Distributed Systems

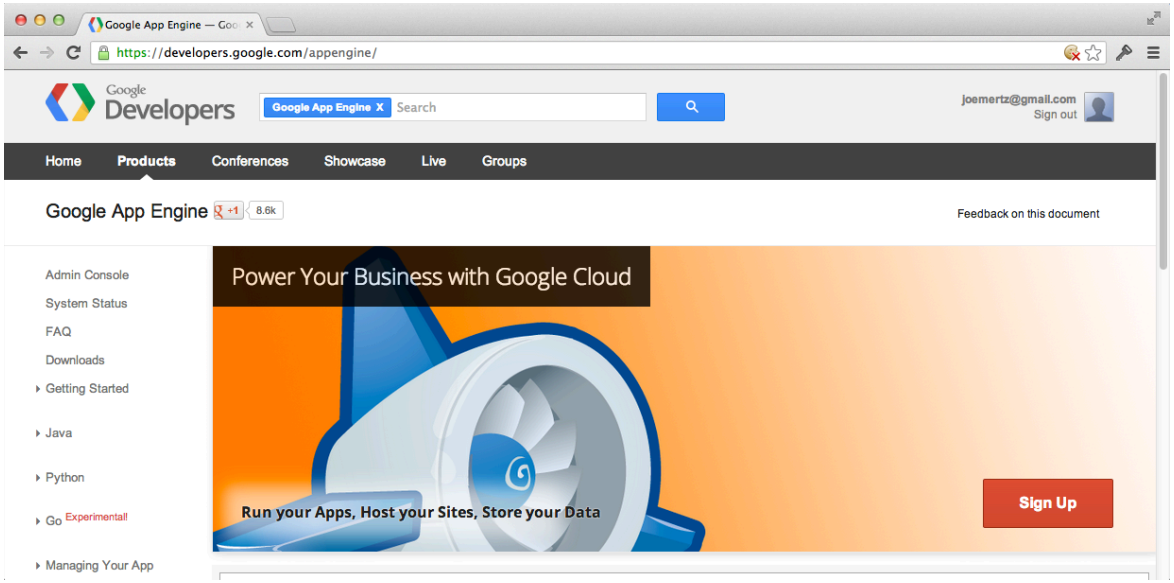
Cloud Computing via Google App Engine

Google App Engine

- Free
  - To some threshold, for development
- Provides a Java Web Container
- Provides a simple development environment
  - As an Eclipse plugin
- It is a good example of a PaaS to experiment with
  - And use in a homework project.

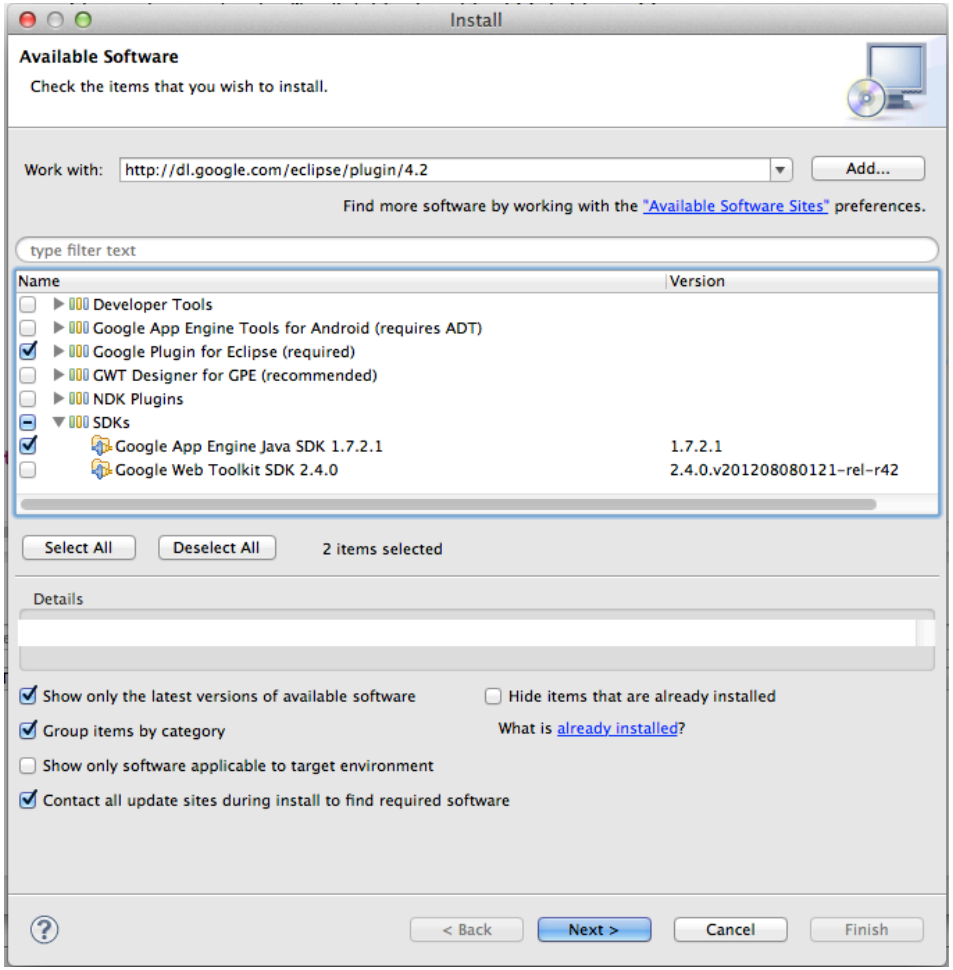
Before today* you should have

- Gone to https://developers.google.com/appengine/docs/java/gettingstarted/introduction
- Signed up for a GAE account
- Downloaded and installed the App Engine plugin for Eclipse



*See the email I sent Saturday.

Installing Eclipse GAE Plugin



When installing, choose:

- Google Plugin for Eclipse
- Google App Engine Java SDK

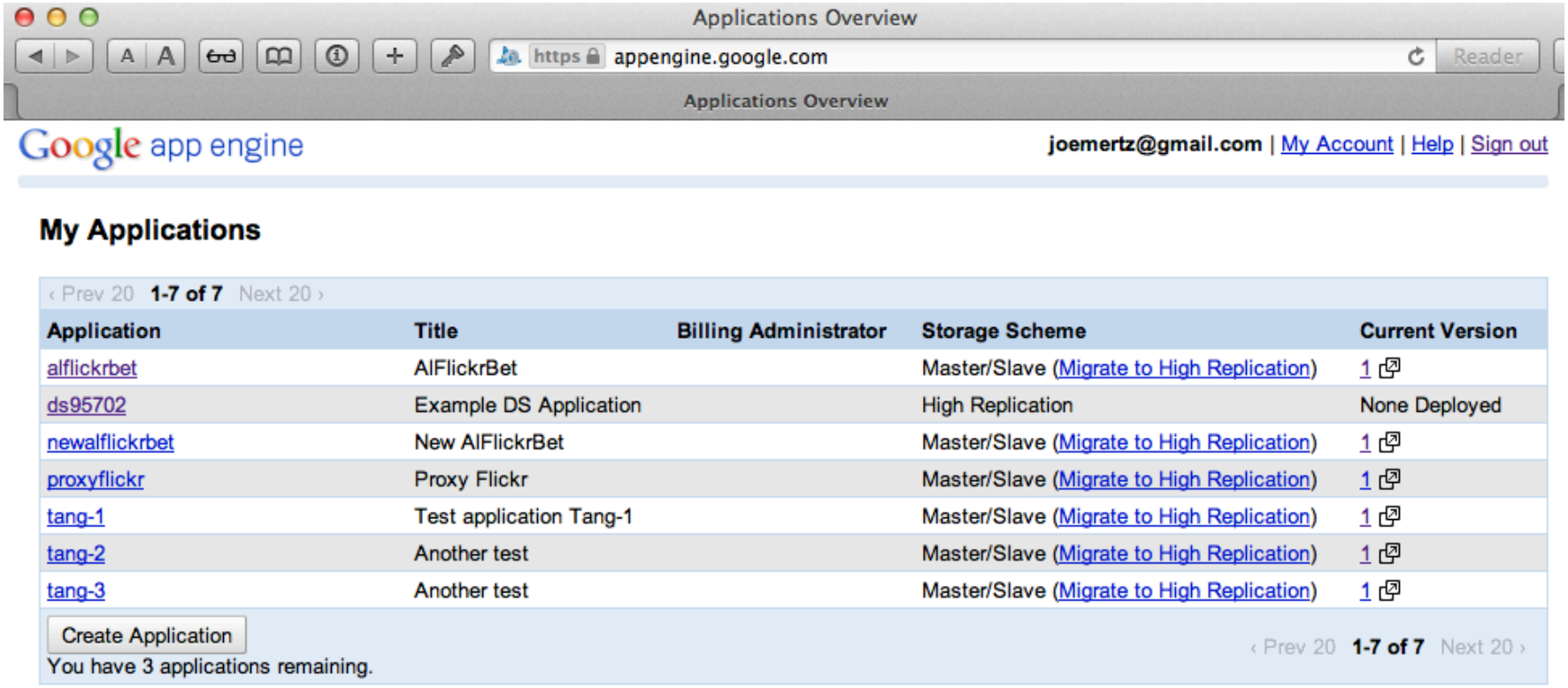
Hints: Installing under Windows

- Make sure to run Eclipse.exe as an Administrator before installing.
- If installation hangs after a certain point (such as 48% of the process), disable Anti-virus software (such as AVG) temporarily and then proceed.

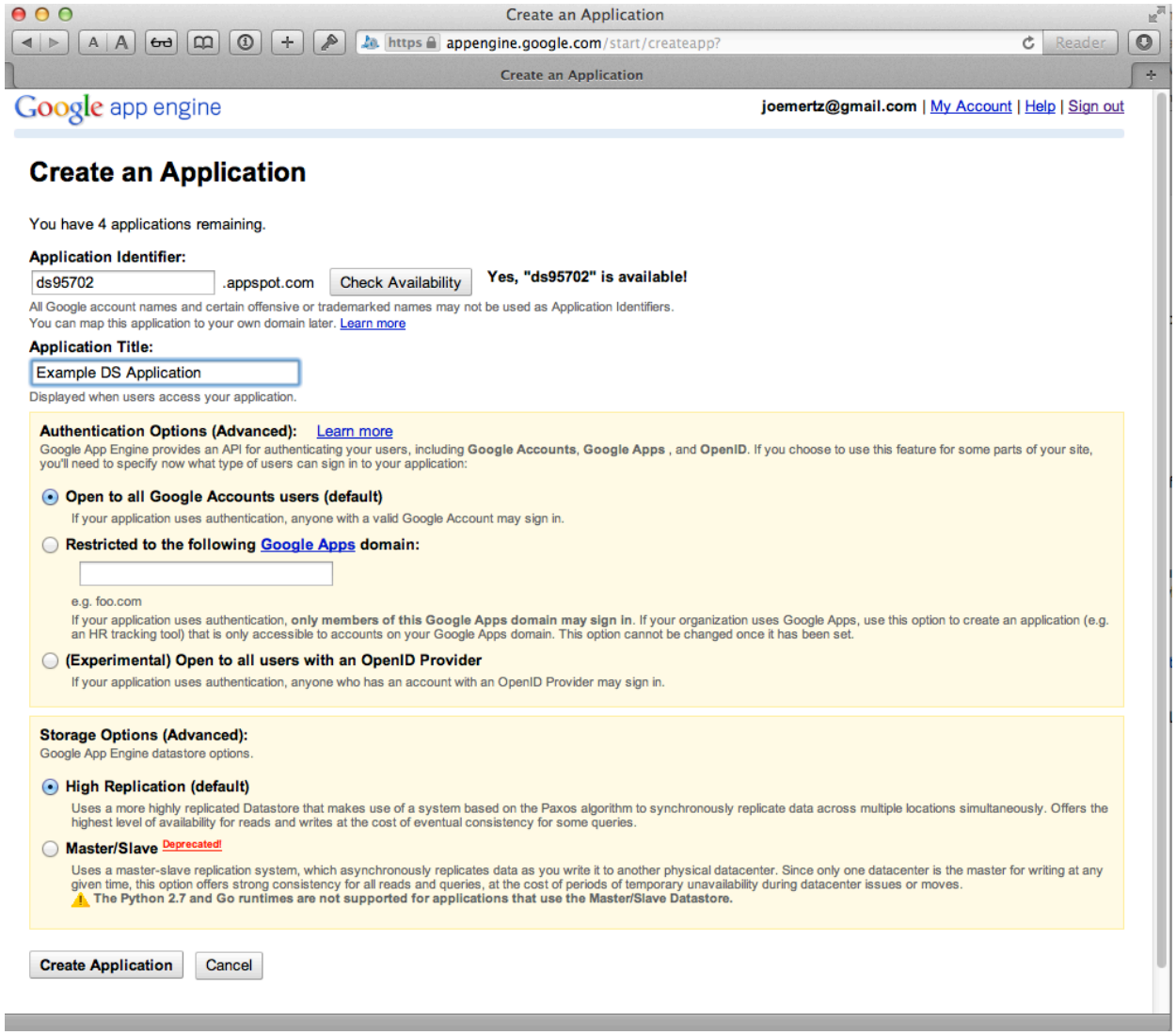
App Engine Development Environment

- I will walk through a simple development cycle
- Show working on localhost
- Identify the deploy button
- Show working on Google App Engine

