# Deploy a Java Servlet in GAE

**STEP 1: Install Eclipse GAE Plugin**

In Eclipse bar menu, select **Help > Install New Software**.

Put the URL: <https://dl.google.com/eclipse/plugin/4.6>

***Note***: This URL is specific for Eclipse Kepler. For other distributions, please go to: <https://developers.google.com/eclipse/docs/install-eclipse-4.6>

For the installation, check the followings plugins

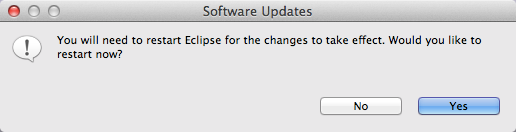
* Google Plugin for Eclipse
* SDK > Google App Engine Java SDK
* SDK > Google Web Toolkit SDK (optional).

Click on **Next**, accept the review licenses and **validate** to start the installation.

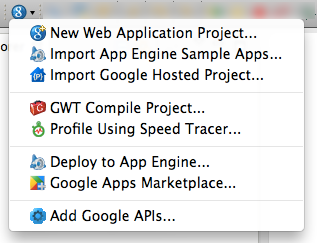
Click on **Ok** to validate the security warning when installing.



Restart Eclipse.



If the installation is successful, a new Google icon with a specific menu will be added to your Eclipse.



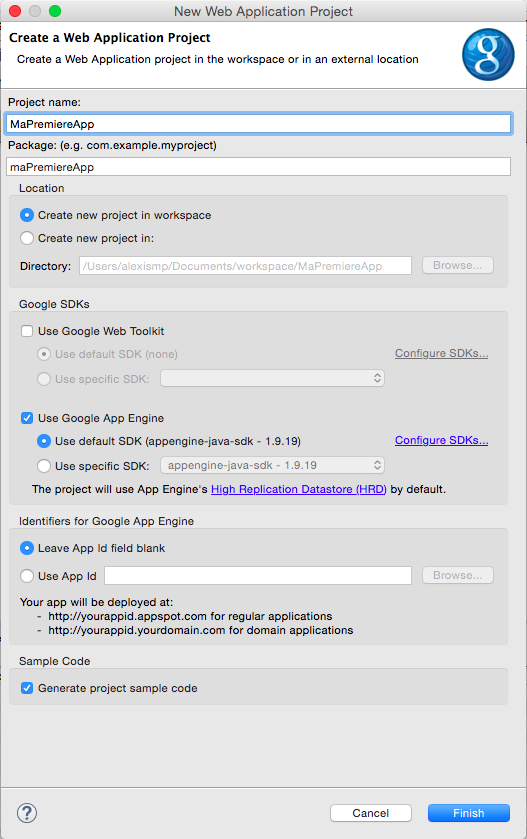
***Note***: In order to support and develop GAE projects with Maven, please refer to: <https://cloud.google.com/appengine/docs/java/tools/maven>

**STEP 2: CREATE YOUR APP**

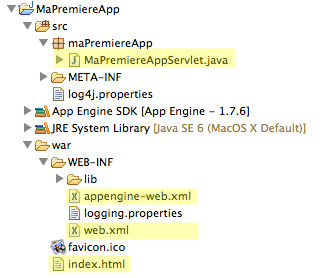
To create a new APP Engine application, Click on the “**g**” Google button on Eclipse and select **New Web Application Project**.

The application creation wizard is displayed.

* Insert the project name (e.g. MyFirstApp).
* Insert the package name (e.g. myFirstApp).
* Check “Use Google App Engine” and “Generate Project Sample Code” options.
* Uncheck “Use Google Web Toolkit” option.

Click on **Finish** to start the application creation. The App Engine will generate the according java project.

The created project architecture consists of a set of files and repositories. The content structure is very similar to a classical JEE project structure. The main differences are highlighted in the following snapshot.



