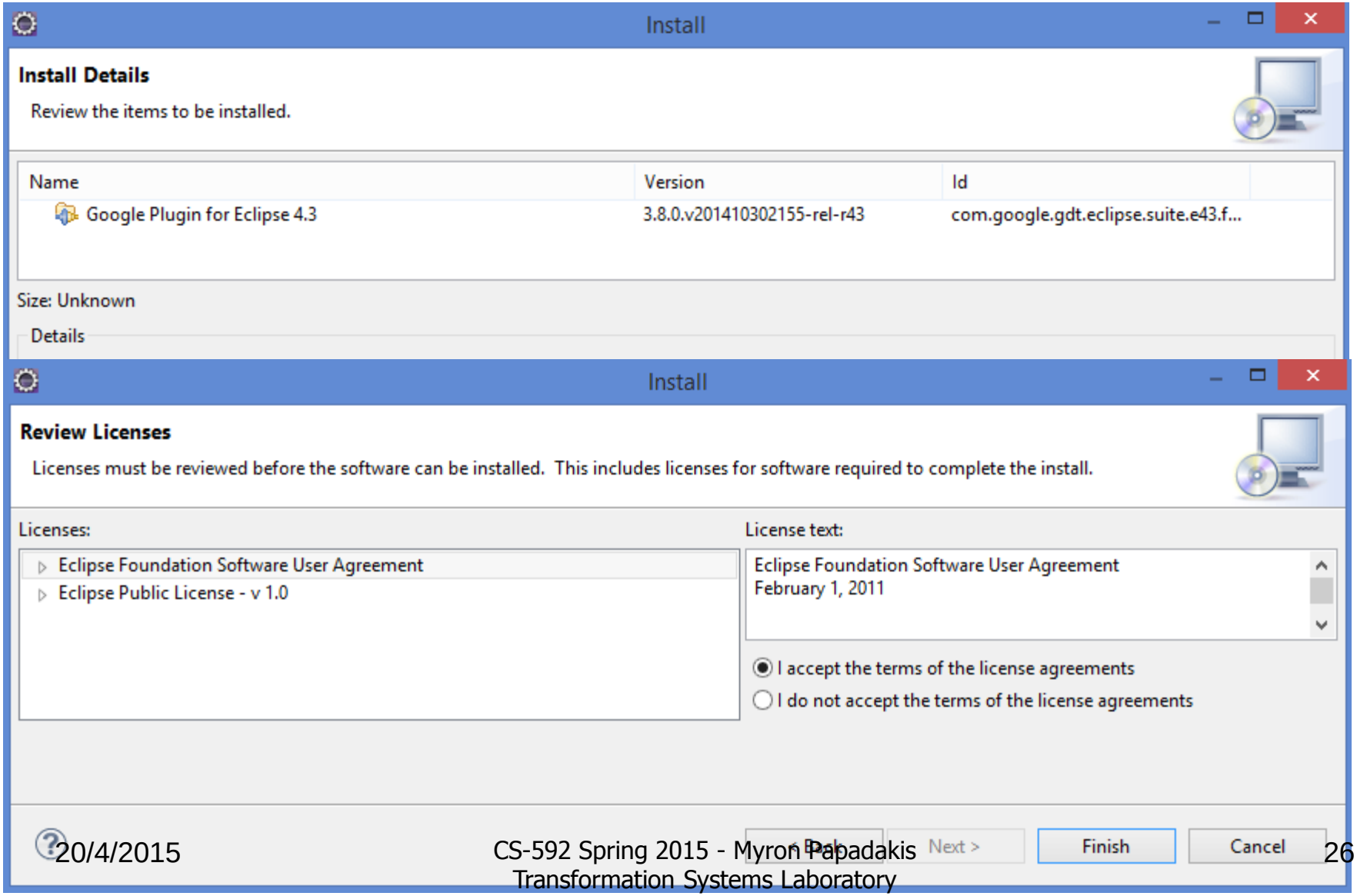




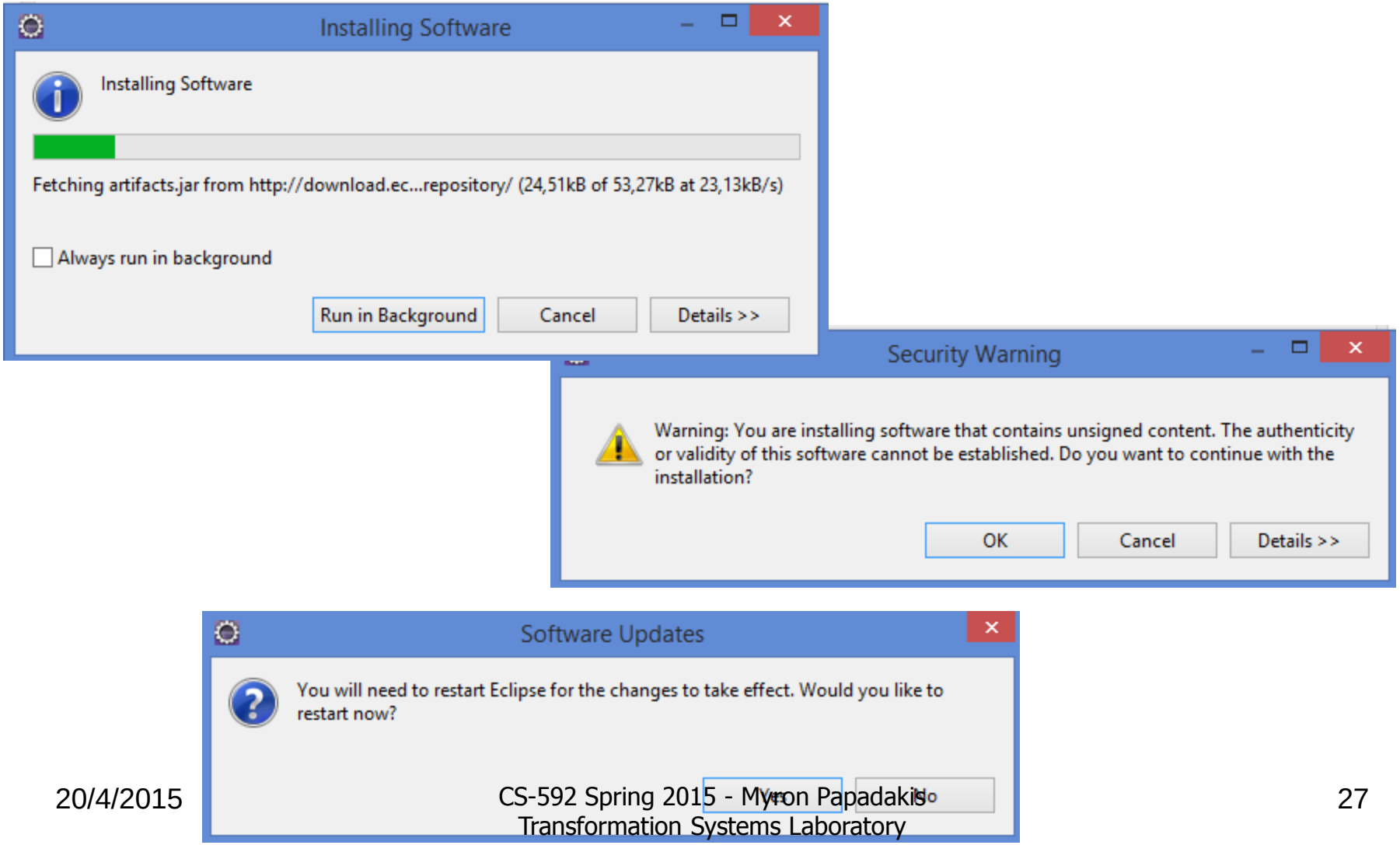
Google Plugin for Eclipse



Google Plugin for Eclipse



Google Plugin for Eclipse

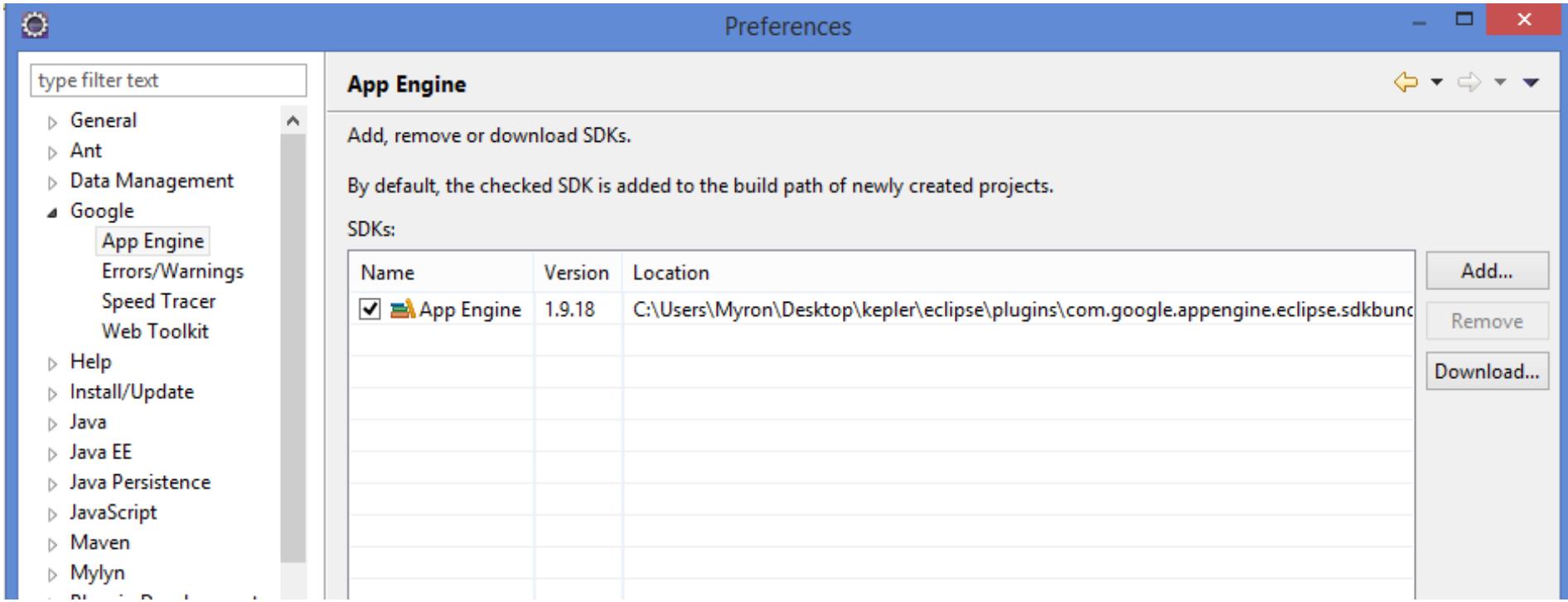


Google App Engine Java Support

Java:

• App Engine runs JAVA apps on a JAVA 7 virtual machine (currently supports JAVA 6 as well).
• Uses JAVA Servlet standard for web applications:
•WAR (Web Applications ARchive) directory structure.
• Servlet classes
• Java Server Pages (JSP)
• Static and data files
• Deployment descriptor (web.xml)
• Other configuration files
• Getting started :
https://developers.google.com/appengine/docs/java/gettingstarted/

Checking Installation



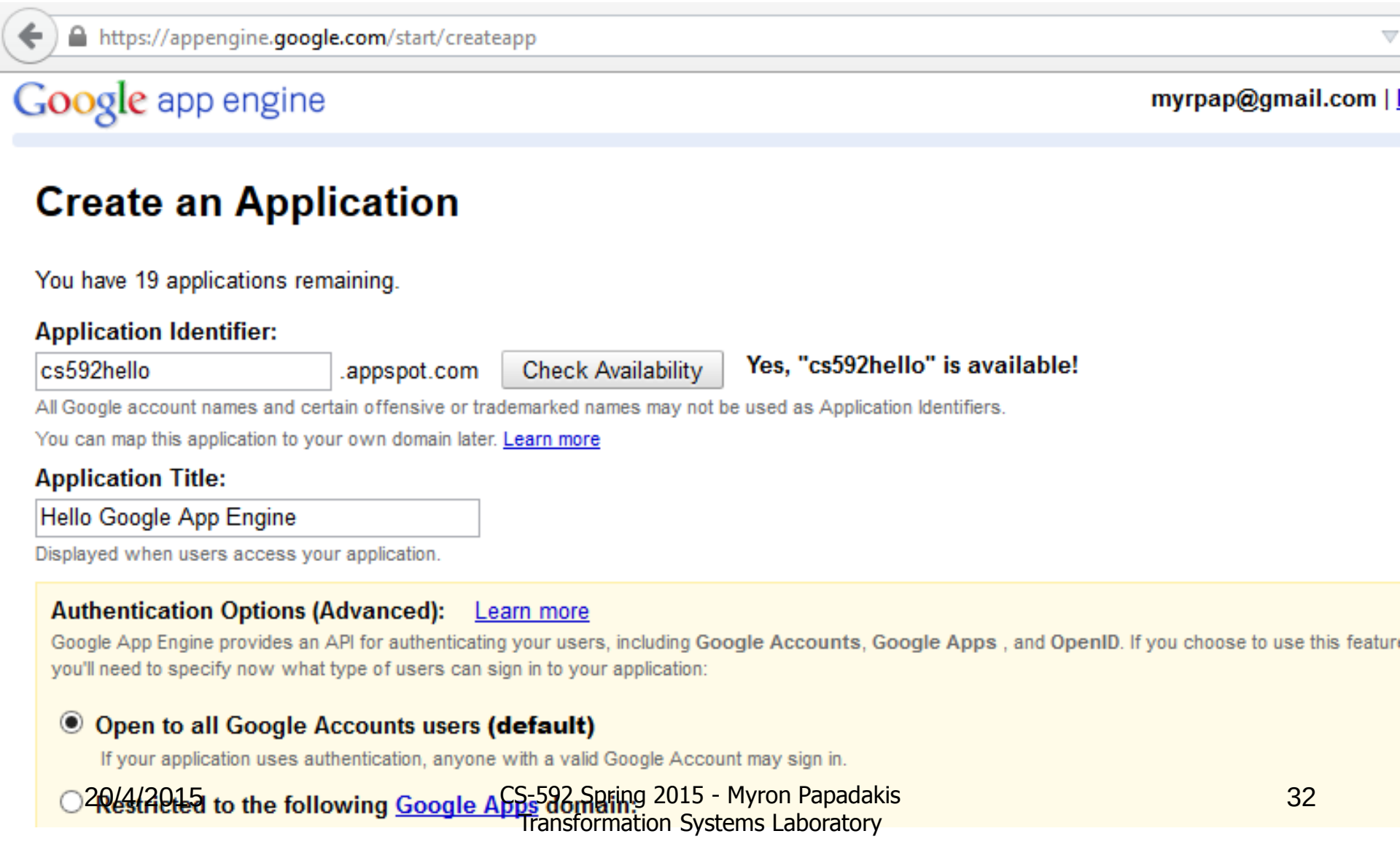
Eclipse Steps

- Create a new application with a certain id in the google app engine.
- Create a new Google Web Application Project
- Configure build path
- Copy to lib all the jersey jars…
- Change web.xml
- Check and fill in the application id in the appengine-web.xml
- There is no need to have Apache Tomcat or Glassfish installed!