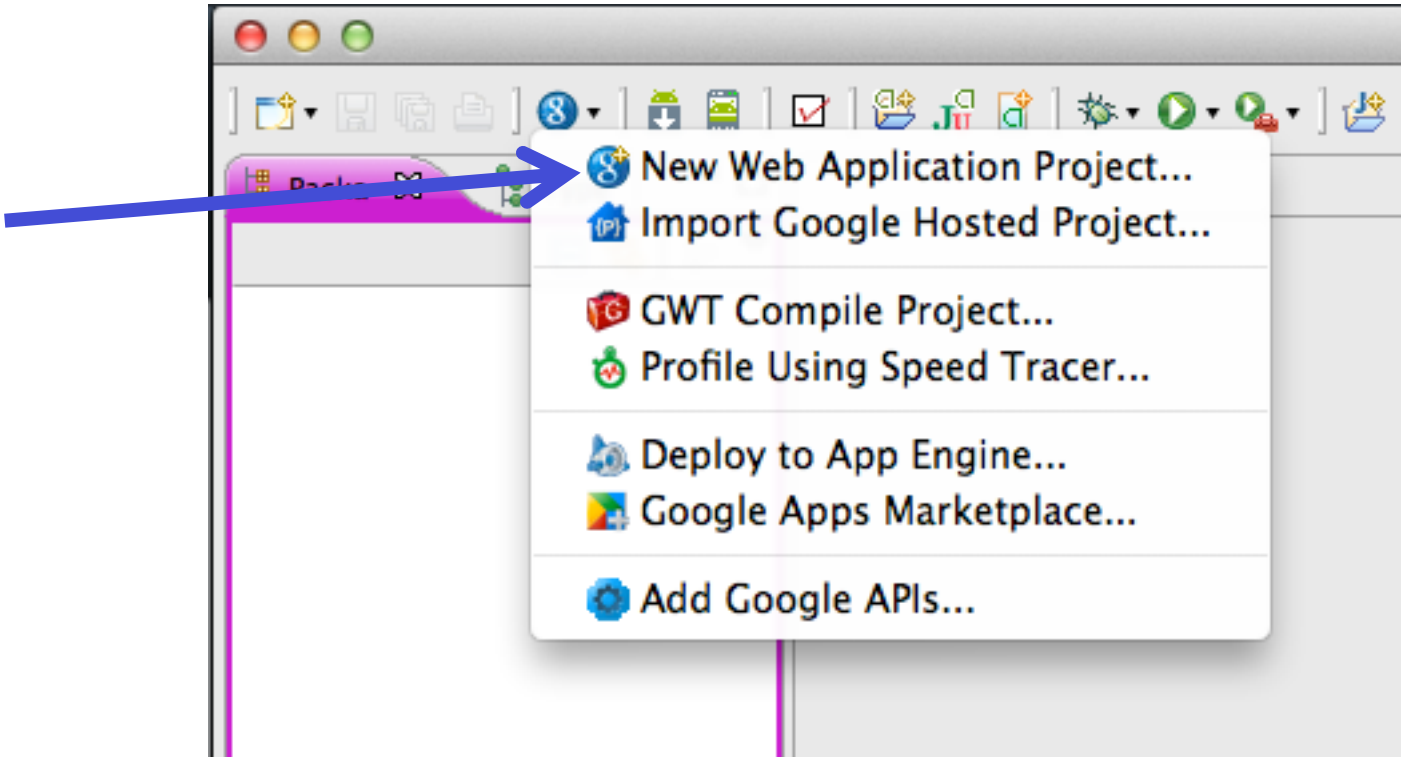
My Applications



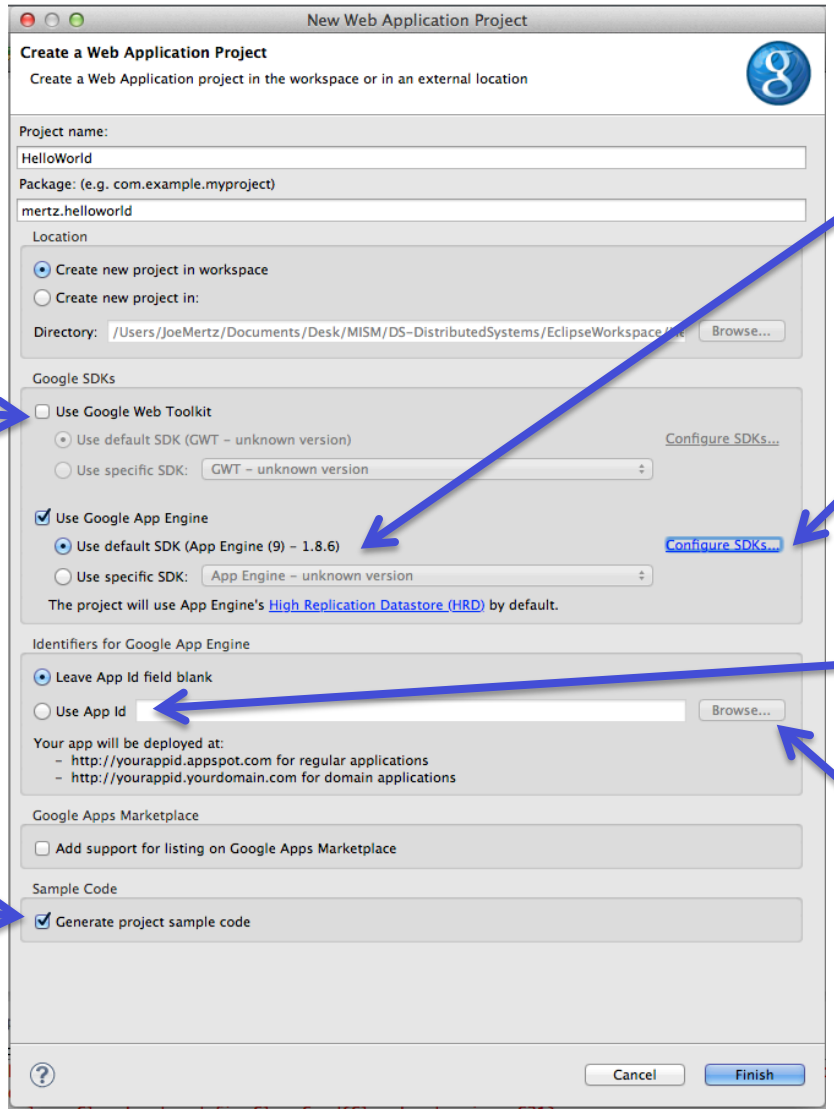
Create an Application



Eclipse: New Web Application Project



Create a Web Application Project



Unselect for we are not using Google Web Toolkit

If default is not 1.9.0 (the one you just installed)