* ***The Servlet file:*** The Java Servlet source code. The servlet code is similar to a classical JEE servlets. The unique difference is that we can access/import App engines libraries (if needed!).

package myFirstApp;

import java.io.IOException;

import javax.servlet.http.\*;

@SuppressWarnings("serial")

public class MyFirstAppServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws IOException {

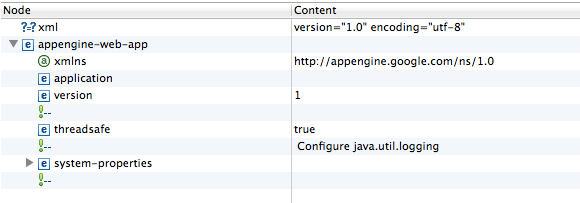
resp.setContentType("text/plain");

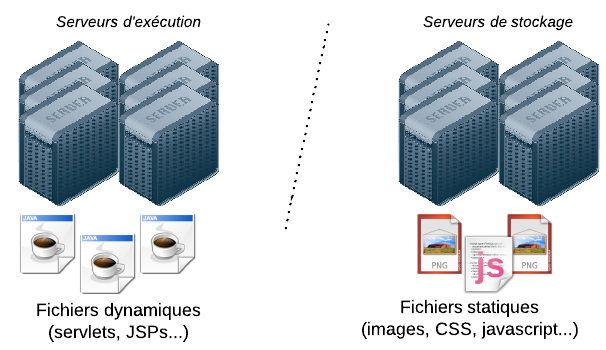
resp.getWriter().println("Hello, world");

}

}

* ***The appengine-web.xml file:*** This is your App Engine configuration file. This is the only concrete difference between a classical JEE project and an App Engine project.

******

This file describes general information about your App Engine application (e.g. name, version, etc.). It allows you to enable/disable some specific functionalities (e.g. sessions, google services, static files, etc.).

Example: static files declaration

<static-files>

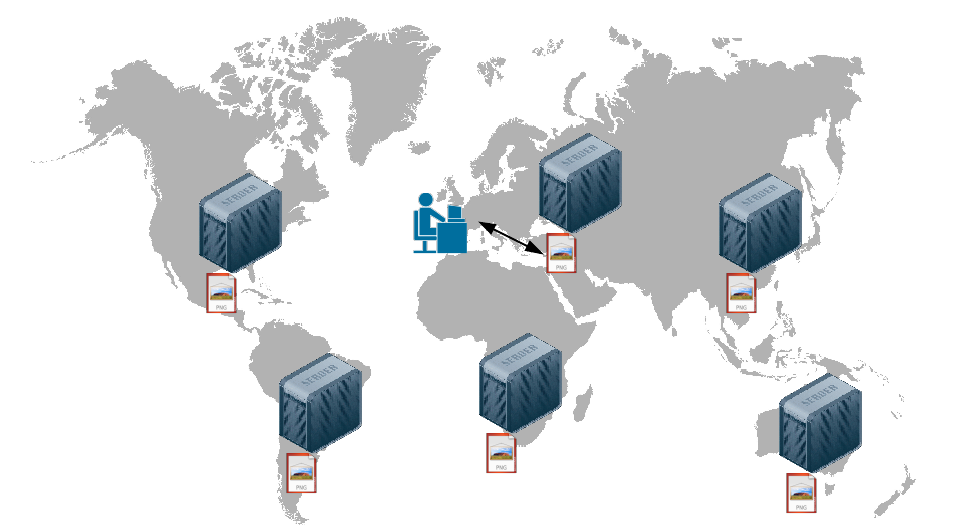
<include path="/\*\*.png" />

<include path="/javascript/\*.js" />

<exclude path="/generator/\*\*.png" />

</static-files>

1. **Content placement**



1. **Content duplication**

* ***The web.xml file:*** This is the classical JEE configuration file. It lists and contains mapping information about your application’s servlets.

<?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>MyFirstApp</servlet-name>

<servlet-class>myFirstApp.MyFirstAppServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyFirstApp</servlet-name>

<url-pattern>/myfirstapp</url-pattern>

</servlet-mapping>

<welcome-file-list>

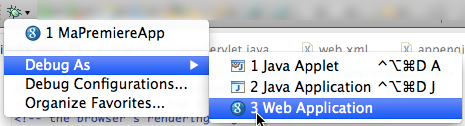
<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

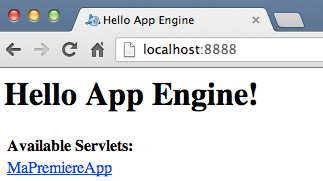
* ***The index.html file:*** The HTML welcome page.

**STEP 3: DEBUG YOUR APP**

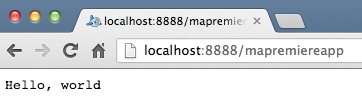
In order to debug your app and test locally, click on **"Debug as" > "Web Application".**

You should have "INFO: Dev App Server is now running" displayed then on your Eclipse console. This means that your local google mini-server is running.