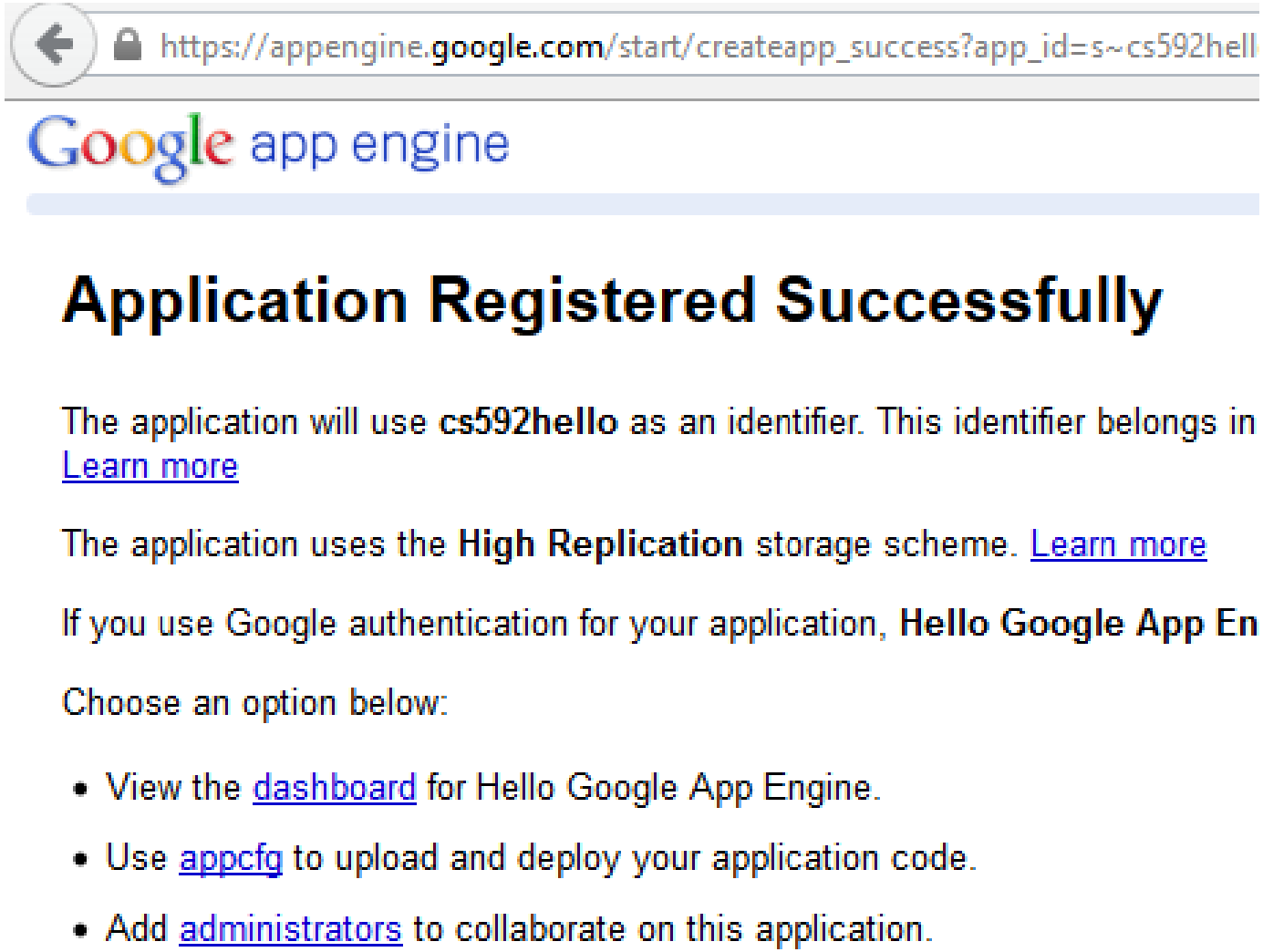
Google App Engine

Hello Example

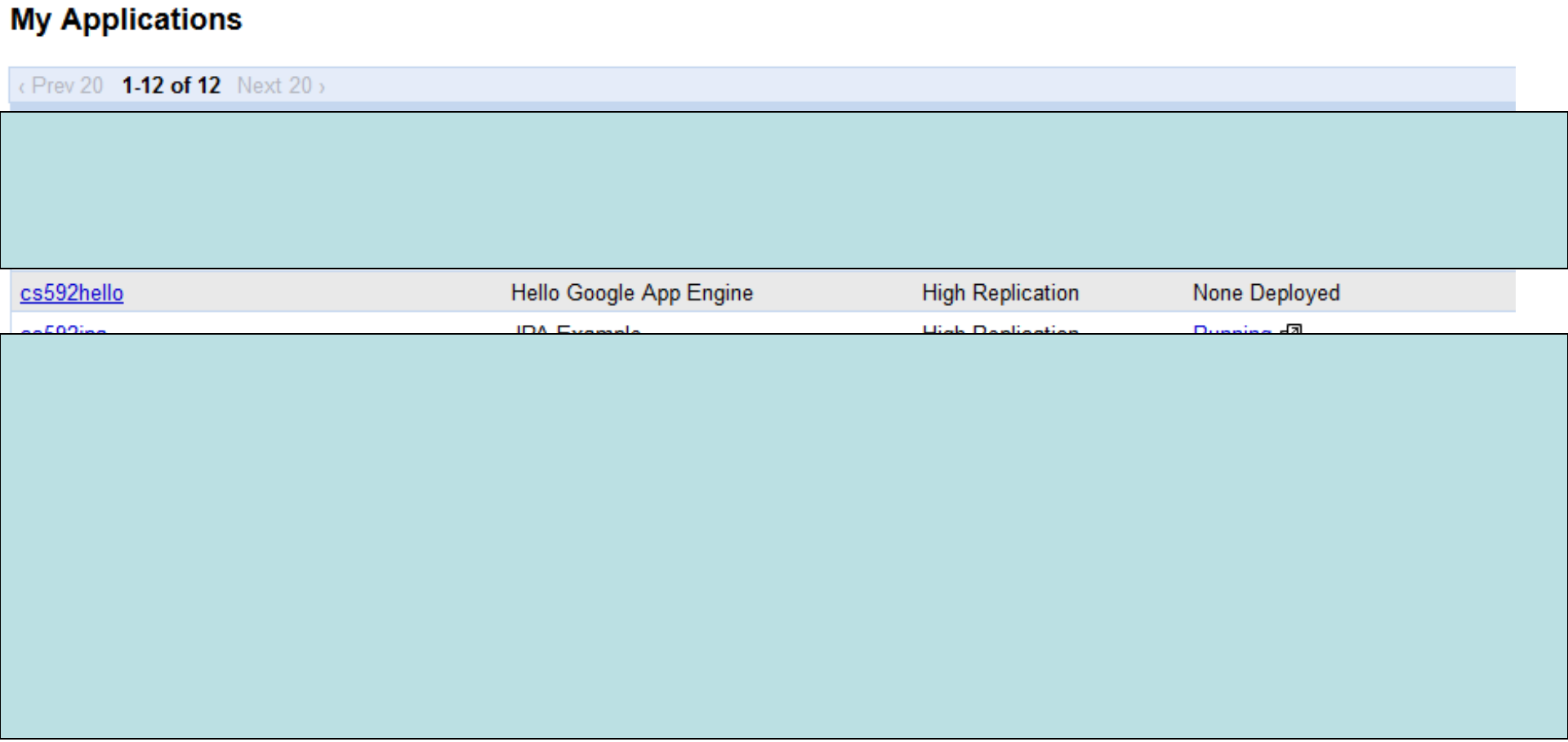
Google app engine > create app



Google app engine > create app

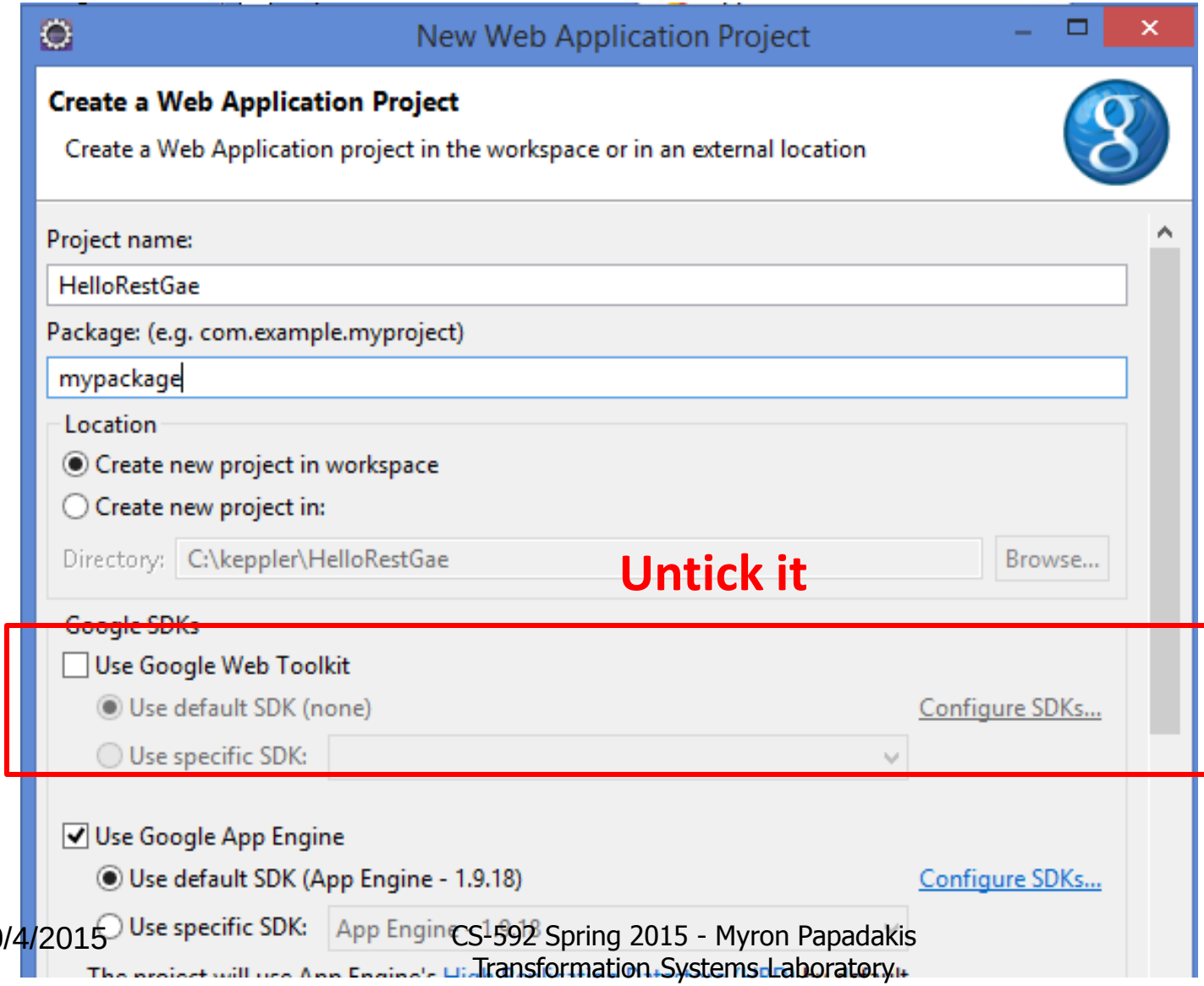


Google app engine > create app



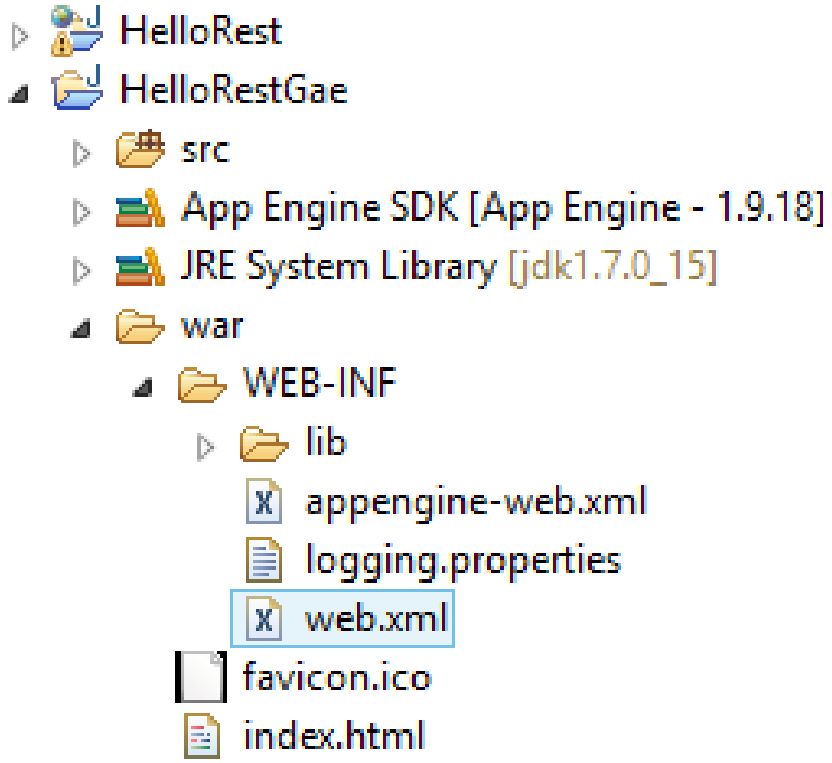
Google App Engine Project

Google App Engine Project



Google App Engine Project > Structure

```
$ tree guestbook/
guestbook/
├── eclipse-launch-profiles
│   ├── DevAppServer.launch
│   └── UpdateApplication.launch
├── nbactions.xml
├── pom.xml
├── README.md
└── src
    ├── main
    │   ├── java
    │   └── webapp
    │       └── WEB-INF
    │           ├── appengine-web.xml
    │           ├── logging.properties
    │           └── web.xml
    └── test
```



Creating the project

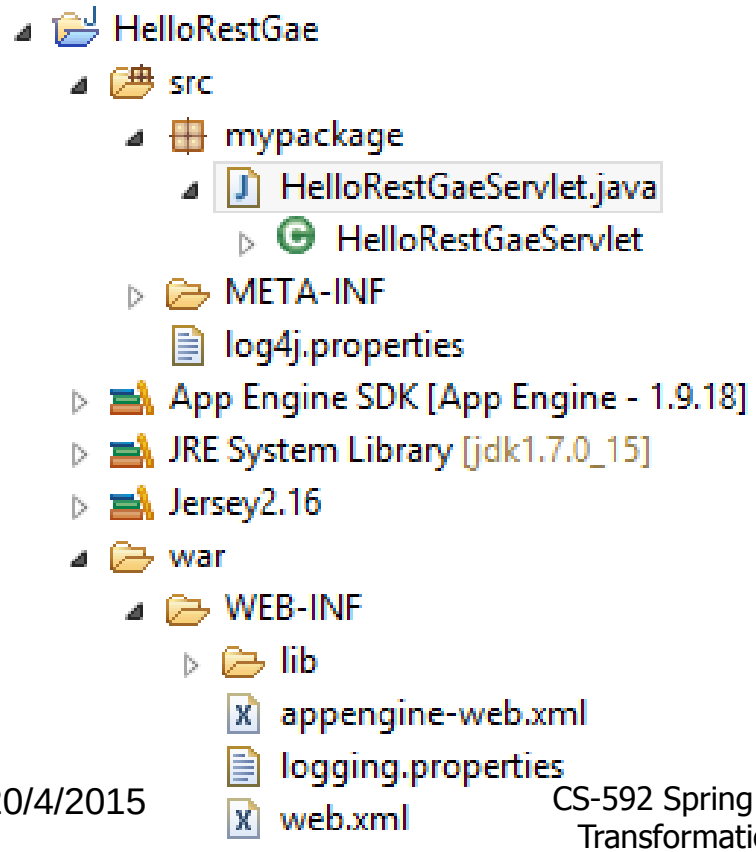
1. You'll add your own application Java classes to src/main/java/...

