MIND STORM SOFTWARE PVT LTD

Google App Engine Training

App Engine – Hello World

This exercise covers how to write your first App Engine application in Java. It covers the following:

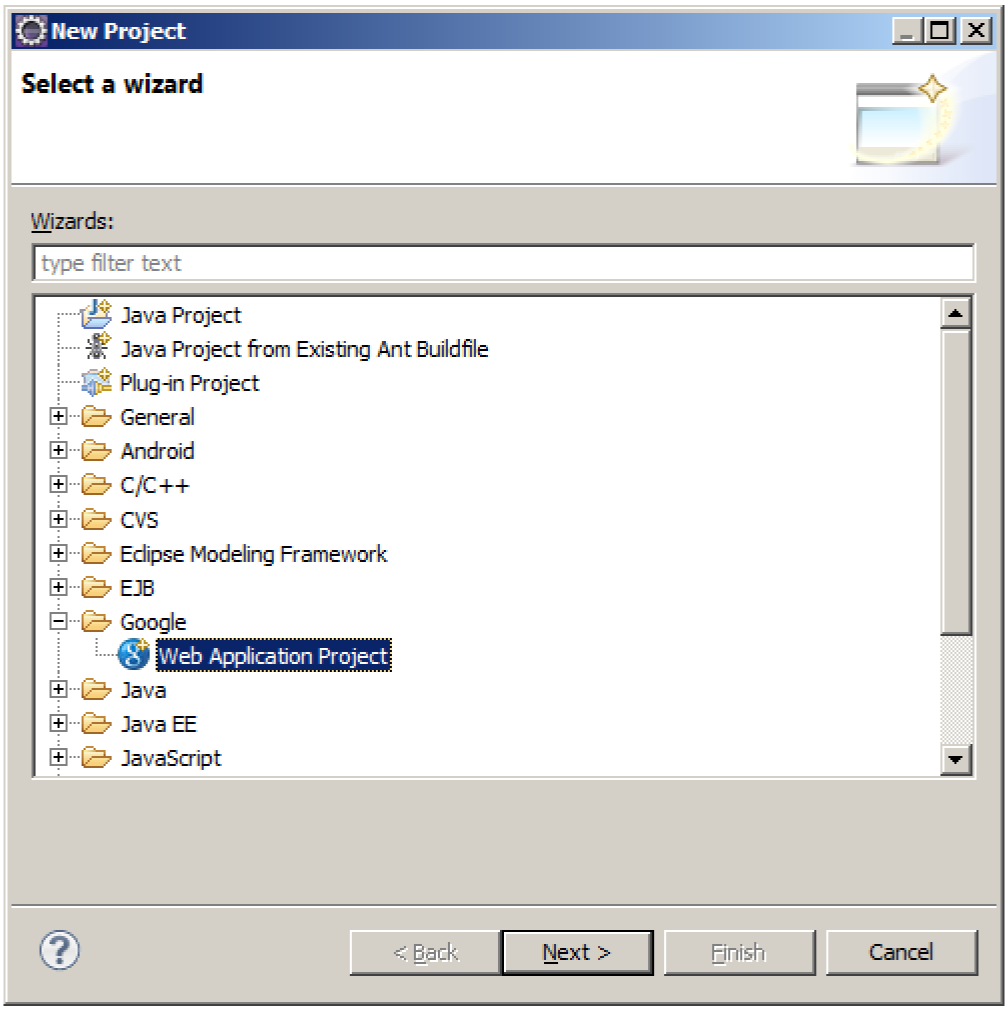
- Use Eclipse and Google Eclipse Plug-in to generate the App Engine application
- Test the application locally
- Setup your App Engine account and create Application Id
- Upload your application to App Engine and see it live.