Click on Configure SDKs and choose 1.9.0

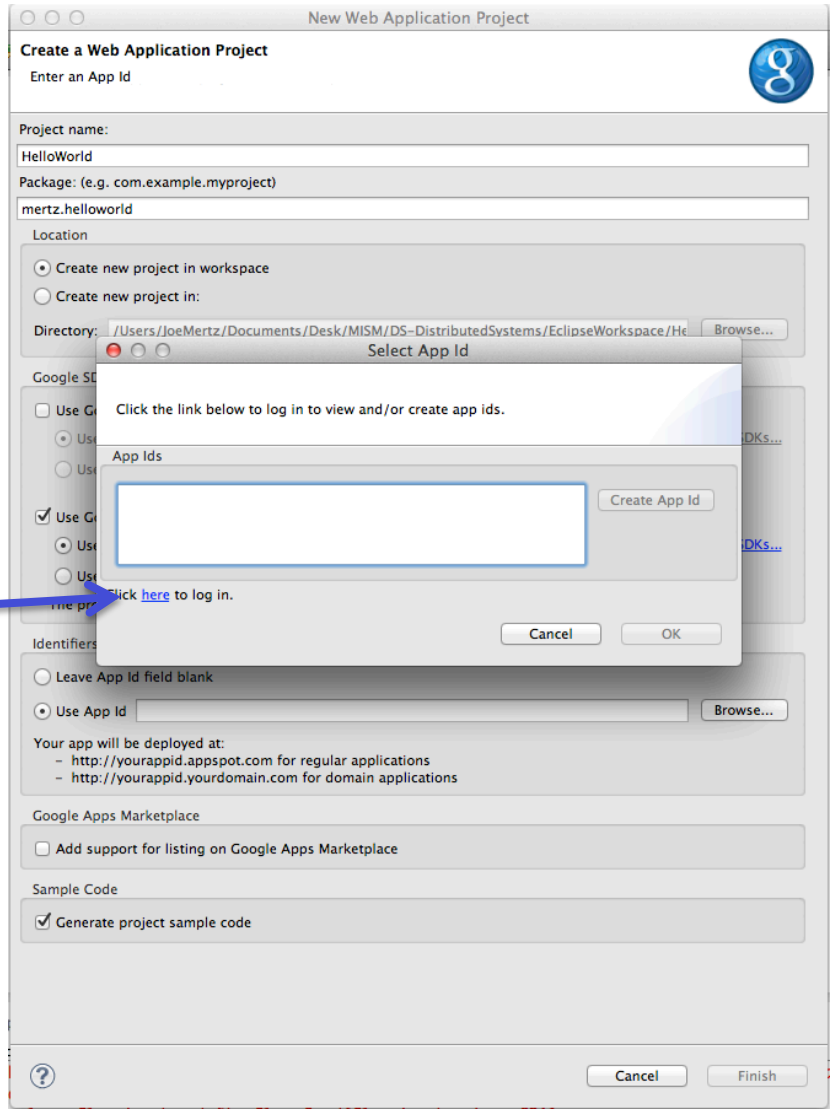
Select "Use App ID" and click on Browse… to set the Application ID you created earlier.

(See next slide)

Select (Provides a good starting place)

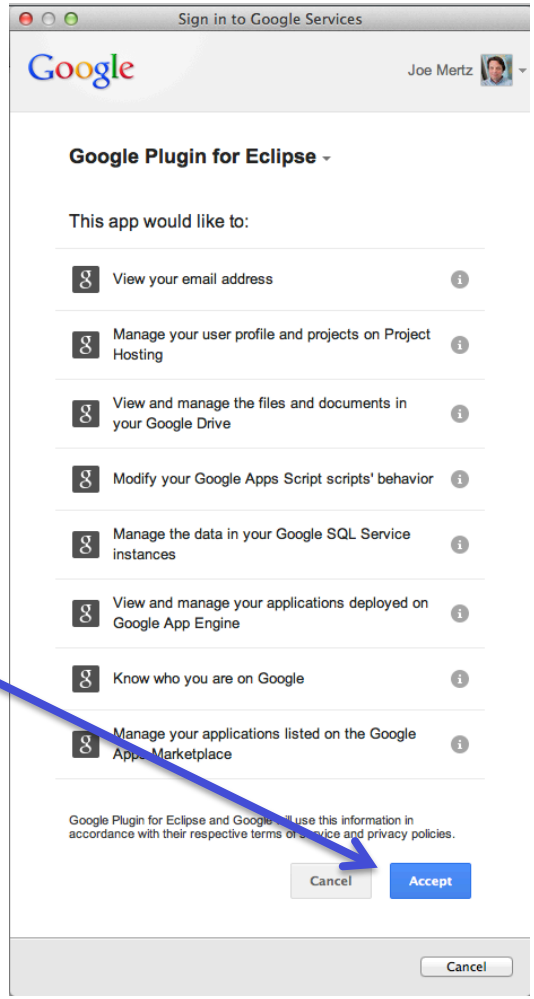
Create a Web Application Project

If not logged in, click here to log in



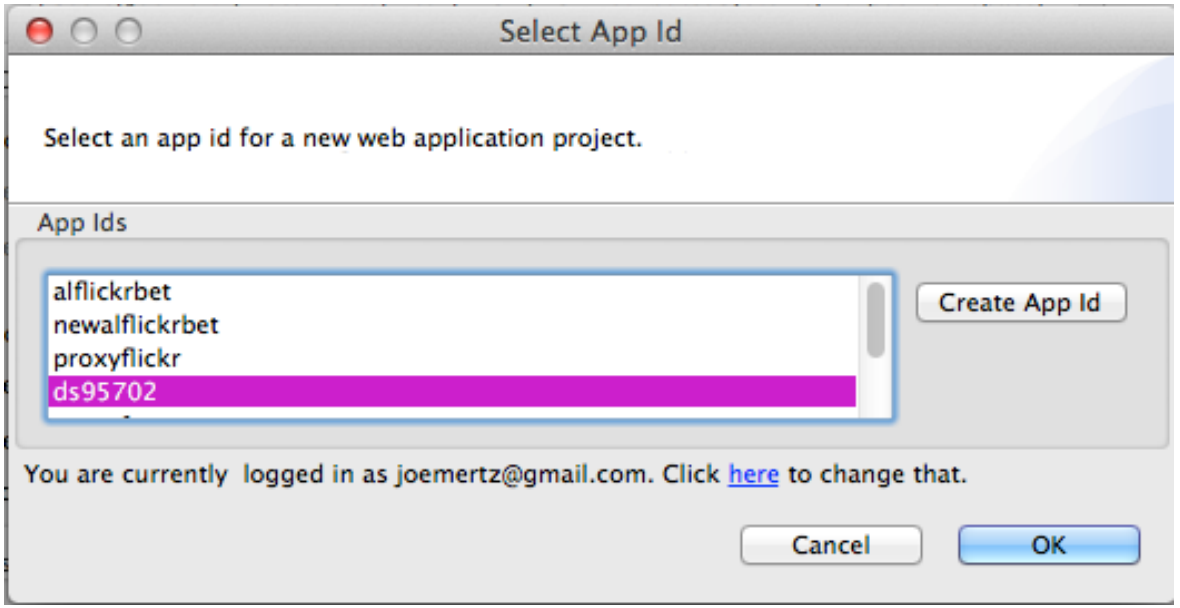
Create a Web Application Project

Authentication will look something like this.
"Accept"