2. You'll configure your application using the file src/main/webapp/WEB-INF/appengine-web.xml

3. You'll configure your application deployment using the file src/main/webapp/WEB-INF/web.xml

Creating the project > Step1

- So first we will create the resources (actually copy them from the previous project)



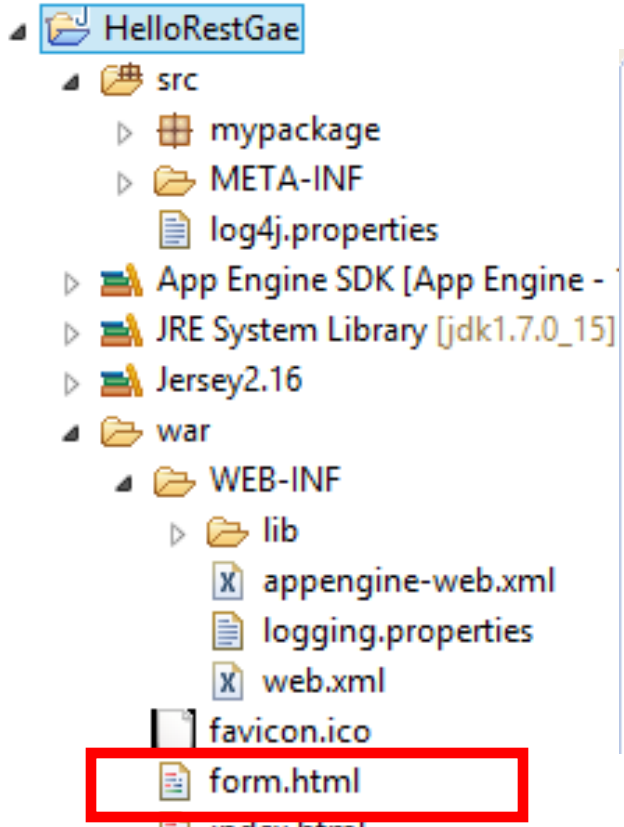
Creating the project > Step1

web.xml | appengine-web.xml | form.html | HelloRestGaeServlet.java | HelloResource.java

```
1  package mypackage;
2
3  import javax.ws.rs.GET;
11 @Path("/hello")
12 public class HelloResource {
13
14
15
16         @GET
17         @Path("/query")
18         @Produces("text/html")
19         public String getGreetingQueryParam(@QueryParam("name") String name) {
20             return "<html><body><h1>Hello " + name + "!</h1></body></html>";
21         }
22
26         public String getGreetingParam(@PathParam("name") String name) {
29
30         /*
31          * Html File should be located to the WebContent folder
32          */
35         public Response sayHiToUser(
44
45 }
```

Creating the project > Step1

- I will also copy the html for the form but I will have to change the path of the action



form.html

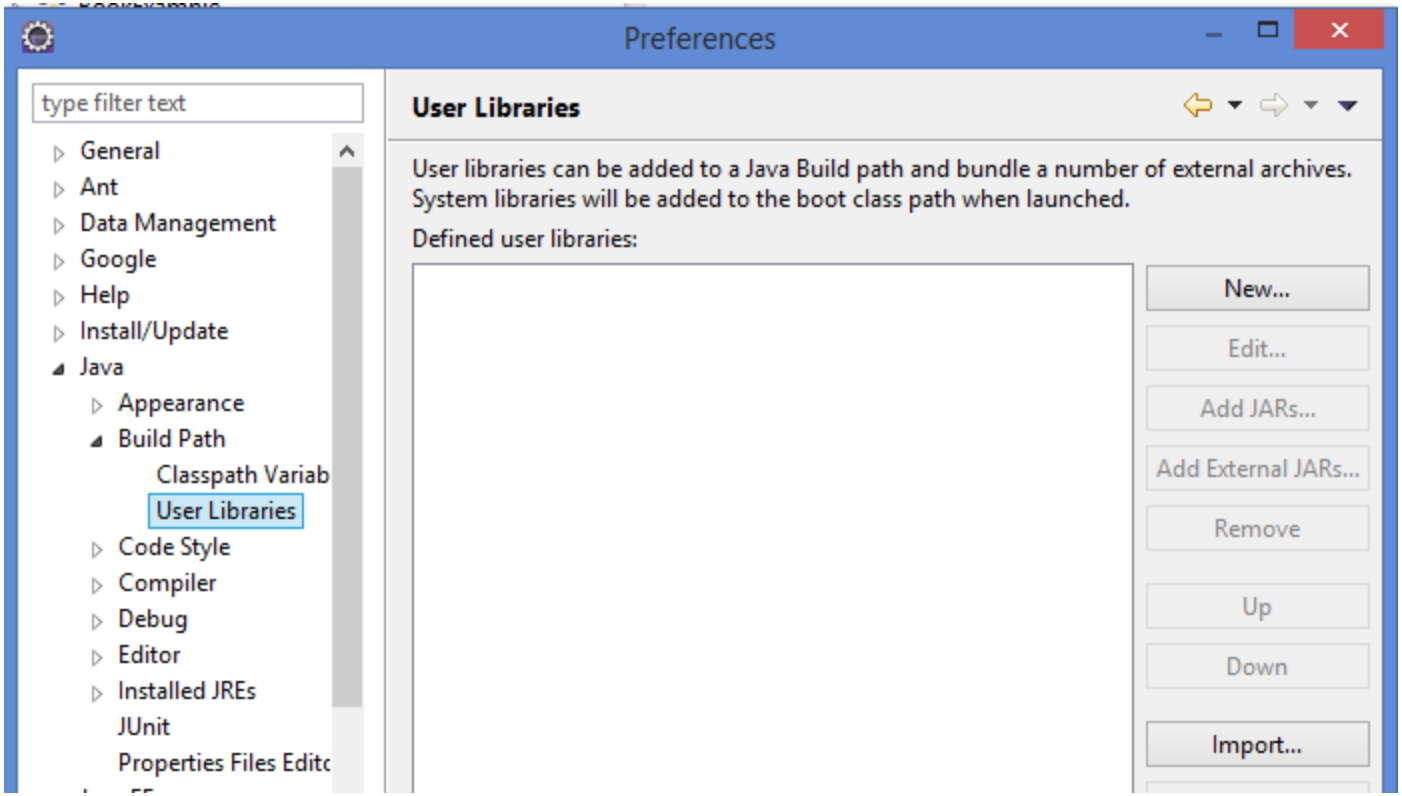
```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h1>JAX-RS @FormQuery Testing</h1>
5
6      <form action="/hellorestgae/hello/form" method="post">
7          <p>
8              FirstName : <input type="text" name="lname" />
9          </p>
10         <p>
11             LastName : <input type="text" name="fname" />
12         </p>
13         <input type="submit" value="Submit" />
14     </form>
15
16 </body>
17 </html>
```

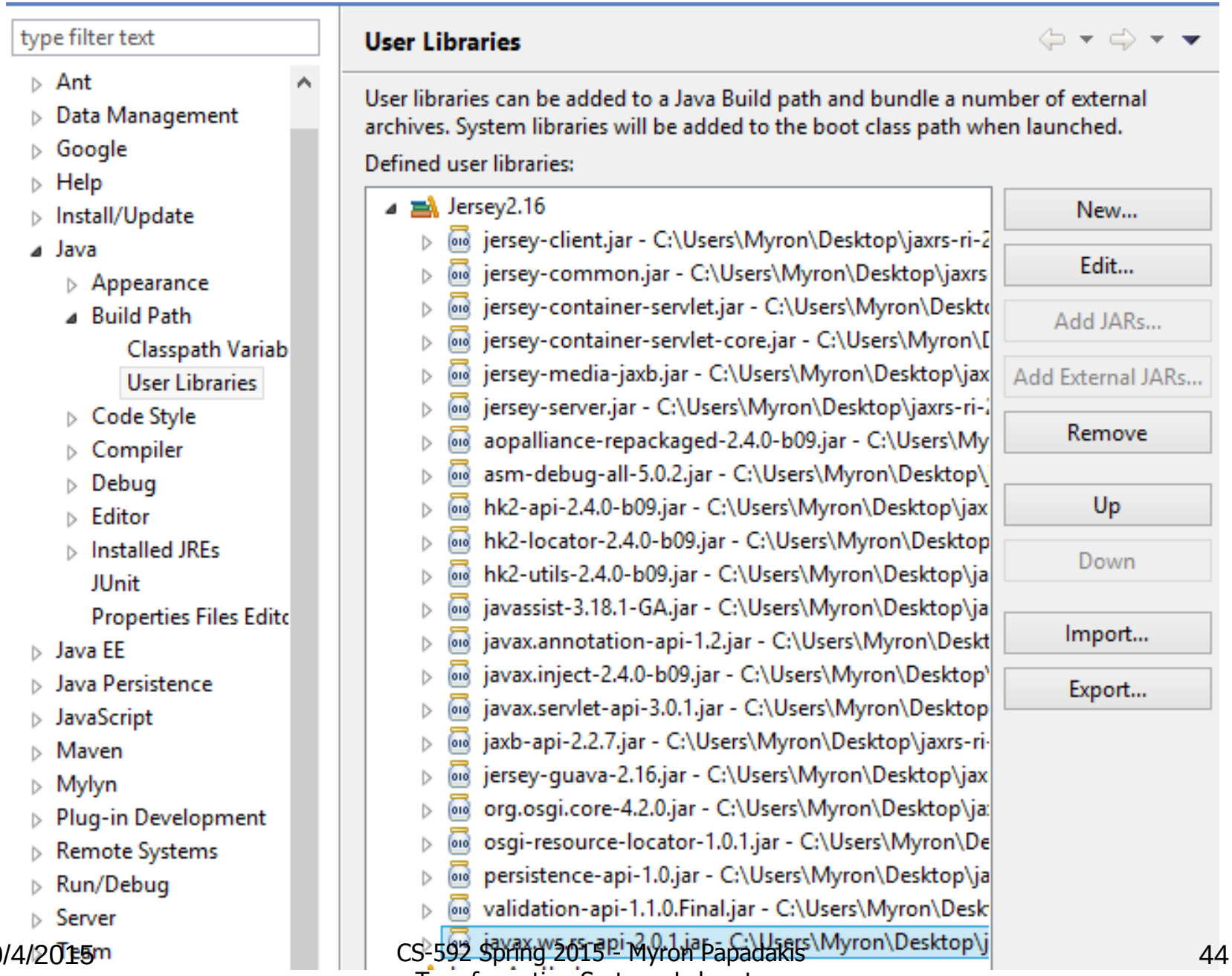
Configuration

- **Configuring the REST support in the application**
  - To be able to create and run REST services in your application you need to:
    Add the JAX-RS, JAXB Jars in your project and application
  - Configure the web application (web.xml) to handle REST requests
- **Add JAX-RS, JAXB to your project**
  Right click on the project and select menu entry **Build Path > Configure Build Path...**
- Click on the Add External JARs button
- Select all the JARs located in $JERSEY_HOME/lib and $JAXB_HOME/lib folders. You can for better visibility and reuse create a user library with all these JARs
- You also need to copy the JARs in the web-inf/lib directory of your application, this step is mandatory to be sure that the JARs are included in the application when deployed to App Engine.

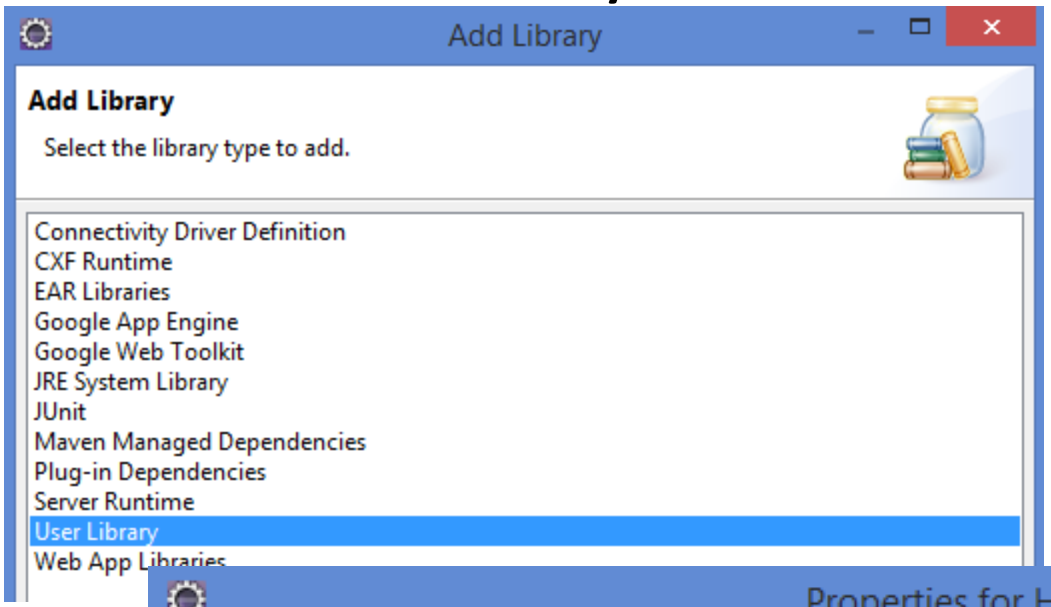
Create a user library for Jersey

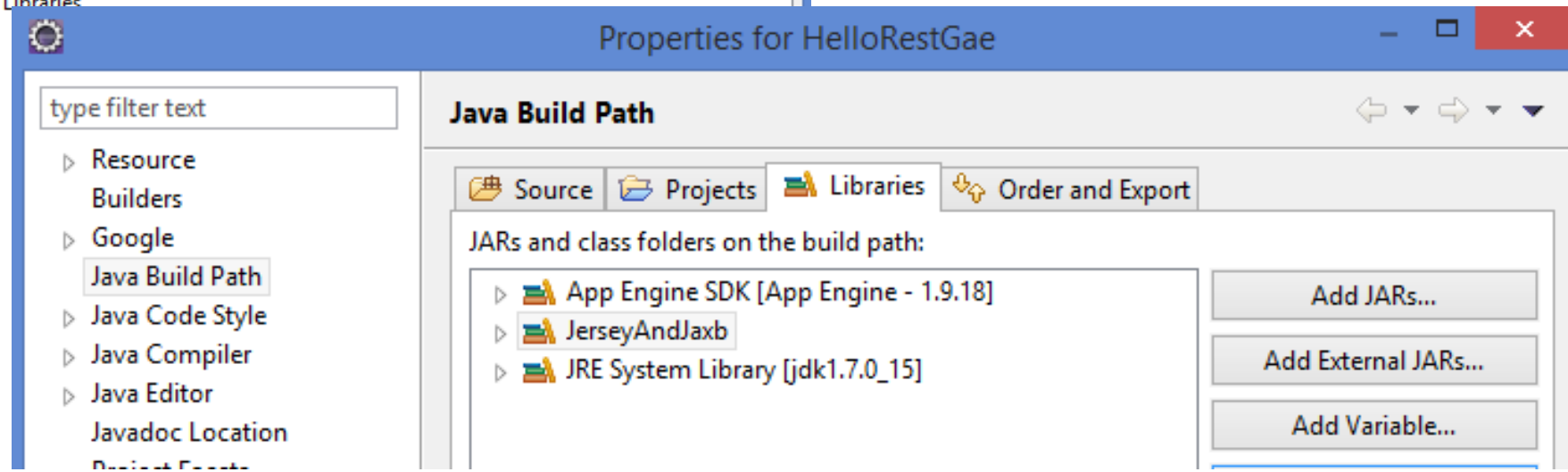
- Window > Preferences > Java > Build Path > User Libraries > New > .. Add External Jars > add the jersey jars




type filter text
Ant
Data Management
Google
Help
Install/Update
Java
Appearance
Build Path
Classpath Variab
User Libraries
Code Style
Compiler
Debug
Editor
Installed JREs
JUnit
Properties Files Editc
Java EE
Java Persistence
JavaScript
Maven
Mylyn
Plug-in Development
Remote Systems
Run/Debug
Server
User Libraries
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.
Defined user libraries:
Jersey2.16
jersey-client.jar - C:\Users\Myron\Desktop\jaxrs-ri-2
jersey-common.jar - C:\Users\Myron\Desktop\jaxrs
jersey-container-servlet.jar - C:\Users\Myron\Deskt
jersey-container-servlet-core.jar - C:\Users\Myron\D
jersey-media-jaxb.jar - C:\Users\Myron\Desktop\jax
jersey-server.jar - C:\Users\Myron\Desktop\jaxrs-ri-
aopalliance-repackaged-2.4.0-b09.jar - C:\Users\My
asm-debug-all-5.0.2.jar - C:\Users\Myron\Desktop\
hk2-api-2.4.0-b09.jar - C:\Users\Myron\Desktop\jax
hk2-locator-2.4.0-b09.jar - C:\Users\Myron\Desktop
hk2-utils-2.4.0-b09.jar - C:\Users\Myron\Desktop\ja
javassist-3.18.1-GA.jar - C:\Users\Myron\Desktop\ja
javax.annotation-api-1.2.jar - C:\Users\Myron\Deskt
javax.inject-2.4.0-b09.jar - C:\Users\Myron\Desktop\
javax.servlet-api-3.0.1.jar - C:\Users\Myron\Desktop
jaxb-api-2.2.7.jar - C:\Users\Myron\Desktop\jaxrs-ri
jersey-guava-2.16.jar - C:\Users\Myron\Desktop\jax
org.osgi.core-4.2.0.jar - C:\Users\Myron\Desktop\ja
osgi-resource-locator-1.0.1.jar - C:\Users\Myron\De
persistence-api-1.0.jar - C:\Users\Myron\Desktop\ja
validation-api-1.1.0.Final.jar - C:\Users\Myron\Desk
javax.ws.rs-api-2.0.1.jar - C:\Users\Myron\Desktop\j
New...
Edit...
Add JARs...
Add External JARs...
Remove
Up
Down
Import...
Export...