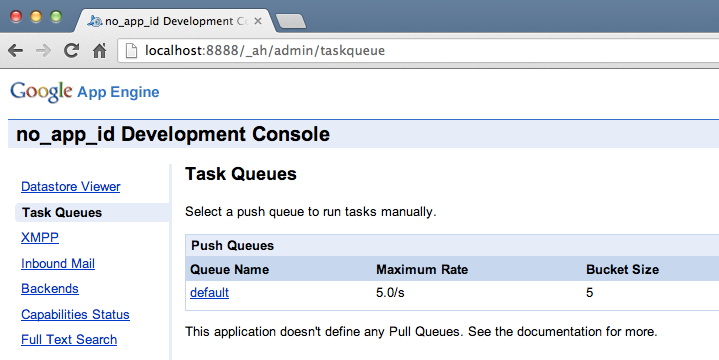
Use a browser and go to <http://localhost:8888/> to test. If everything works well, you will be redirected to your index.html page.



Click on the link in order to execute your servlet.

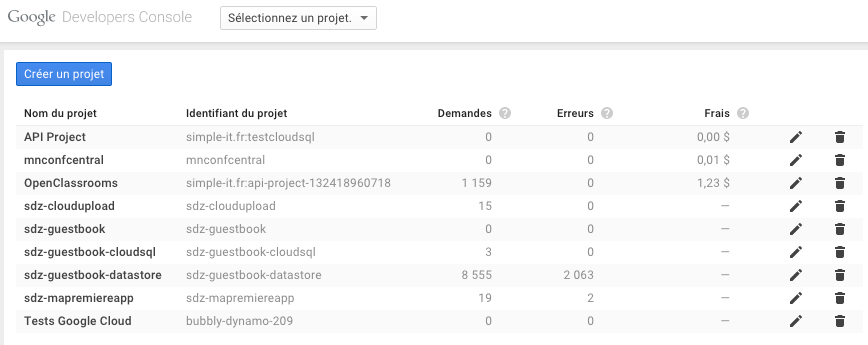


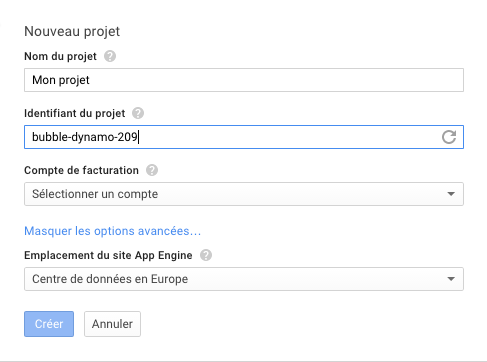
Note that it is possible to administrate and monitor your application using the local administration tool provided by the plugin. The administration console is accessible at: <http://localhost:8888/_ah/admin>



**STEP 4: DEPLOY YOU APP**

In order to deploy our application on google servers, we need to declare it to google first and get a unique ID for it. To that end, go to <http://console.developers.google.com>



Then, create a new project.

A project ID will created automatically. You can also select in where you want deploy you project.

Copy and paste this ID between the <application> elements in your appengine-web.xml file.

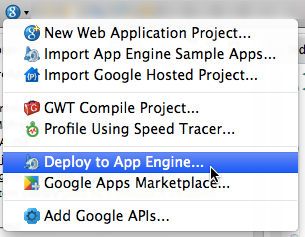
<?xml version="1.0" encoding="utf-8"?>

<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">

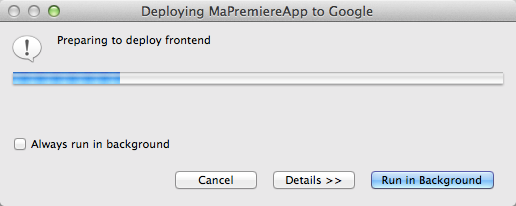
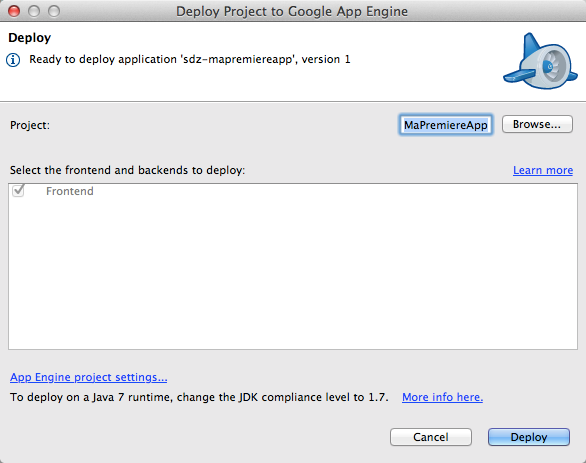
<application>sdz-myfirstapp</application>