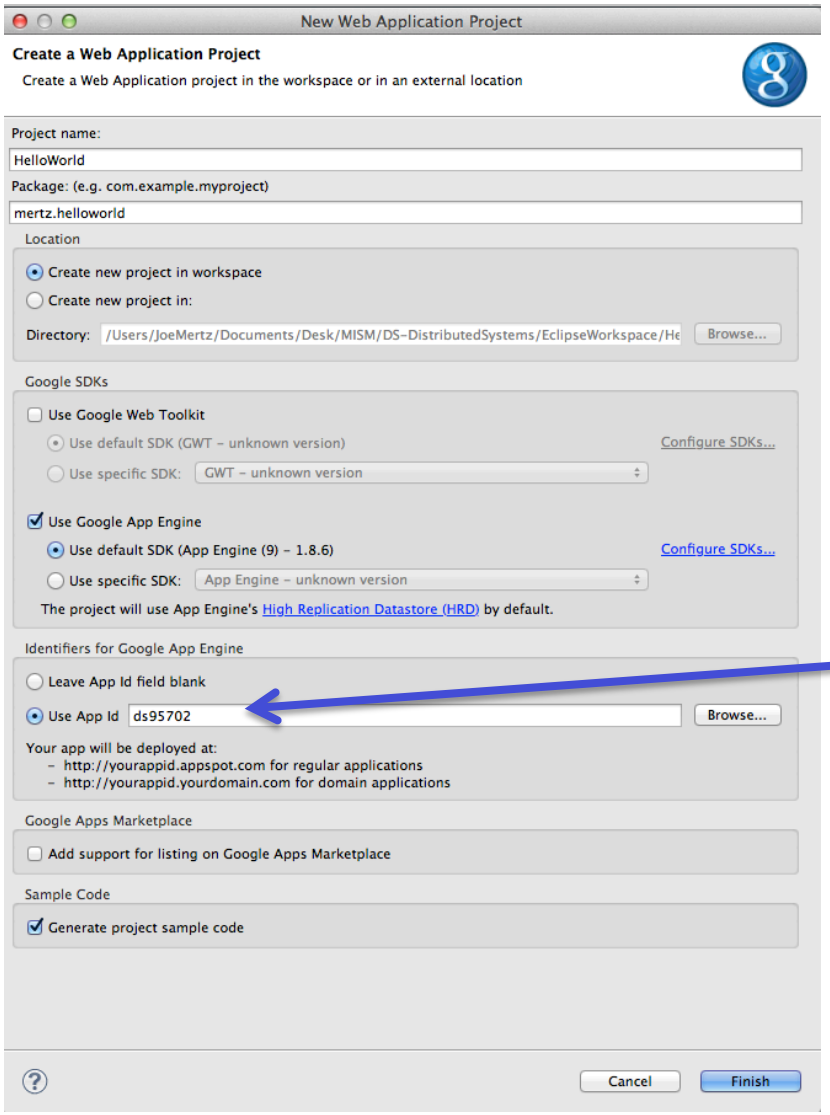
Create a Web Application Project

Once you are authenticated, you will see your list of applications you created on Google App Engine



Select the app id you want to link to this application and click OK

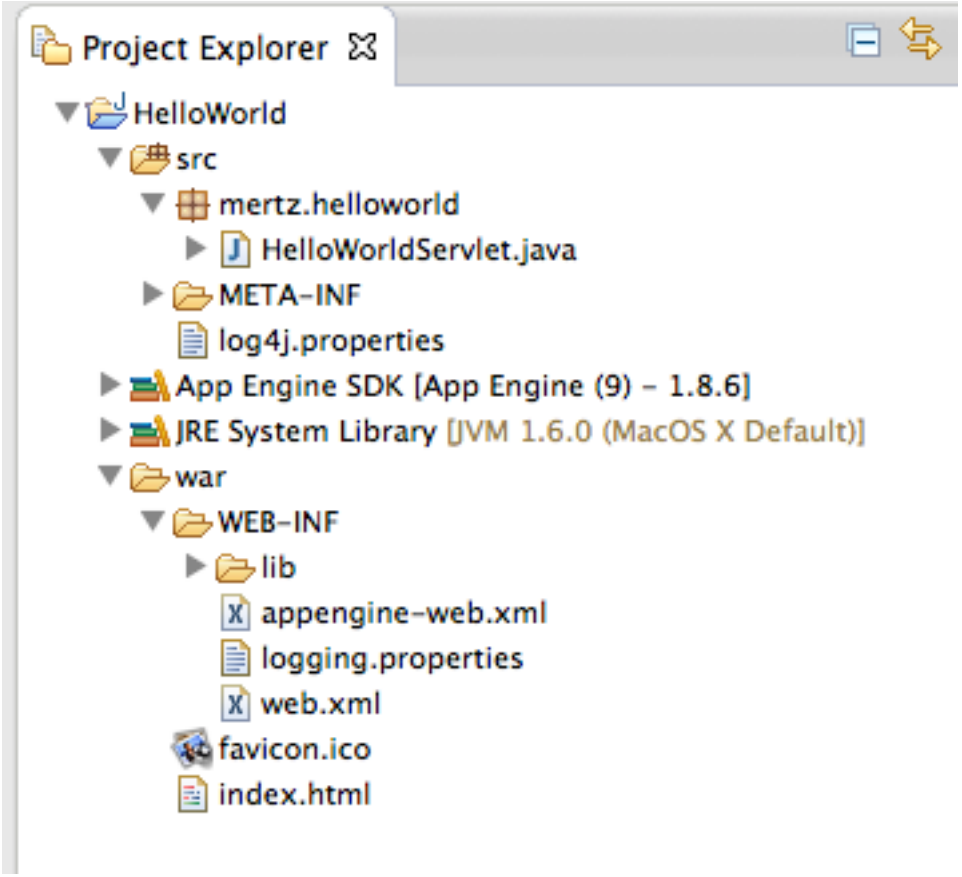
Create a Web Application Project



The app id should now be included.

Click Finish

Familiar Directory Structure



Servlet

HelloWorldServlet.java

```
1  package mertz.helloworld;
2
3  import java.io.IOException;
5
6  @SuppressWarnings("serial")
7  public class HelloWorldServlet 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
```

- Same standard servlet code
- Eclipse does not provide ProcessRequest
  - Different than the default in NetBeans
- Only an example of doGet
  - You could implement doPost, doPut, doDelete, etc.

Deployment Descriptor (web.xml)

- GAE installs Jetty
- Jetty is a FOSS web container
- Works with Eclipse and Google App Engine Plugin
- Does not use @WebServlet annotation
  - Must map URL to servlet in web.xml

web.xml

```
1  <?xml version="1.0" encoding="utf-8" standalone="no"?>
2  <web-app    xmlns="http://java.sun.com/xml/ns/javaee"
3              xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
4              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5              version="2.5"
6              xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
7                                  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
8      <servlet>
9          <servlet-name>HelloWorld</servlet-name>
10         <servlet-class>mertz.helloworld.HelloWorldServlet</servlet-class>
11     </servlet>
12     <servlet-mapping>
13         <servlet-name>HelloWorld</servlet-name>
14         <url-pattern>/helloworld</url-pattern>
15     </servlet-mapping>
16     <welcome-file-list>
17         <welcome-file>index.html</welcome-file>
18     </welcome-file-list>
19 <servlet>
20  <servlet-name>SystemServiceServlet</servlet-name>
21  <servlet-class>com.google.api.server.spi.SystemServiceServlet</servlet-class>
22  <init-param>
23   <param-name>services</param-name>
24   <param-value/>
25  </init-param>
26 </servlet>
```

Replicate servlet & servlet-mapping when adding servlets.