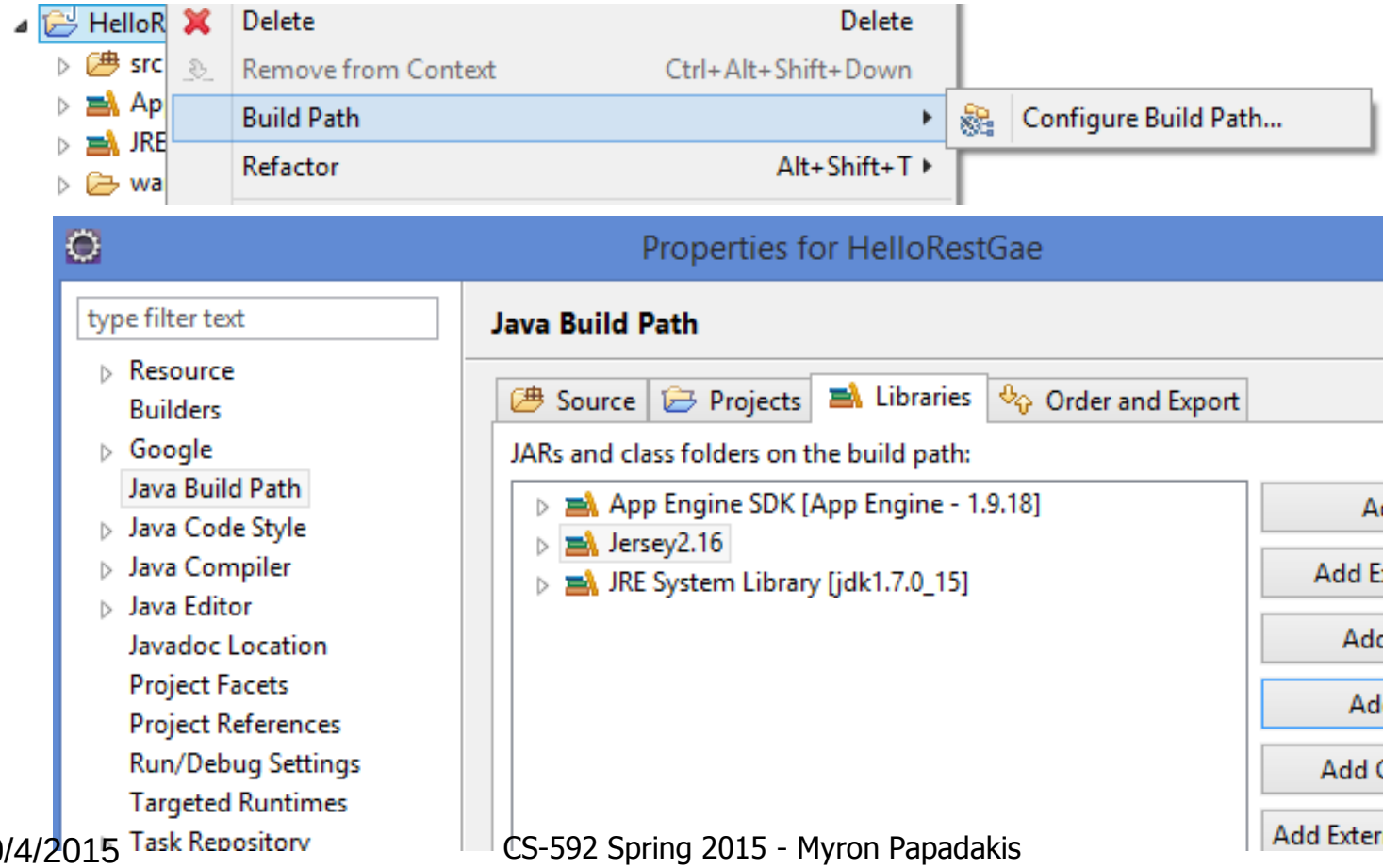
Library Loaded to build path

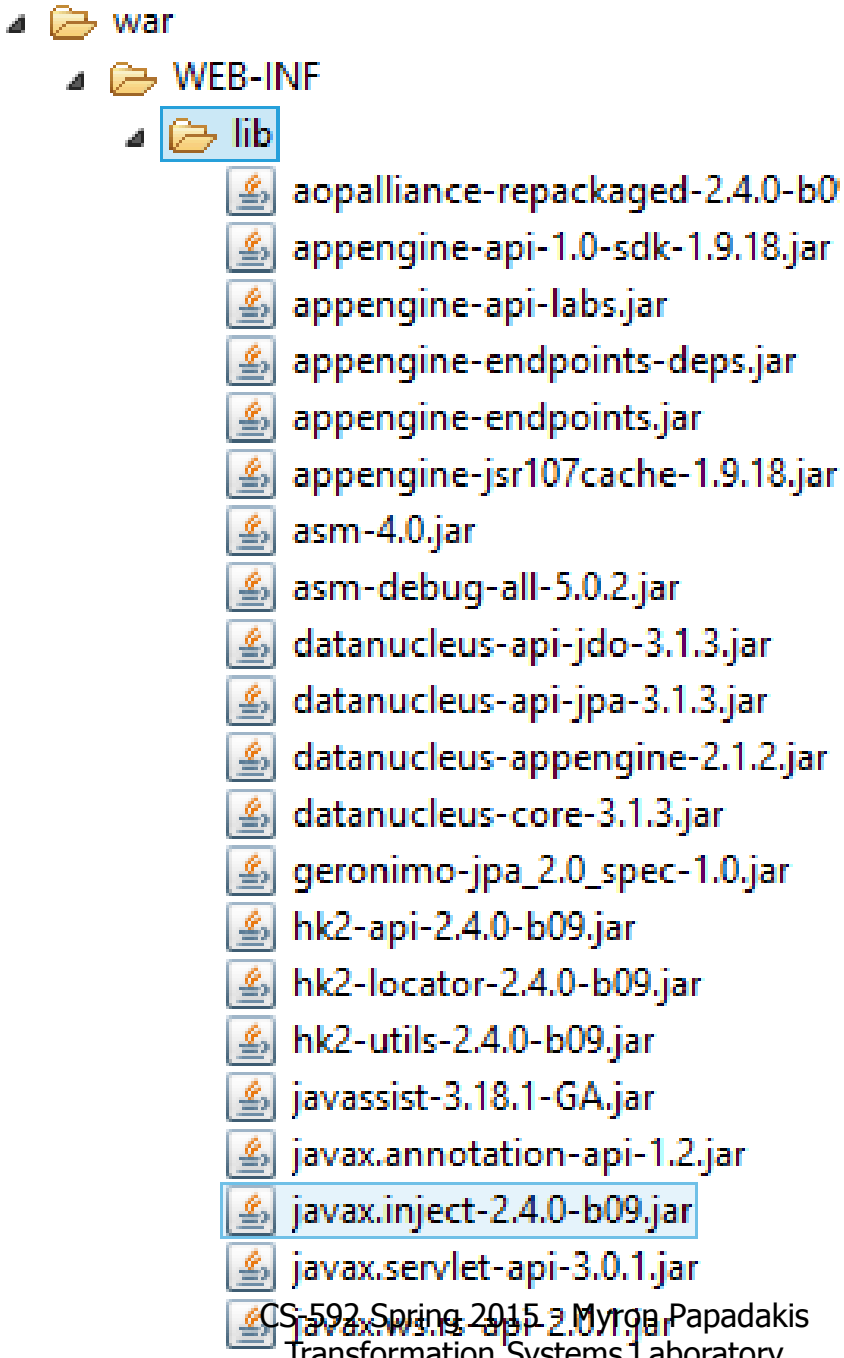




Configuring the build path

- You must also copy the jars to the lib folder




war
WEB-INF
lib
aopalliance-repackaged-2.4.0-b0
appengine-api-1.0-sdk-1.9.18.jar
appengine-api-labs.jar
appengine-endpoints-deps.jar
appengine-endpoints.jar
appengine-jsr107cache-1.9.18.jar
asm-4.0.jar
asm-debug-all-5.0.2.jar
datanucleus-api-jdo-3.1.3.jar
datanucleus-api-jpa-3.1.3.jar
datanucleus-appengine-2.1.2.jar
datanucleus-core-3.1.3.jar
geronimo-jpa_2.0_spec-1.0.jar
hk2-api-2.4.0-b09.jar
hk2-locator-2.4.0-b09.jar
hk2-utils-2.4.0-b09.jar
javassist-3.18.1-GA.jar
javax.annotation-api-1.2.jar
javax.inject-2.4.0-b09.jar
javax.servlet-api-3.0.1.jar

Configuration > Default generated xml file

web.xml

```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" version="2.5">
    <servlet>
        <servlet-name>HelloRestGae</servlet-name>
        <servlet-class>mypackage.HelloRestGaeServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloRestGae</servlet-name>
        <url-pattern>/hellorestgae</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Configuration > Modified web.xml

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  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
7      <servlet>
8          <servlet-name>HelloRestGae</servlet-name>
9          <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
10         <init-param>
11             <param-name>jersey.config.server.provider.packages</param-name>
12             <param-value>mypackage</param-value>
13         </init-param>
14     </servlet>
15     <servlet-mapping>
16         <servlet-name>HelloRestGae</servlet-name>
17         <url-pattern>/hellorestgae/*</url-pattern>
18     </servlet-mapping>
19     <welcome-file-list>
20         <welcome-file>form.html</welcome-file>
21     </welcome-file-list>
22 </web-app>
```

- The configuration parameter jersey.config.server.provider.packages is used by Jersey to list the packages where REST services implementation are located.
- You can put as many package as you need to, you just need to separate the package names by a ;

Deploying the application to Google App Engine

- Before deploying the application you need
  - to register a new application in Google App Engine using the Administration Console (see next slide).
  - In my example I have used “cs592hello" as Application ID.

- You can easily now deploy the application to Google App Engine by clicking on the "Deploy App Engine Project" button available in the Eclipse toolbar.

- To be able to deploy your application to Google App Engine, you need to check that your application can be registered, the application ID is stored in the WEB-INF/lib/appengine-web.xml.

Deploying the application to Google App Engine > appengine.web-xml

- Has extra configuration file appengine-web.xml which specifies:
  - Name of application:
    - <application name>.appspot.com
  - Version of application
  - Logging
  - Enabling HTTP Sessions
  - Enabling SSL
  - System properties and environment variables
  - Differentiating between static and resource files
    - By default all files under /war directory are both resource and static files except .JSP and those under /WEB-INF

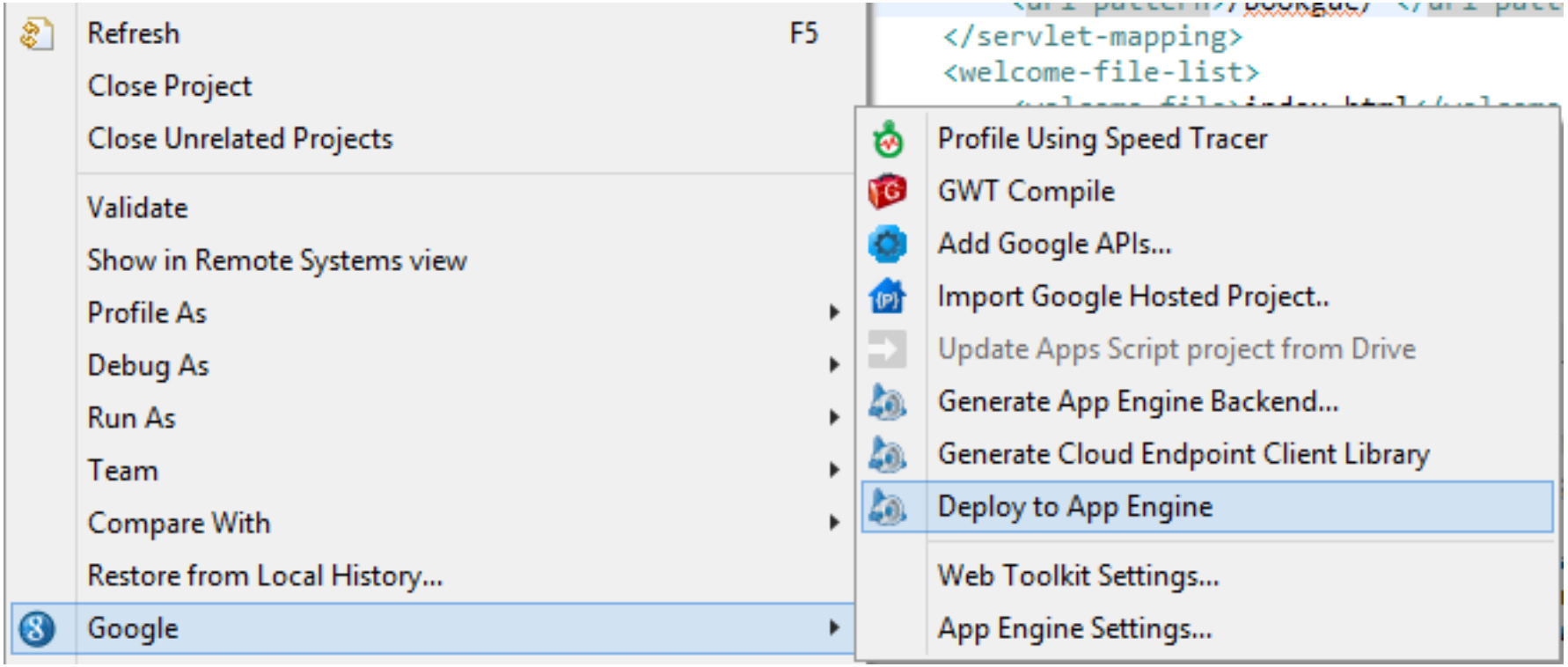
Deploying the application to Google App Engine > appengine.web-xml