Generate Project

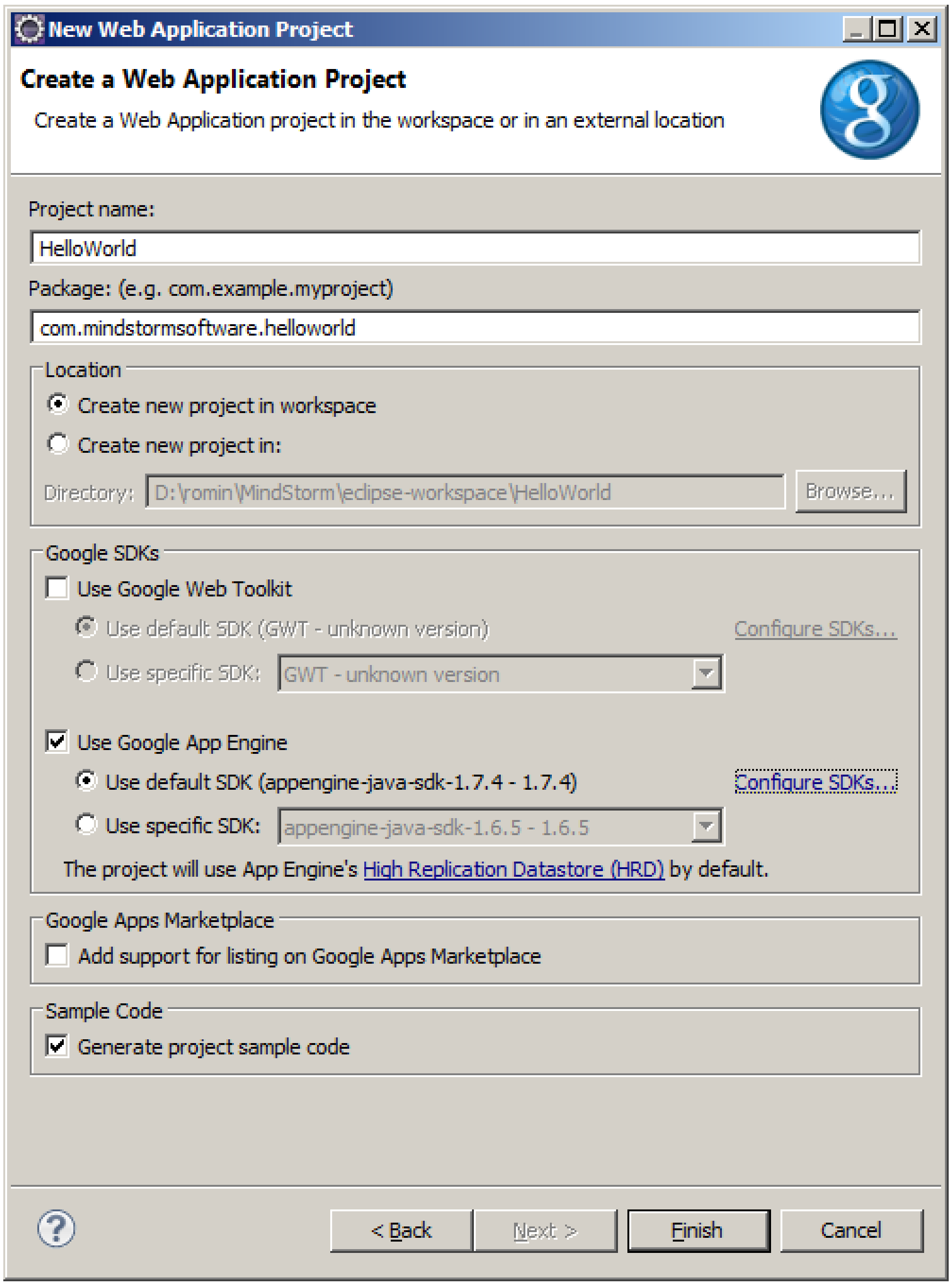
This step assumes that you have successfully completed the setup of your development environment. It assumes that you have:

- Java SDK installed (Version 6 is preferred)
- Eclipse Juno (4.2) with Google Eclipse Plug-in is installed

1. To generate the project, click **File → New → Project ...** . This will bring up the New Project dialog as shown below. Select **Google → Web Application Project** and click on **Next.**



2. This will bring up the project details project which you will need to fill up as shown below:



New Web Application Project

Create a Web Application Project

Create a Web Application project in the workspace or in an external location

Project name: HelloWorld

Package: (e.g. com.example.myproject) com.mindstormsoftware.helloworld

Location
- [✓ Create new project in workspace]
- Create new project in:
- Directory: D:\romin\MindStorm\eclipse-workspace\HelloWorld

Google SDKs
- Use Google Web Toolkit
  - Use default SDK (GWT - unknown version)
  - Use specific SDK: GWT - unknown version
- [✓ Use Google App Engine]
  - [✓ Use default SDK (appengine-java-sdk-1.7.4 - 1.7.4)]
  - Use specific SDK: appengine-java-sdk-1.6.5 - 1.6.5

The project will use App Engine's High Replication Datastore (HRD) by default.