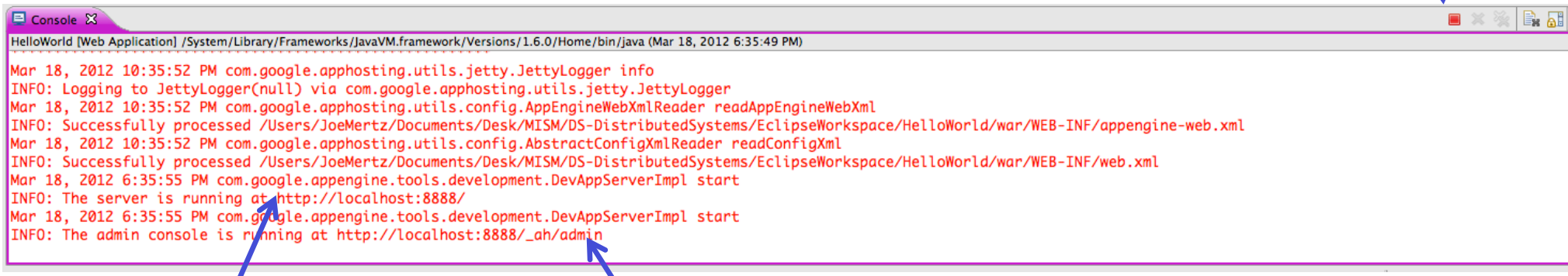
servlet-mapping maps a URL to a servlet-name

servlet maps the servlet name to the servlet-class.

Run on local server

Click  or Run As->Web Application

Stop

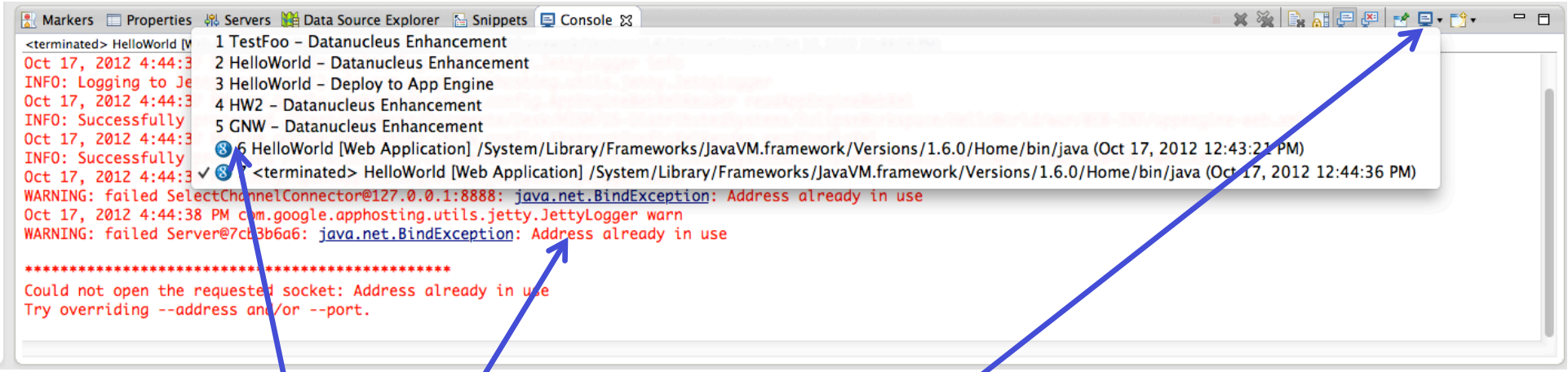


local URL

local admin URL

Address already in use

Unlike NetBeans, Eclipse "Run" deploys, it does not redeploy.
So it does not stop your previous deployment.



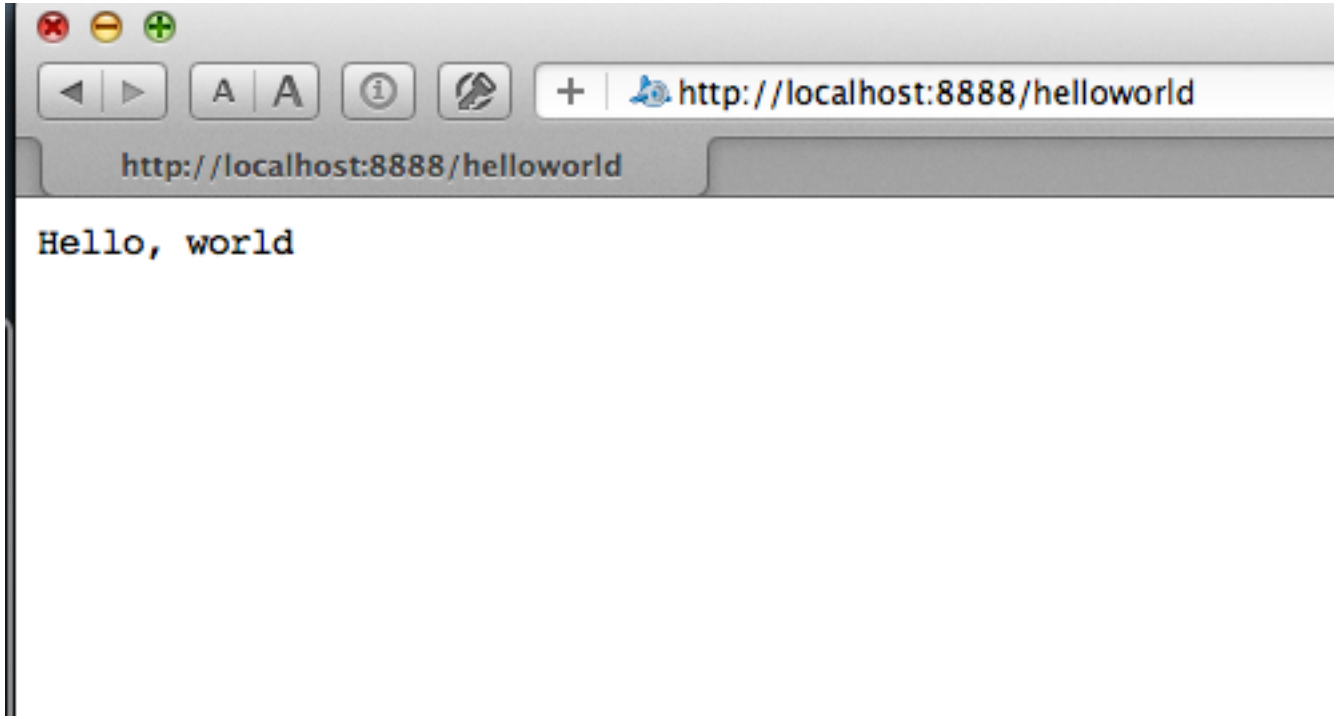
If you get "Address already in use"