web.xml | web.xml | appengine-web.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
3    <application>cs592hello</application>
4    <version>1</version>
5
6    <!--
7      Allows App Engine to send multiple requests to one instance in parallel:
8    -->
9    <threadsafe>true</threadsafe>
10
11   <!-- Configure java.util.logging -->
12   <system-properties>
13     <property name="java.util.logging.config.file" value="WEB-INF/logging.prop
14   </system-properties>
15
```

Deploying the application to Google App Engine

- Right click on the project



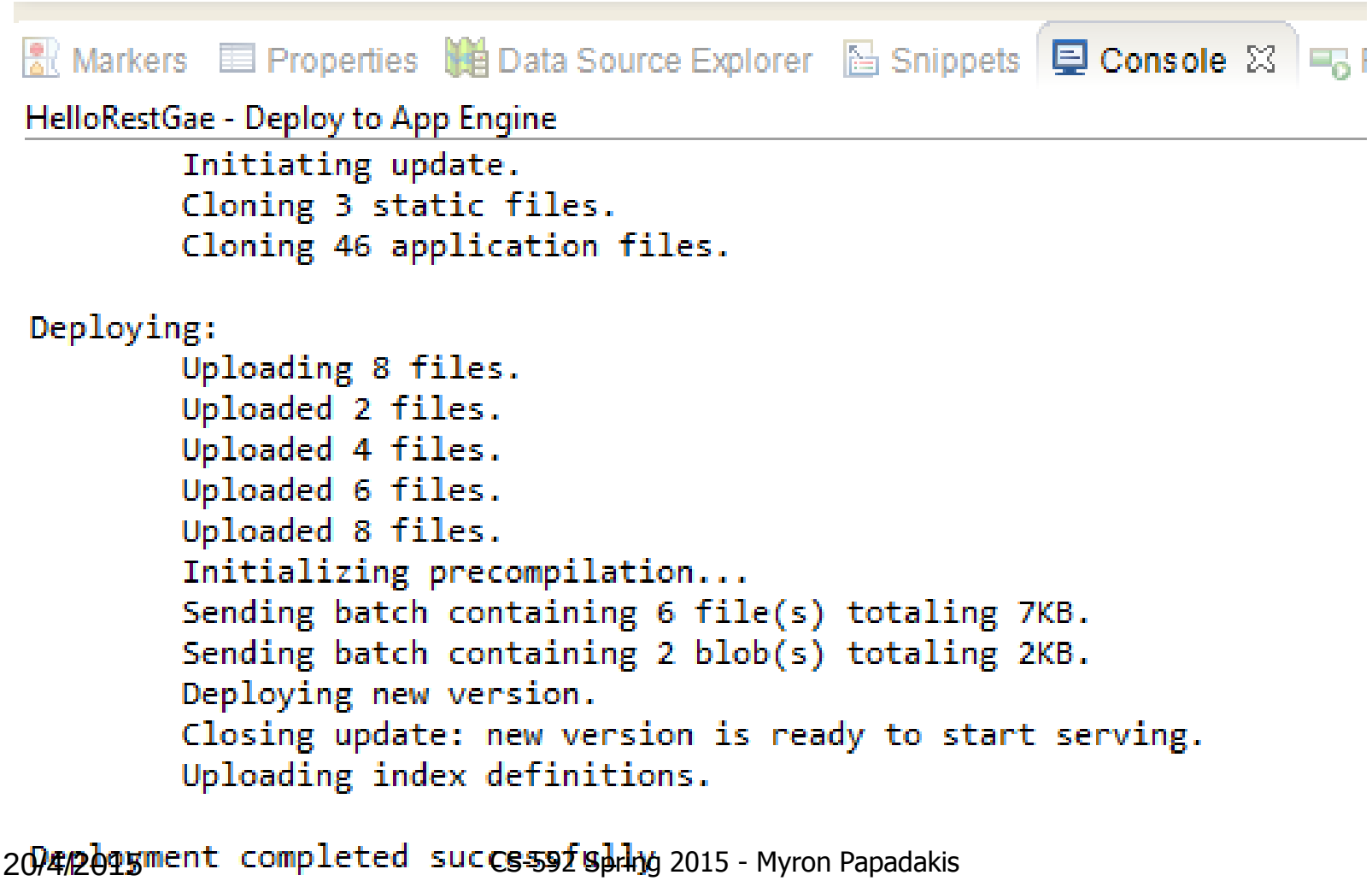
Deploying the application to Google App Engine > Credentials

- The App Engine deploy button prompts you for multiple informations:
  - username (your Google account) and password.

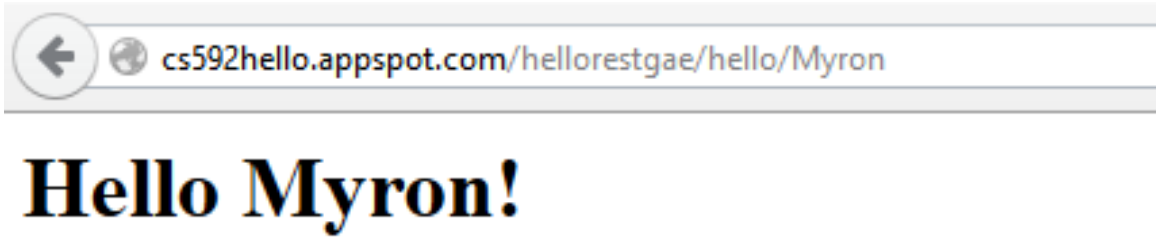
- When the deployment is complete you can access your application using the following URL: http://[your-application-id].appspot.com/url-pattern/resourcepath/methodpath

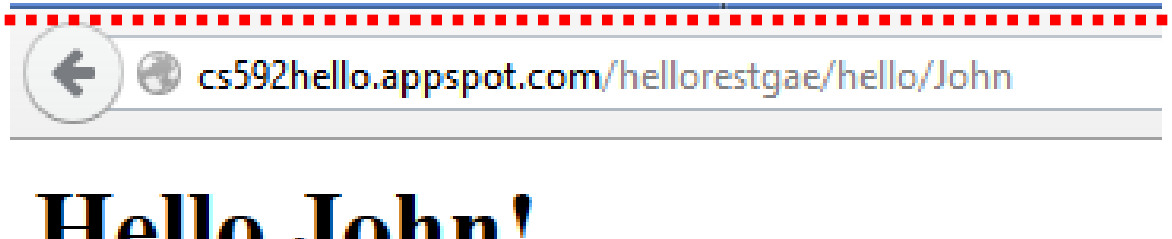
- http://cs592hello.appspot.com/hellorestgae/hello/Myron

Deploying and uploading…



GAE success ☺


cs592hello.appspot.com/hellorestgae/hello/Myron
Hello Myron!


cs592hello.appspot.com/hellorestgae/hello/John
Hello John!


cs592hello.appspot.com/hellorestgae/hello/query?name=myrpap
Hello myrpap!

GAE success ☺


cs592hello.appspot.com
JAX-RS @FormQuery Testing
FirstName : Myron
LastName : Papadakis
Submit
cs592hello.appspot.com/hellorestgae/hello/form
Hello Papadakis Myron!

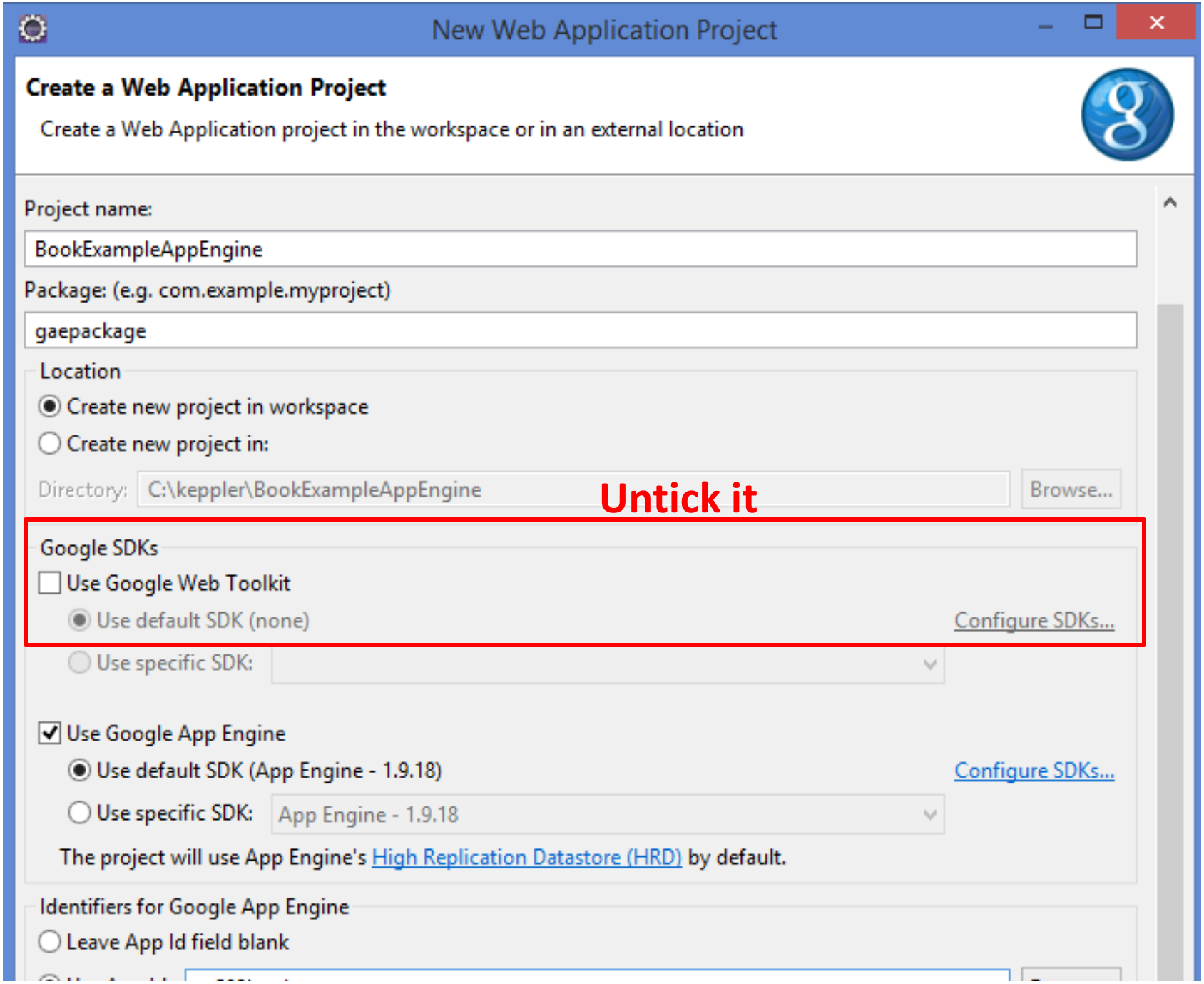
Book Example

Book Restful Web Service

- Remember the Book Example?
- See Assisting Lecture 7b to recall…
  - We have developed it in Netbeans…
  - Now we will develop it in Eclipse
- We want to deploy this example to the Google App Engine
- No need to modify existing Java code (packages etc)

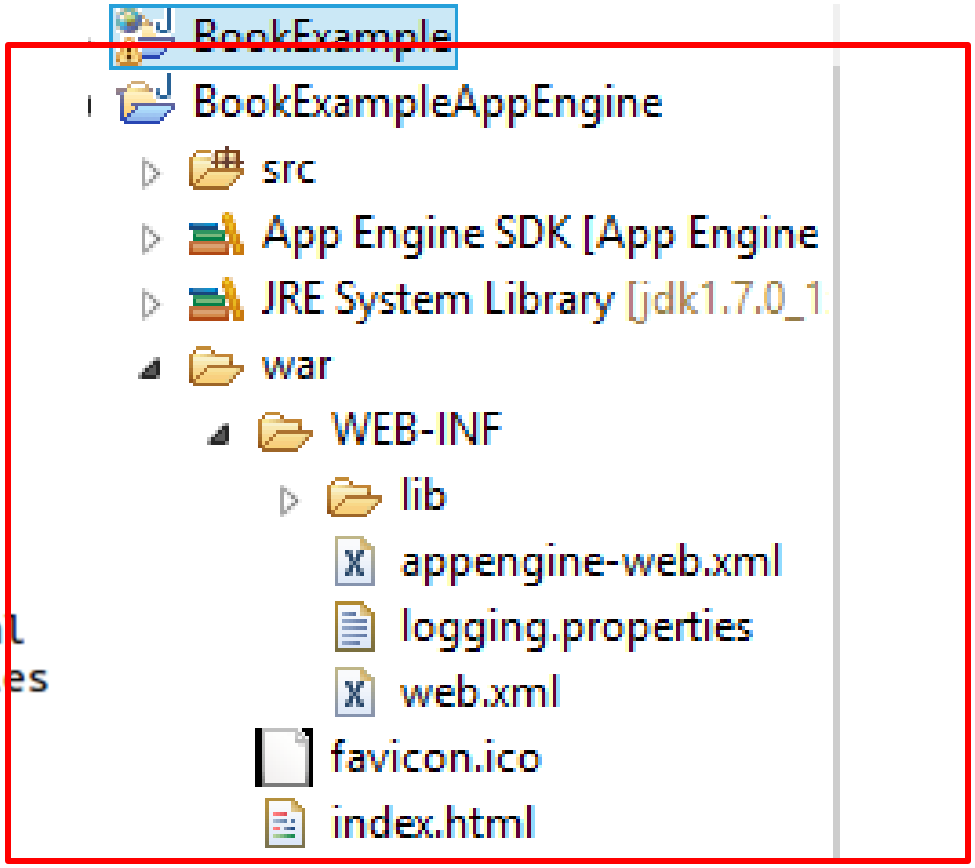
Book Restful Web Service

- For starters develop a Restful Service that
  - Returns all books
  - Returns a book with a given id
  - Adds a book
- Obviously we will need a structure for the books (list, map or whatever)
- Firstly, the Restful Web Service must create a dummy book
- Each book has
  - Id
  - Name
  - Author
  - Isbn
  - Price


New Web Application Project
Create a Web Application Project
Create a Web Application project in the workspace or in an external location
Project name:
BookExampleAppEngine
Package: (e.g. com.example.myproject)
gaepackage
Location
Create new project in workspace
Create new project in:
Directory:
C:\keppler\BookExampleAppEngine
Browse...
Untick it
Google SDKs
Use Google Web Toolkit
Use default SDK (none)
Configure SDKs...
Use specific SDK:
Use Google App Engine
Use default SDK (App Engine - 1.9.18)
Configure SDKs...
Use specific SDK:
App Engine - 1.9.18
The project will use App Engine's High Replication Datastore (HRD) by default.
Identifiers for Google App Engine
Leave App Id field blank

Creating the project

```
$ tree guestbook/
guestbook/
├── eclipse-launch-profiles
│   ├── DevAppServer.launch
│   └── UpdateApplication.launch
├── nbactions.xml
├── pom.xml
├── README.md
└── src
    ├── main
    │   ├── java
    │   └── webapp
    │       └── WEB-INF
    │           ├── appengine-web.xml
    │           ├── logging.properties
    │           └── web.xml
    └── test
```



*Book.java | BookResource.java | web.xml

```
1  package gaepackage;
2
3  import javax.xml.bind.annotation.XmlRootElement;
4
5
6  @XmlRootElement(name = "book")
7  public class Book {
8
9      private int id;
10     private String bookName;
11     private String bookAuthor;
12     private String bookISBN;
13     private double price;
14
15     public int getId() {
16         return id;
17     }
18
19     public void setId(int id) {
20         this.id = id;
21     }
22
23     public String getBookName() {
24         return bookName;
25     }
26
27     public void setBookName(String bookName) {
28         this.bookName = bookName;
29     }
30
31     public String getBookAuthor() {
32         return bookAuthor;
```

BookResource Class

Book.java | BookResource.java | web.xml

```
22   * myrpap@gmail.com
23   */
24  @Path("bookresource")
25  @Singleton
26  public class BookResource {
27
28      @Context
29      private UriInfo context;
30      private TreeMap<Integer, Book> bookMap = new TreeMap<Integer, Book>();
31
32      public BookResource() {
33          Book book = new Book();
34          book.setBookAuthor("Bhaveh Thaker");
35          book.setBookName("Introduction to RESTful Web Services");
36          book.setBookISBN("ISBN 10: 0-596-52926-0");
37          addBook(book);
38      }
39
40      @GET
41      @Path("books")
42      @Produces(MediaType.APPLICATION_XML)
43      public List<Book> getBooks() {
```

BookResource Class

```
@GET
@Path("books")
@Produces(MediaType.APPLICATION_XML)
public List<Book> getBooks() {
    List<Book> books = new ArrayList<Book>();
    books.addAll(bookMap.values());
    return books;
}

@GET
@Path("{id}")
public Book getBook(@PathParam("id") int bookId) {
    return bookMap.get(bookId);
}

@POST
@Path("add")
public String addBook(Book book) {
    int id = bookMap.size();
```

```
@POST
@Path("add")
public String addBook(Book book) {
    int id = bookMap.size();
    if (book == null) {
        return "Book is null";
    } else {
        book.setId(id);
        bookMap.put(id, book);
        return "Book \"" + book.getBookName() + "\" added with Id "
                + id + " size=" + bookMap.size();
    }

}
@DELETE
@Path("delete/{delId}")
public String deleteBook(@PathParam("delId") int bookid) {
    if (bookMap.containsKey(bookid)) {
        bookMap.remove(bookid);
        return "Book successfully deleted";
    } else {
        return "No such key";
    }
}

@PUT
@Path("update/{bookid}/{price}")
public String updateBook(@PathParam("bookid") int bookid,
        @PathParam("price") double price ) {
```

Configuration > Default generated xml file

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  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
7      <servlet>
8          <servlet-name>BookExampleAppEngine</servlet-name>
9          <servlet-class>gaepackage.BookExampleAppEngineServlet</servlet-class>
10     </servlet>
11     <servlet-mapping>
12         <servlet-name>BookExampleAppEngine</servlet-name>
13         <url-pattern>/bookexampleappengine</url-pattern>
14     </servlet-mapping>
15     <welcome-file-list>
16         <welcome-file>index.html</welcome-file>
17     </welcome-file-list>
18 </web-app>
```

Configuration > Modified web.xml

web.xml | web.xml | BookResource.java | persistence.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  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2
7      <servlet>
8          <servlet-name>BookExampleAppEngine</servlet-name>
9          <servlet-class>org.glassfish.jersey.servlet.ServletCo
10         <init-param>
11             <param-name>jersey.config.server.provider.package
12             <param-value>gaepackage</param-value>
13         </init-param>
14     </servlet>
15     <servlet-mapping>
16         <servlet-name>BookExampleAppEngine</servlet-name>
17         <url-pattern>/bookgae/*</url-pattern>
18     </servlet-mapping>
19     <welcome-file-list>
20         <welcome-file>index.html</welcome-file>
21     </welcome-file-list>
22 </web-app>
23
```

- This servlet that will answer all request the /bookgae/* URL and redirect them to classes representing RESTfull services
- The configuration parameter com.sun.jersey.config.property.packages is used by Jersey to list the packages where REST services implementation are located.
- You can put as many package as you need to, you just need to separate the package names by a ;
- You do not need the welcome file too (since you will not use a servlet)

Optional (If you want to use servlets)