Google Apps Marketplace
- Add support for listing on Google Apps Marketplace

Sample Code
- [✓ Generate project sample code]

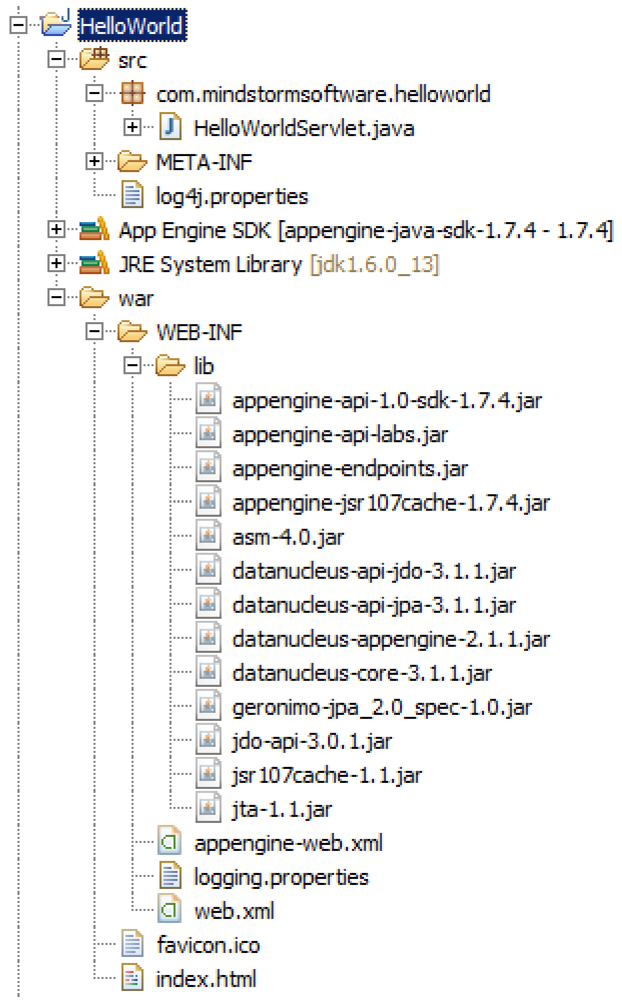
Enter the following values:

- **Project Name** : Enter this as HelloWorld
- **Package**: Enter a unique package name. This is the same as a Java package name. Select any one of your choice. E.g. **com.<some-company-name>.app-name** is a good convention to go with.
- Deselect the Google Web Toolkit option
- Ensure that the latest/default version of App Engine is selected. If you have multiple versions of SDK installed, you can select any one of them. For our scenario, most likely you will have just one version of the SDK installed.

Click on **Finish** button. This will generate the Eclipse project. Next, we will look at the main Project structure and files.

Project File Structure

The screenshot below shows the various folders that were generated.



```
HelloWorld
  src
    com.mindstormsoftware.helloworld
      HelloWorldServlet.java
    META-INF
    log4j.properties
  App Engine SDK [appengine-java-sdk-1.7.4 - 1.7.4]
  JRE System Library [jdk1.6.0_13]
  war
    WEB-INF
      lib
        appengine-api-1.0-sdk-1.7.4.jar
        appengine-api-labs.jar
        appengine-endpoints.jar
        appengine-jsr107cache-1.7.4.jar
        asm-4.0.jar
        datanucleus-api-jdo-3.1.1.jar
        datanucleus-api-jpa-3.1.1.jar
        datanucleus-appengine-2.1.1.jar
        datanucleus-core-3.1.1.jar
        geronimo-jpa_2.0_spec-1.0.jar
        jdo-api-3.0.1.jar
        jsr107cache-1.1.jar
        jta-1.1.jar
      appengine-web.xml
      logging.properties
      web.xml
    favicon.ico
    index.html
```

The folder structure is similar to any Java Web Application. App Engine follows the Java Servlet 2.5 specification and if you were familiar with the Web Application Archive (WAR) directory for Java Web Applications, this should look familiar.

Few key directories:

1. **src**: All source code will be present in this folder. This includes your Java classes, Servlets, etc.
2. **war**: This folder is the standard Web Application Archive folder. All JSP/HTML/JS/CSS files go in this folder.
3. **war/WEB-INF/lib**: All AppEngine standard JARs are present here. If you use any external libraries, the will need to be placed here and you use set the Java Build Path in your Eclipse project to reference the external JAR files.

Other important files are listed below:

war/WEB-INF/appengine-web.xml

This file contains information about your application and various other services and runtime requirements that you application expects when it is running in the App Engine infrastructure.