Click on the right of the "Display Selected Console" icon

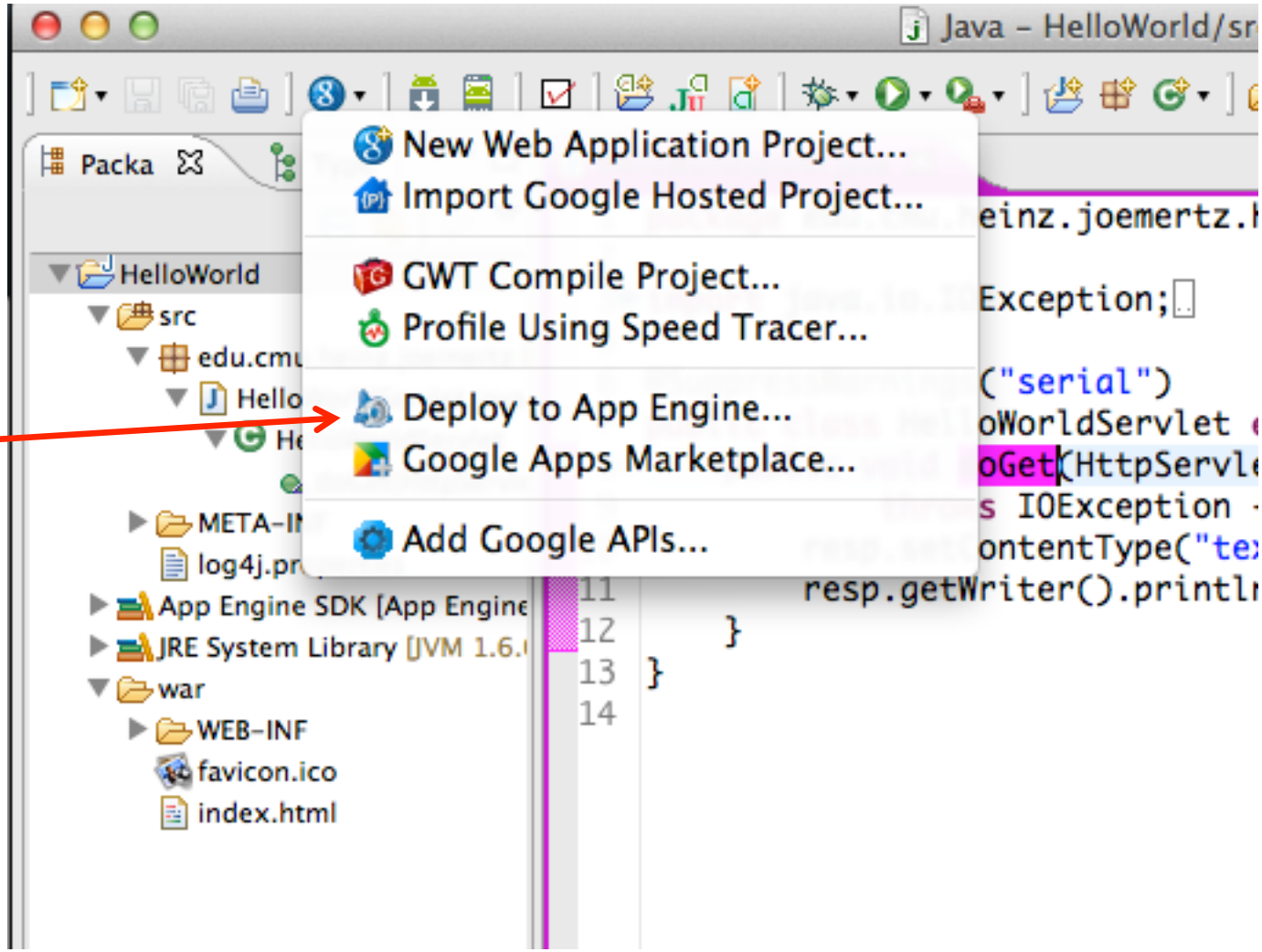
Select the console of the already-running web app.