```
<servlet>
    <servlet-name>BookExampleAppEngineServlet</servlet-name>
    <servlet-class>gaepackage.BookExampleAppEngineServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>BookExampleAppEngineServlet</servlet-name>
    <url-pattern>/BookExampleAppEngineServlet</url-pattern>
</servlet-mapping>
```

```
<!-- but

<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Hello App Engine</title>
  </head>

  <body>
    <h1>Hello App Engine!</h1>

    <table>
      <tr>
        <td colspan="2" style="font-weight:bold;">Available Servlets:</td>
      </tr>
      <tr>
        <td><a href="BookExampleAppEngineServlet">BookExampleAppEngine</a></td>
      </tr>
    </table>
  </body>
</html>
```

Deploying the application to Google App Engine

- Before deploying the application you need
  - to register a new application in Google App Engine using the Administration Console (see next slide).
  - In my example I have used "**cs592book**" as Application ID.

- You can easily now deploy the application to Google App Engine by clicking on the "Deploy App Engine Project" button available in the Eclipse toolbar.

- To be able to deploy your application to Google App Engine, you need to check that your application can be registered, **the application ID is stored in the WEB-INF/lib/appengine-web.xml.**

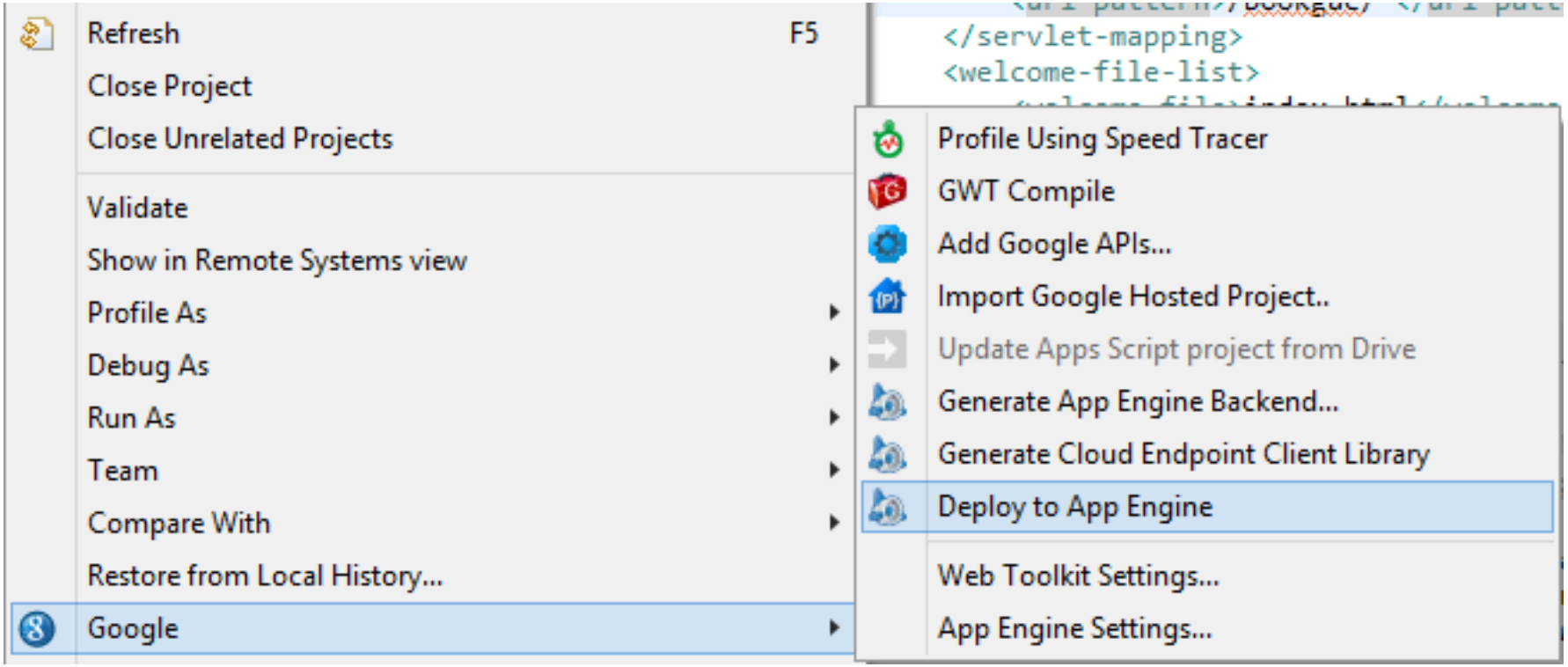
Deploying the application to Google App Engine > appengine.web-xml

main.js | Tomcat v7.0... | web.xml | BookResource... | appengine-we...

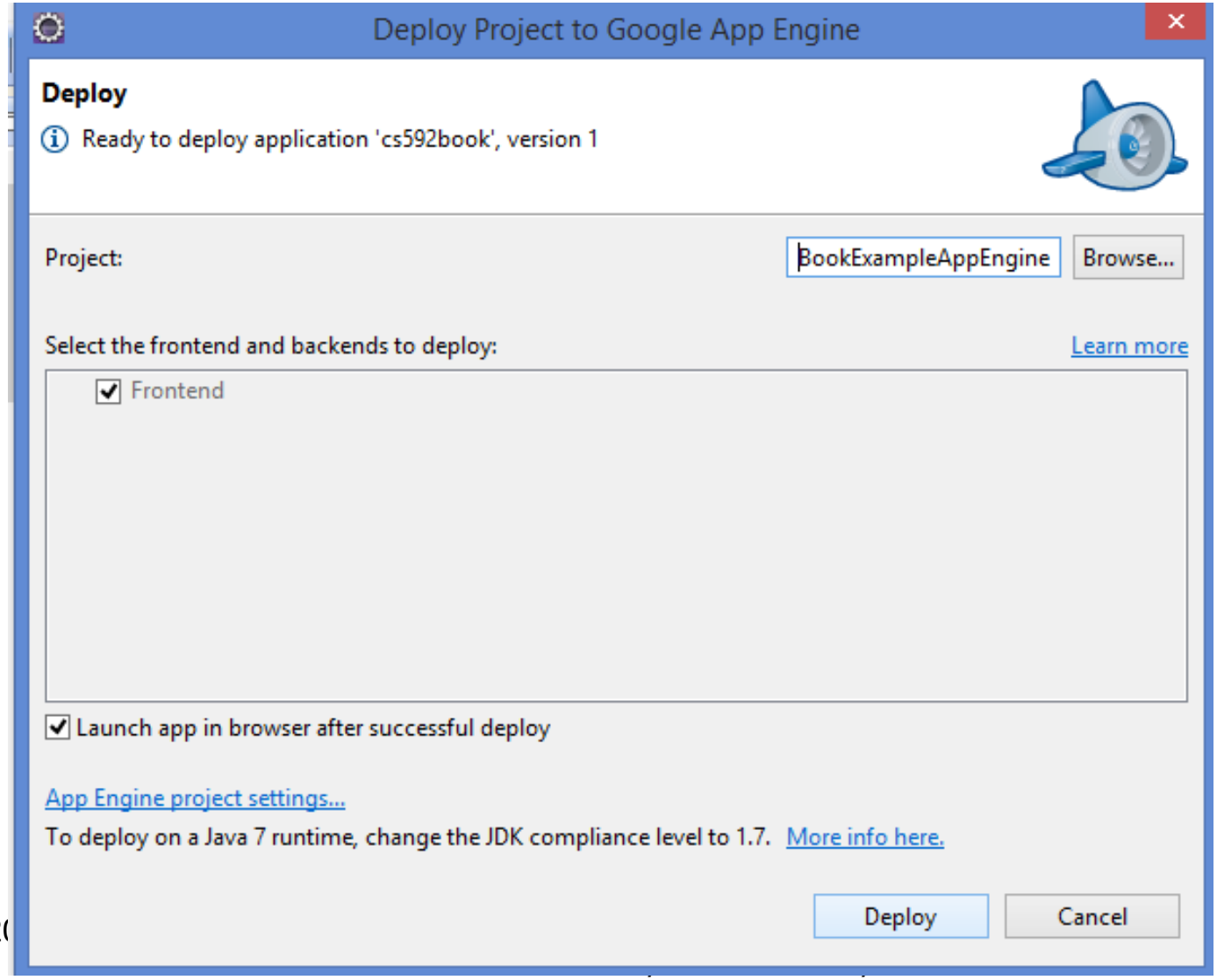
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
3    <application>cs592book</application>
4    <version>1</version>
5
6    <!--
7      Allows App Engine to send multiple requests to one instance in parallel:
8    -->
9    <threadsafe>true</threadsafe>
10
11   <!-- Configure java.util.logging -->
12   <system-properties>
13     <property name="java.util.logging.config.file" value="WEB-INF/logging.properties"/>
14   </system-properties>
15
16   <!--
17     HTTP Sessions are disabled by default. To enable HTTP sessions specify:
18
19       <sessions-enabled>true</sessions-enabled>
20
21     It's possible to reduce request latency by configuring your application to
22     asynchronously write HTTP session data to the datastore:
23
24       <async-session-persistence enabled="true" />
25
```

Deploying the application to Google App Engine

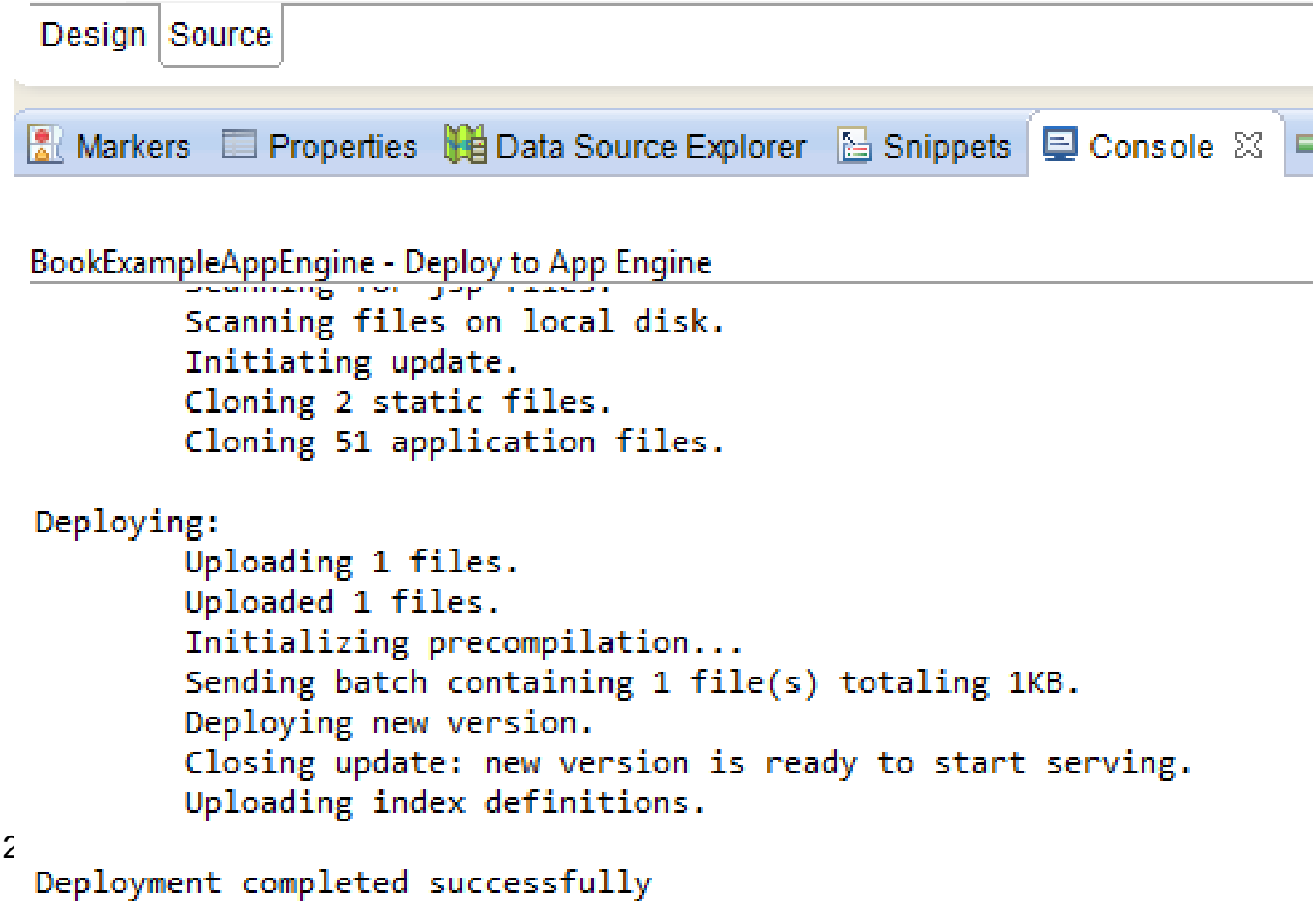
- Right click on the project



Deploying the application to Google App Engine

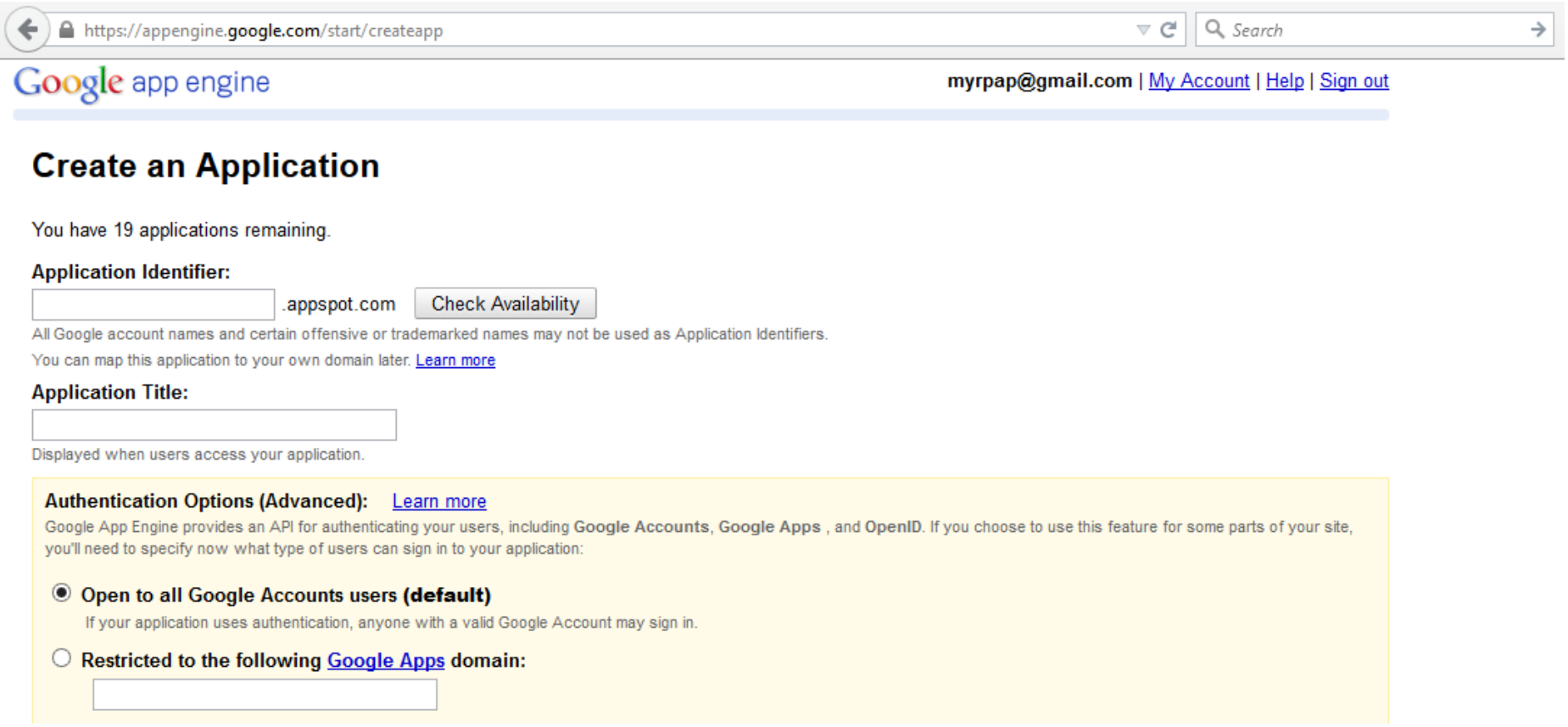


Deploying the application to Google App Engine > Uploading



Google App Engine > Create Application

25 Applications for free ☺

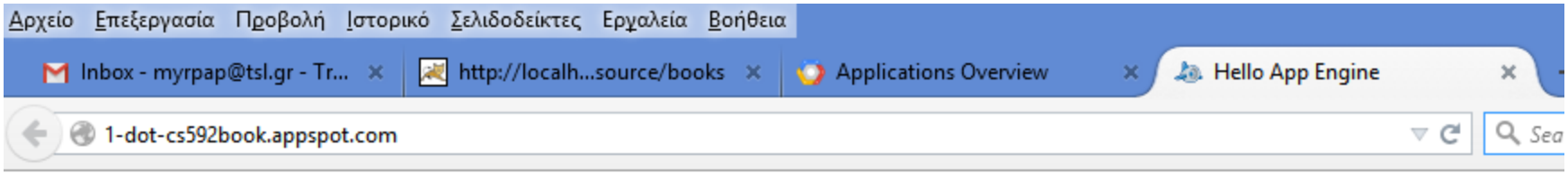


Deploying the application to Google App Engine > Credentials

- The App Engine deploy button prompts you for multiple informations:
  - username (your Google account) and password.