<version>1</version>

Your application is ready!! To deploy it, click on “**Deploy to App Engine**” in the google menu (Authentication to Google is required).

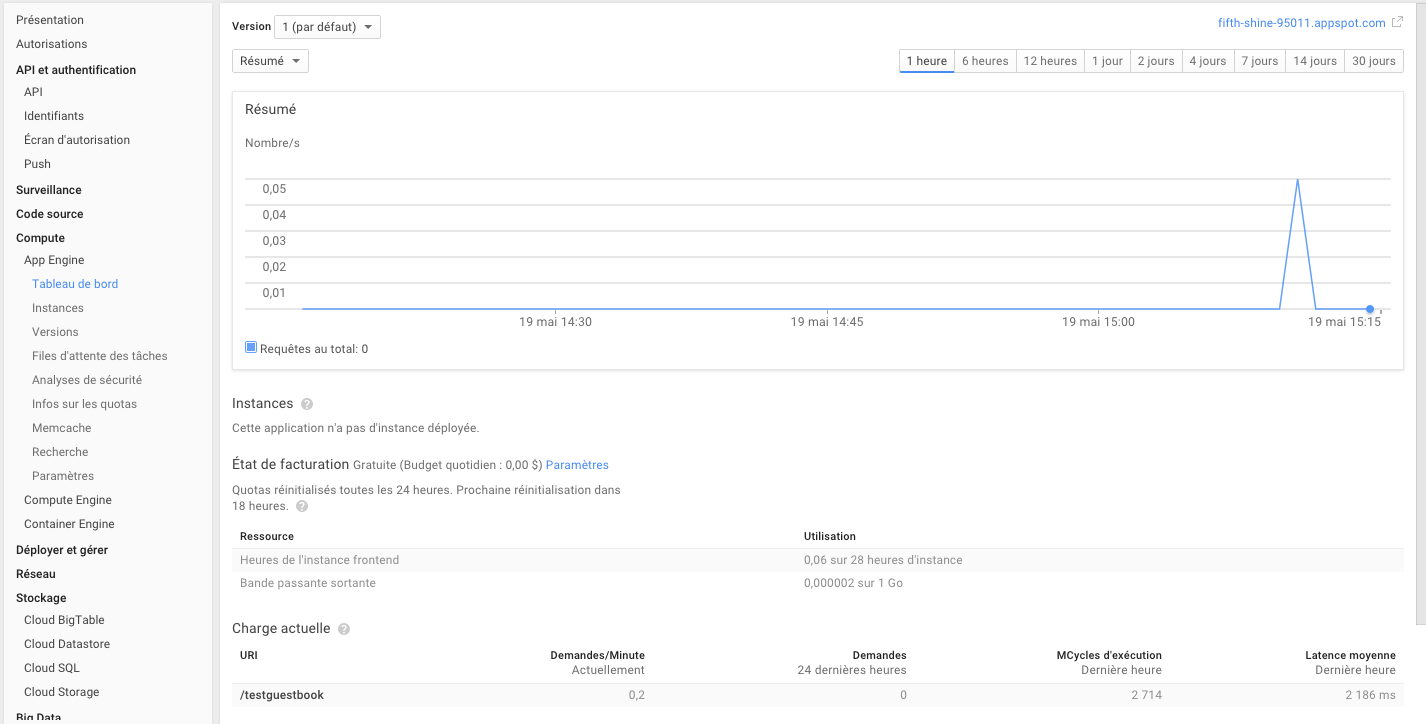


Click on “**Deploy**”.



**STEP 5: MANAGE YOUR APP**

Once your App is deployed, you can manage it though the administration dashboard (Compute / App Engine / Dashbord).

****

# Deploy a Sample Application on Cloud Foundry

The instructions are available at: <https://docs.cloudfoundry.org/devguide/deploy-apps/deploy-app.html#intro>