Then hit 

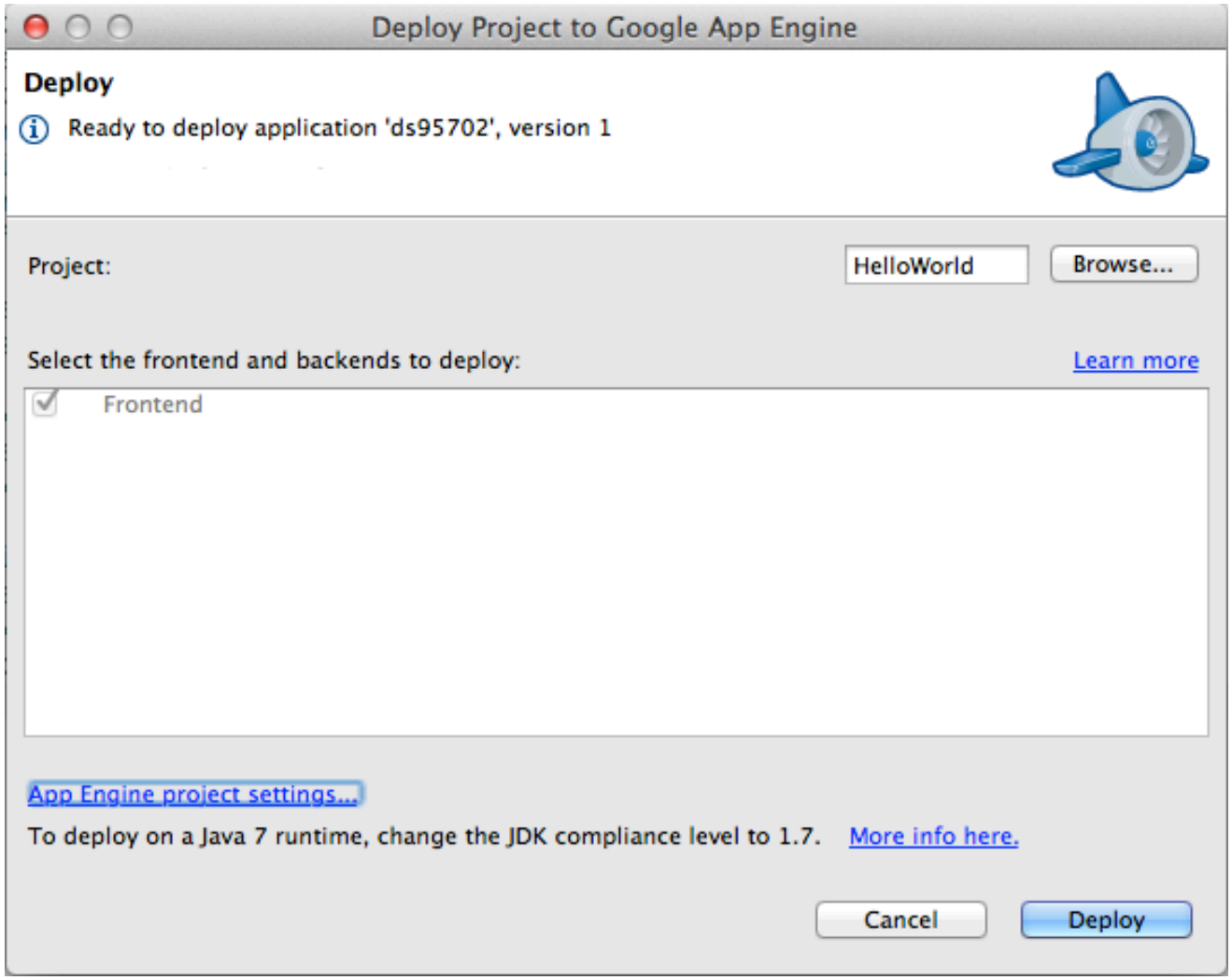
Servlet on local server



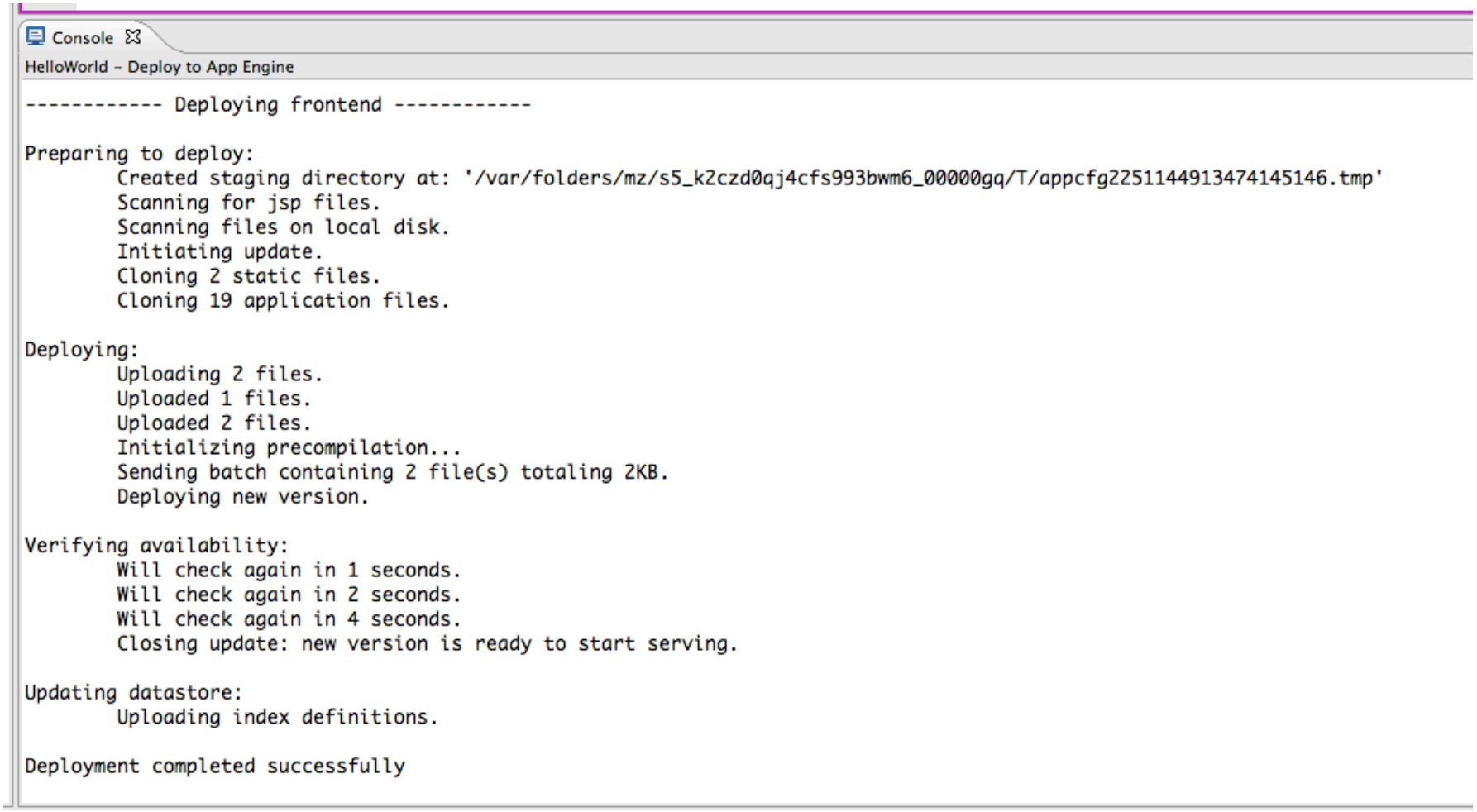
Deploy application to GAE



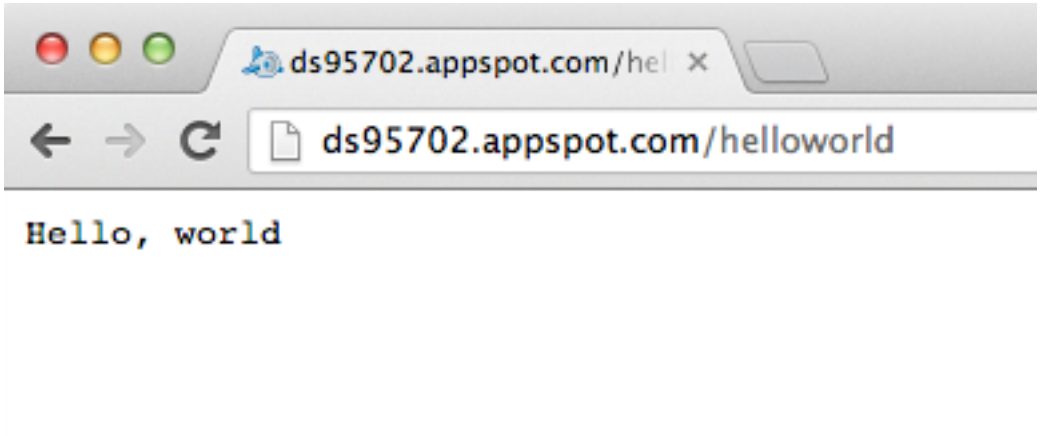
Deploy


Deploy Project to Google App Engine
Deploy
Ready to deploy application 'ds95702', version 1
Project:
HelloWorld
Browse...
Select the frontend and backends to deploy:
Learn more
Frontend
App Engine project settings...
To deploy on a Java 7 runtime, change the JDK compliance level to 1.7.
More info here.
Cancel
Deploy

Typical deployment messages



Servlet on Google App Engine



Exception on GAE

- If you get an exception when running your application on GAE:
  - Go to https://appengine.google.com/
  - Select your application (on left)
  - Click on "Logs" (on left)
  - Expand the top log error, to see the full stack trace.

GAE Lab

1. Get HelloWorld working locally and on GAE
   - Follow the screen shot slides
2. Deploy one of your servlets: Project 1, Task 2, 3, or 4
   - Not Project1Task1, sun.misc.BASE64Encoder is not available
   - Task 4 is the most challenging and most fun
   - Create files in Eclipse
   - Copy the code into those files
     - Remove @WebServlet's and add mappings to web.xml
   - Test on localhost
   - Deploy to GAE
3. To get credit, fill in the form at:
   - http://tinyurl.com/95702gae
   - For this lab only, you do **not** need to show a TA
   - As always, lab is due before class in one week.

TimeZone problems

- If you get the runtime exception:
  - "Unable to restore the previous TimeZone".
  - Add the following in run configuration:
    - "-Dappengine.user.timezone=UTC"
  - I.e.
    - Go to "run configuration",
    - Under the "Arguments" tab add to VM arguments:
      - "-Dappengine.user.timezone=UTC"

Unable to update app: Failed to compile jsp files.

- If you run into a problem with Eclipse recognizing .jsp files, you might need to let Eclipse know about the location of the JDK. Check out this solution:
  http://stackoverflow.com/questions/3112497/jsp-file-not-working-for-google-app-engine-guestbook-tutorial