- When the deployment is complete you can access your application using the following URL: http://[your-application-id].appspot.com/url-pattern/resourcepath/methodpath

- http://cs592book.appspot.com/bookgae/bookresource/books

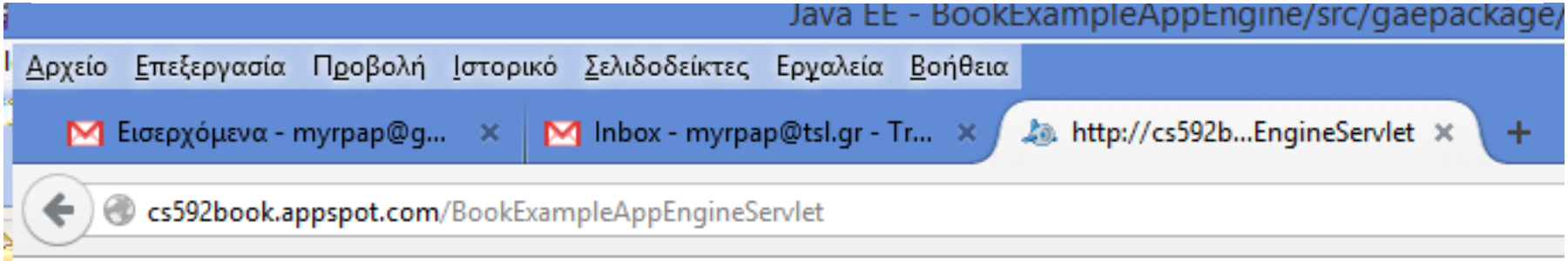


Hello App Engine!

Available Servlets:

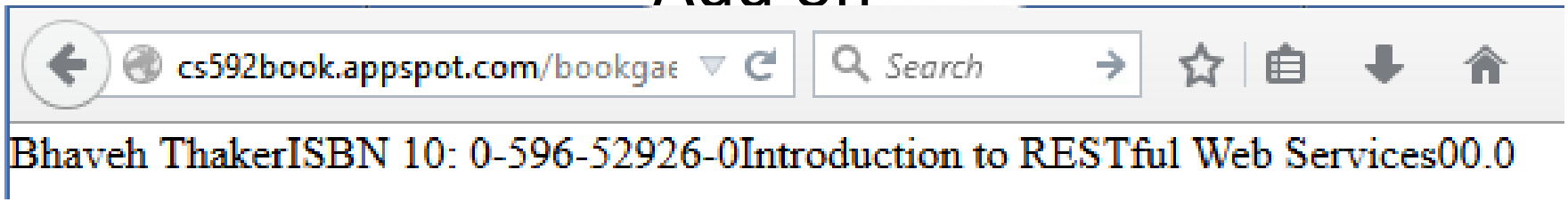
BookExampleAppEngine

Initial page as set from the index.html (which is the welcome file)

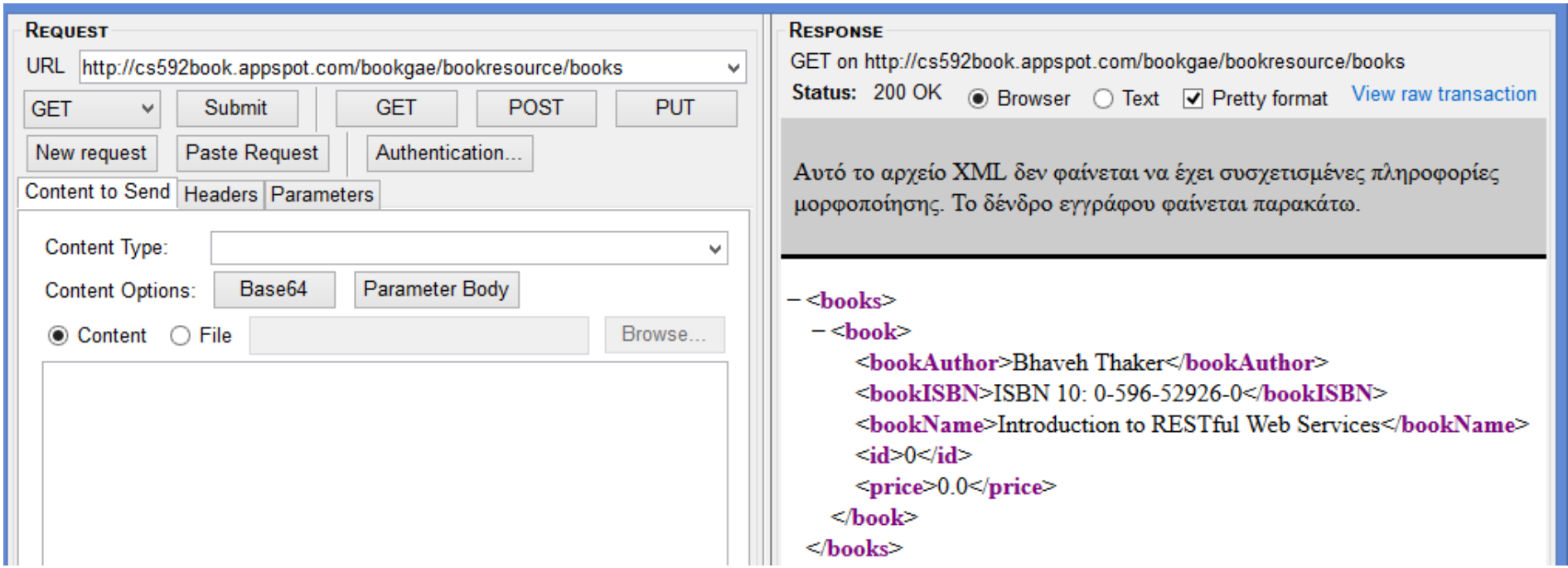



```
1  package gaepackage;
2
3  import java.io.IOException;
5
6  @SuppressWarnings("serial")
7  public class BookExampleAppEngineServlet extends HttpServlet {
8      public void doGet(HttpServletRequest req, HttpServletResponse resp)
9              throws IOException {
10         resp.setContentType("text/plain");
11         resp.getWriter().println("Hello, world");
12     }
13 }
14
```

Testing on HttpRequester Mozilla Add on

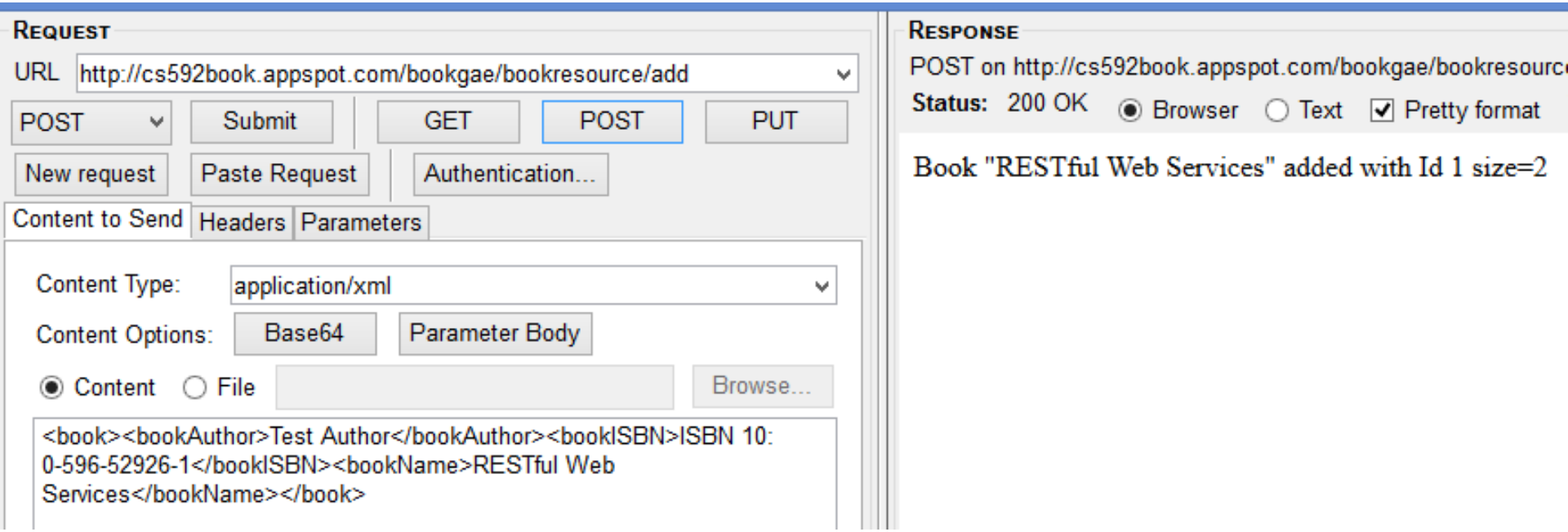


- http://cs592book.appspot.com/bookgae/bookresource/books



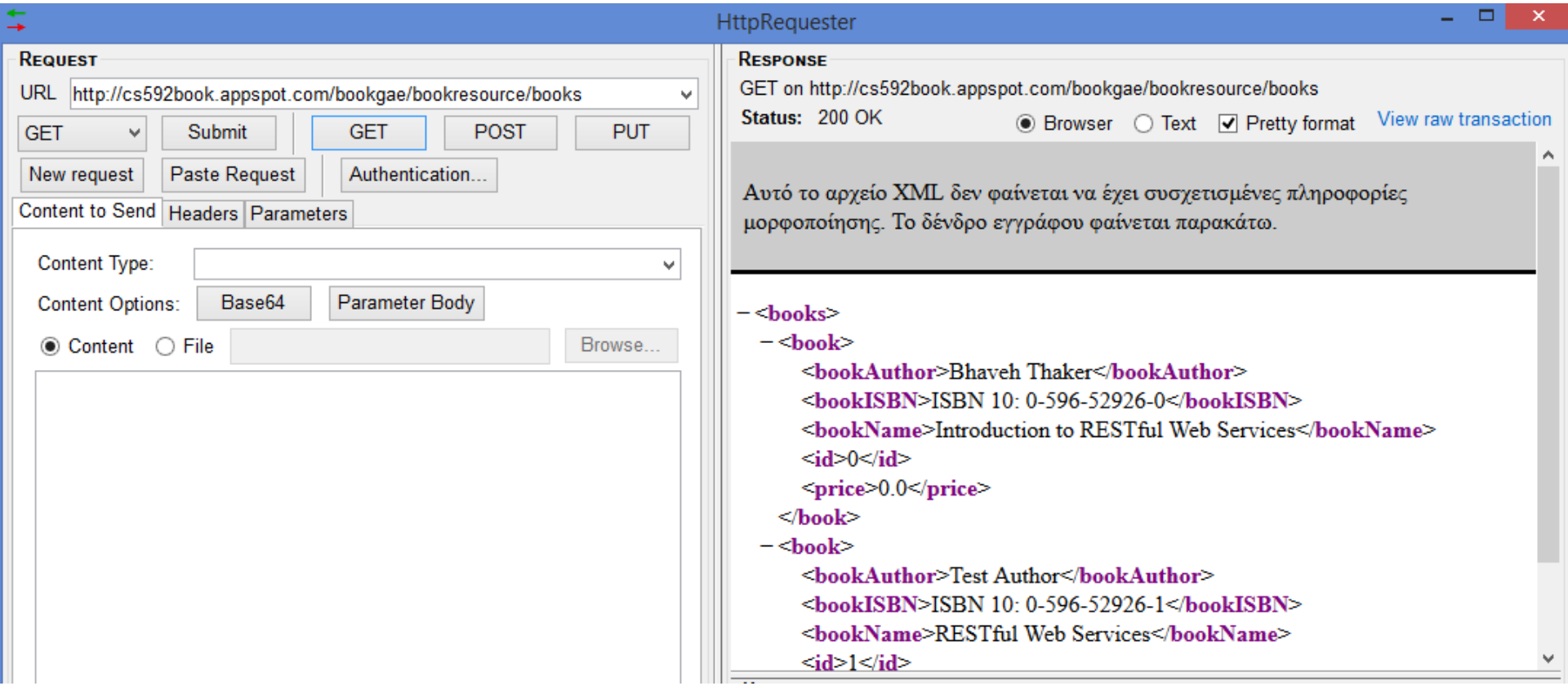
Post Example

- Post a Book (in XML format)



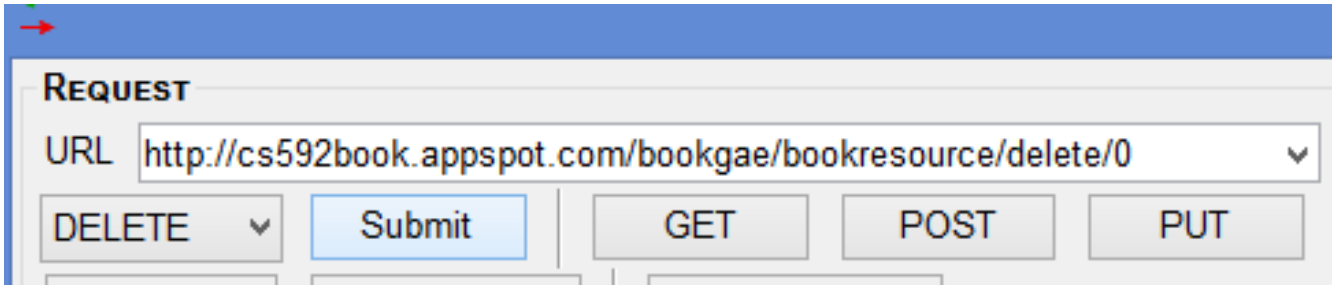
Get Example > Book is added

- The second book has been added successfully

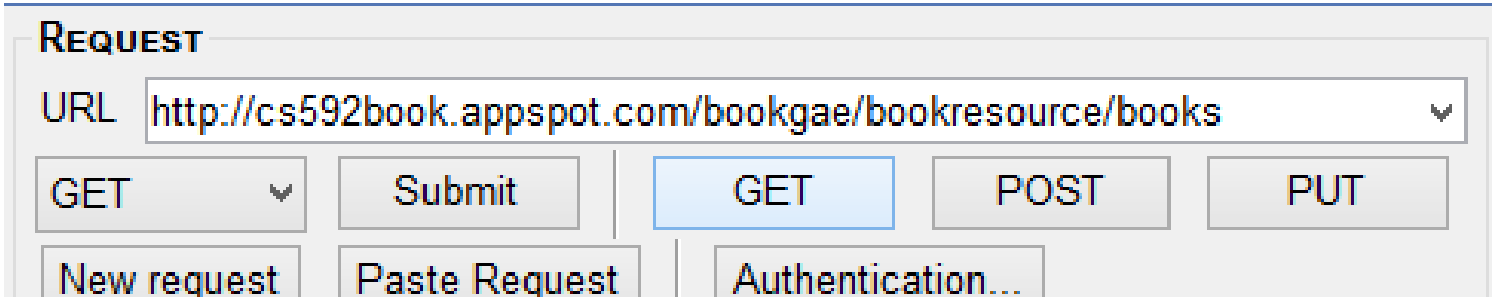


Delete a book

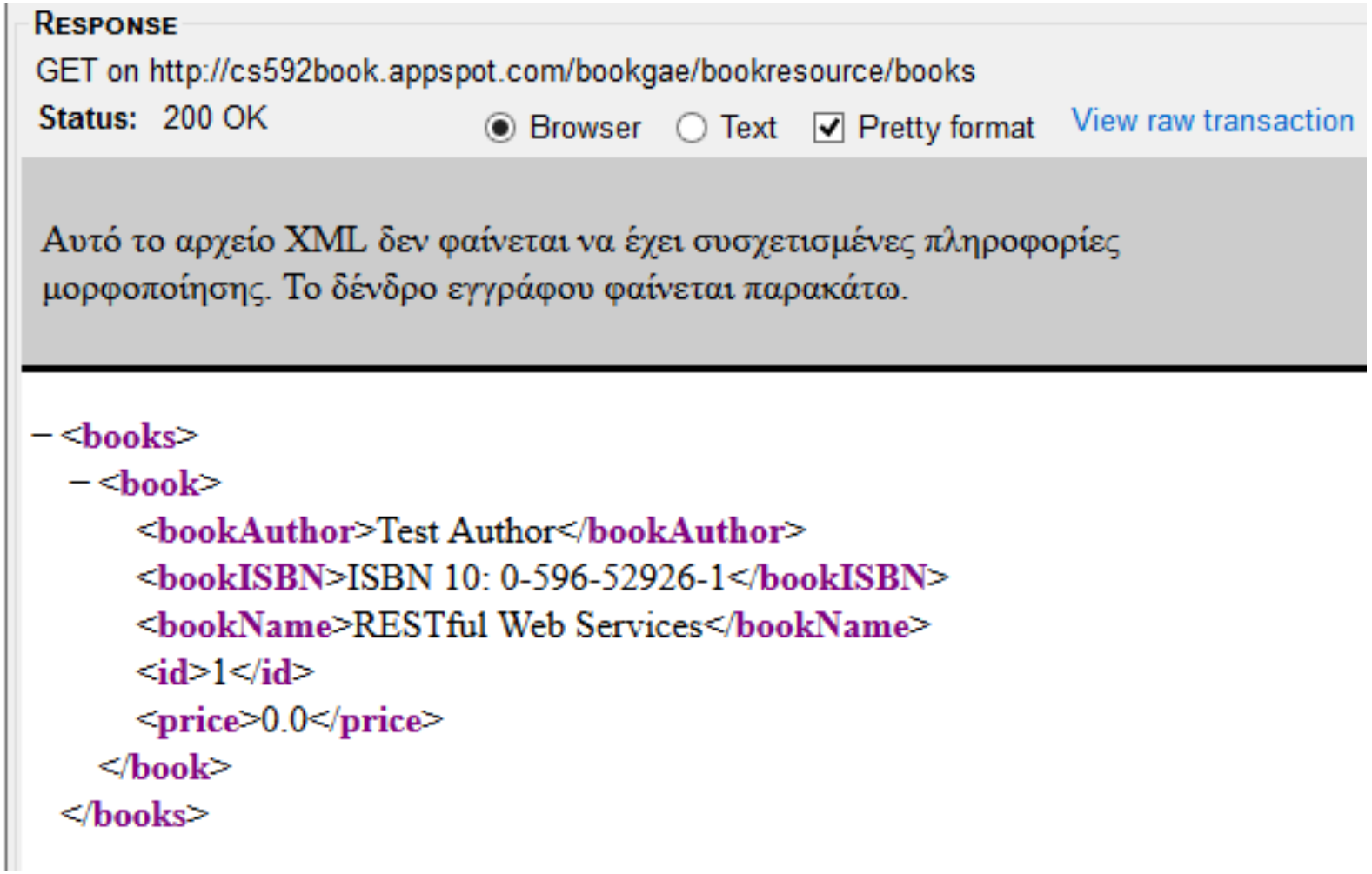
- Delete the first book



- Get back all books (to check if deleted)



Get books (after deletion)



Google Gson


google-gson
A Java library to convert JSON to Java objects and vice-versa

GSON

- Gson is a Java library that can be used
  - to convert Java Objects into their JSON representation.
  - It can also be used to convert a JSON string to an equivalent Java object.

    https://code.google.com/p/google-gson/

    http://www.studytrails.com/java/json/java-google-json-introduction.jsp

Gson Example (for Book)

```
1  package gson;
2
3  import com.google.gson.Gson;
4  import gaepackage.Book;
5
6  public class GSONDemo {
7
8      public static void convertJavaToJSON(){
9          Book book = new Book();
.0         book.setBookAuthor("Bhaveh Thaker");
.1         book.setBookName("Introduction to RESTful Web Services");
.2         book.setBookISBN("ISBN 10: 0-596-52926-0");
.3         Gson gson = new Gson();
.4         System.out.println(gson.toJson(book));
.5     }
.6     public static void convertJSONtoJava(){
.7         Gson gson = new Gson();
.8
.9         System.out.println(
!0 gson.fromJson("{'id':1,'bookName':'RESTful Java Web Services','bookAuth
!1         Book.class));
!2     }
```

Gson Example (for Book)

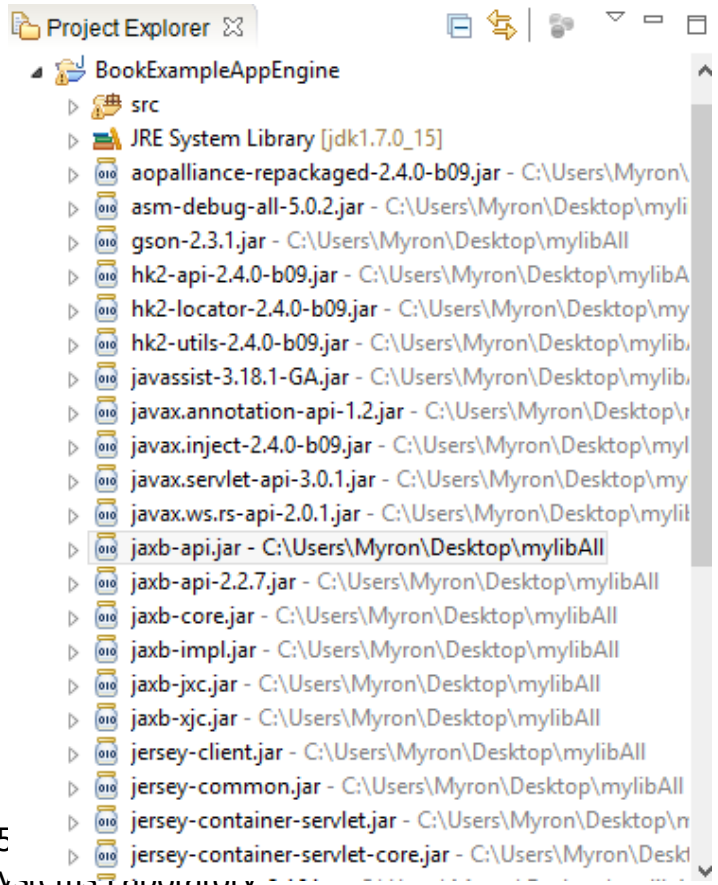
```
23    public static void main(String[] args) {
24        convertJavaToJSON();
25        convertJSONtoJava();
```

Markers | Properties | Data Source Explorer | Snippets | Console | Progress | Remote Systems | His

```
<terminated> GSONDemo [Java Application] C:\Program Files\Java\jdk1.7.0_15\bin\javaw.exe (18 Απρ 2015 - 10:51:33 π.μ.)
{"id":0,"bookName":"Introduction to RESTful Web Services","bookAuthor":"Bhaveh Thaker","bookISBN":"IS
Book [id=1, bookName=RESTful Java Web Services, bookAuthor=bookISBN,bookISBN=1847196462, price=50.0]
```

Add Gson Code

- Right click on project->Property->java build path->Add jars and then go to src->jars->gson-2.3.1.jar.

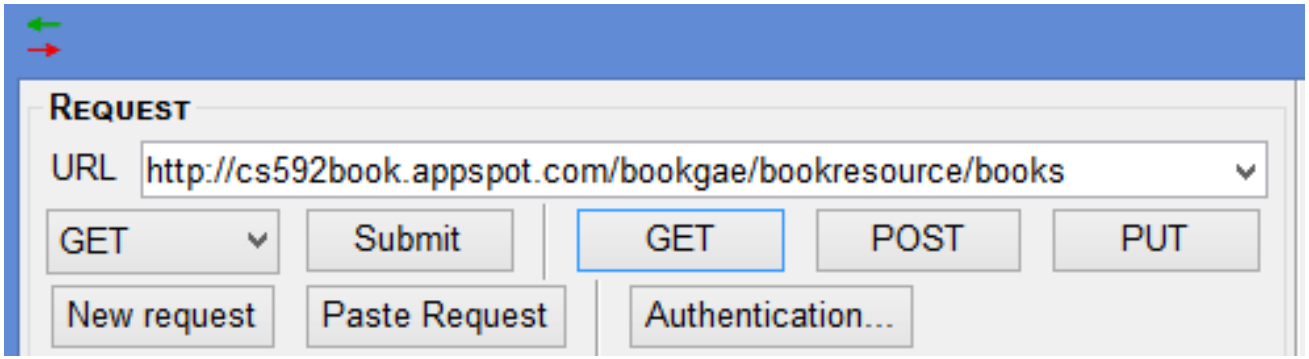


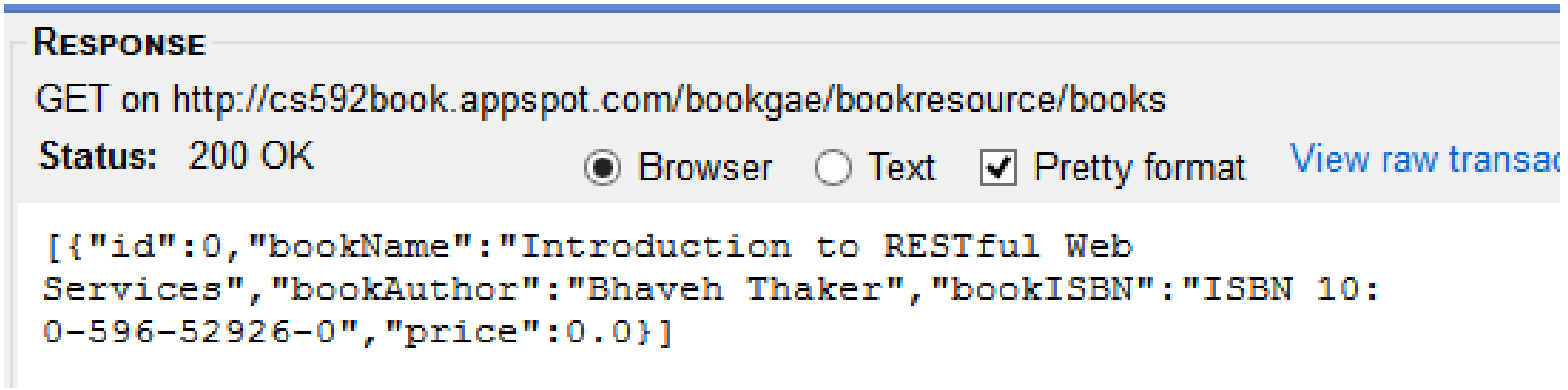
BookExample with GSON > get all books

```
private TreeMap<Integer, Book> bookMap = new TreeMap<Integer, Book>();

public BookResource() {
    Book book = new Book();
    book.setBookAuthor("Bhaveh Thaker");
    book.setBookName("Introduction to RESTful Web Services");
    book.setBookISBN("ISBN 10: 0-596-52926-0");
    addBook(book);

}

@GET
@Path("books")
@Produces(MediaType.APPLICATION_JSON)
public String getBooks() {
    List<Book> books = new ArrayList<Book>();
    books.addAll(bookMap.values());
    Gson gson = new Gson();
    String jsonArray = gson.toJson(books);
    return jsonArray;
}
```

BookExample with GSON > get all books

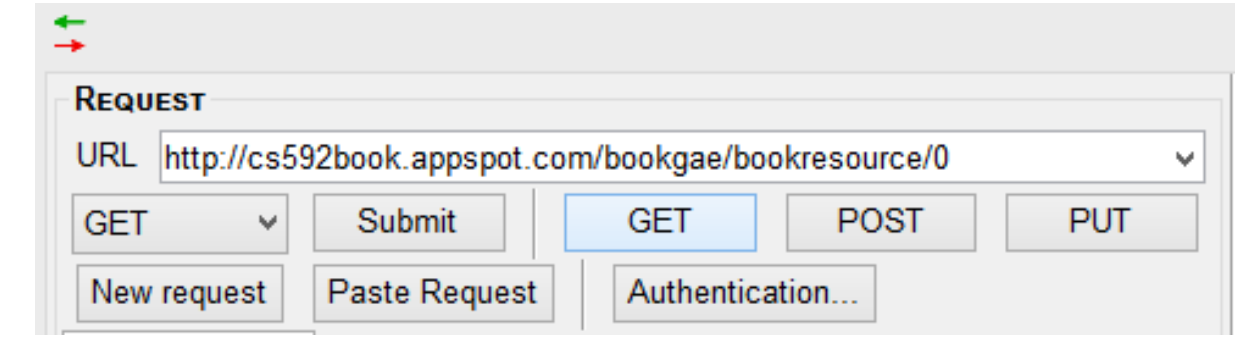
- Returned in JSON format

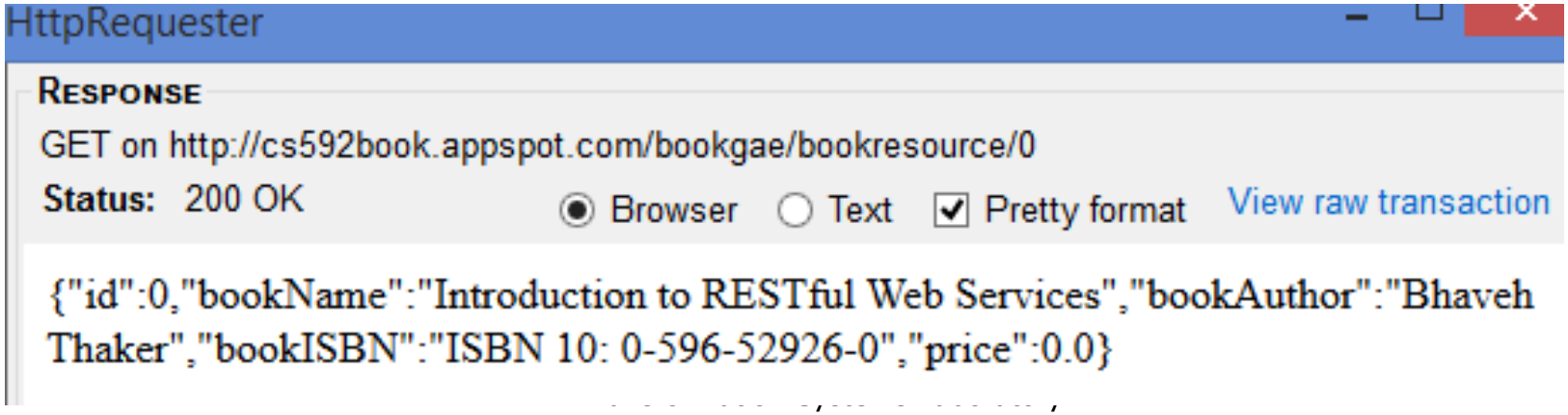




BookExample with GSON > get a book

```
@GET
@Path("{id}")
public String getBook(@PathParam("id") int bookId) {
    Gson gson = new Gson();
    String jsonArray = gson.toJson(bookMap.get(bookId));
    return jsonArray;
}
```





References

- http://howtodoinjava.com/2014/06/17/google-gson-tutorial-convert-java-object-to-from-json/

Τέλος Ενότητας

Χρηματοδότηση

•Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

•Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

•Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/

•Ως **Μη Εμπορική** ορίζεται η χρήση:

–που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο

–που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο

–που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

•Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

.

Σημείωμα Αναφοράς