The key elements to note are given below:

```
  <application></application>
  <version>1</version>
```

The **<application>** element contains the Application ID of your application. When you create an application on App Engine, you will need to specify the application ID over here, so that when you deploy to App Engine, it will use this application ID, to which the version should be deployed. We will get to this later, so leave it as is for now.

By default, the **<version>** element is set to a value of 1. You can change it if you want. Keep in mind that you can deploy multiple versions of the application under the same application ID. However, only one version will be the default one. You can set default versions via the Admin Dashboard, which we will cover later on.

As we utilize other services in the App Engine world, we will keep revisiting this file where we will add support for the services required.

An example of an entry is shown below. This service is required when you need your App Engine application to receive incoming mail sent to your application.

```
<inbound-services>
        <service>mail</service>
</inbound-services>
```

war/WEB-INF/logging.properties

This file contains the logger settings. Your application code will log messages as needed in your applications. Typically logging is done with a message (text) and a Log Level. Different Log Levels exist i.e. ERROR, WARNING, INFO, DEBUG, etc.

This file contains the minimum level of Log Level. If the Application logs a message with a Log Level higher than the one set in this file, then the message is logged otherwise it is ignored.

For e.g. the default level is set to warning in the file via the line shown below:

```
.level = WARNING
```

The Log Level order is given as follows:

```
FATAL > ERROR > WARNING > INFO > DEBUG
```

So if the logging.properties has the level set to ERROR and if you raise a log message with a Log Level of WARNING, then the message will not get logged.

It is suggested that you change this level down to .INFO and save the file.

war/WEB-INF/web.xml

This is the main configuration file for the application. This is the same as a standard **web.xml** file for Java Web Applications.

It will contain entries for :

1) Servlets and their mappings to URLs
2) Welcome File List
3) Error Pages

And much more.

By default, the web.xml contains the only Servlet that is pregenerated for you and which is available on the **/helloworld** endpoint in your application. It also specifies the **index.html** file as the welcome file in your application.

```
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class>com.mindstormsoftware.helloworld.HelloWorldServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/helloworld</url-pattern>
</servlet-mapping>

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

HelloWorldServlet.java

The default Servlet that is generated for you simply prints out the **Hello, world** message as shown below.

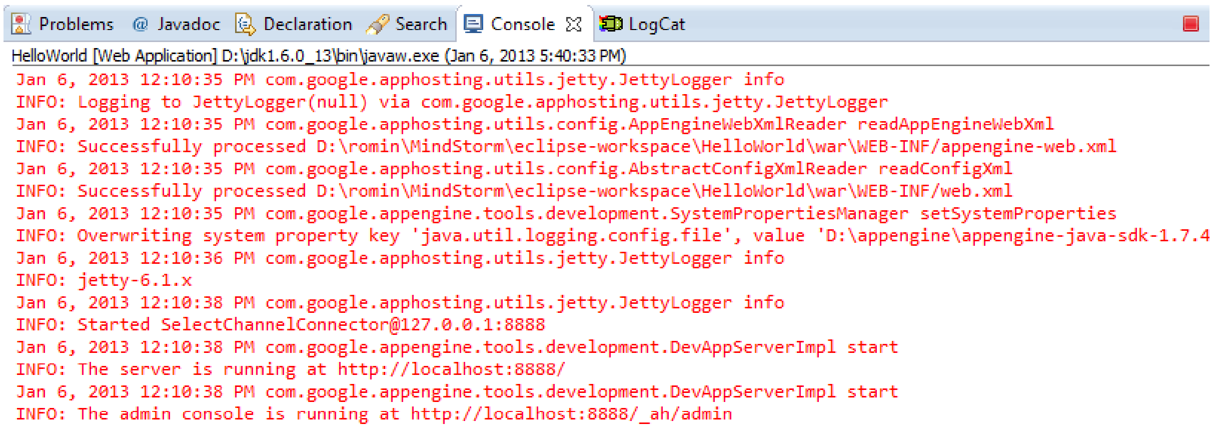
```
package com.mindstormsoftware.helloworld;

import java.io.IOException;
import javax.servlet.http.*;

@SuppressWarnings("serial")
public class HelloWorldServlet extends HttpServlet {
	public void doGet(HttpServletRequest req, HttpServletResponse resp)
			throws IOException {
		resp.setContentType("text/plain");
		resp.getWriter().println("Hello, world");
	}
}
```

Running the Application

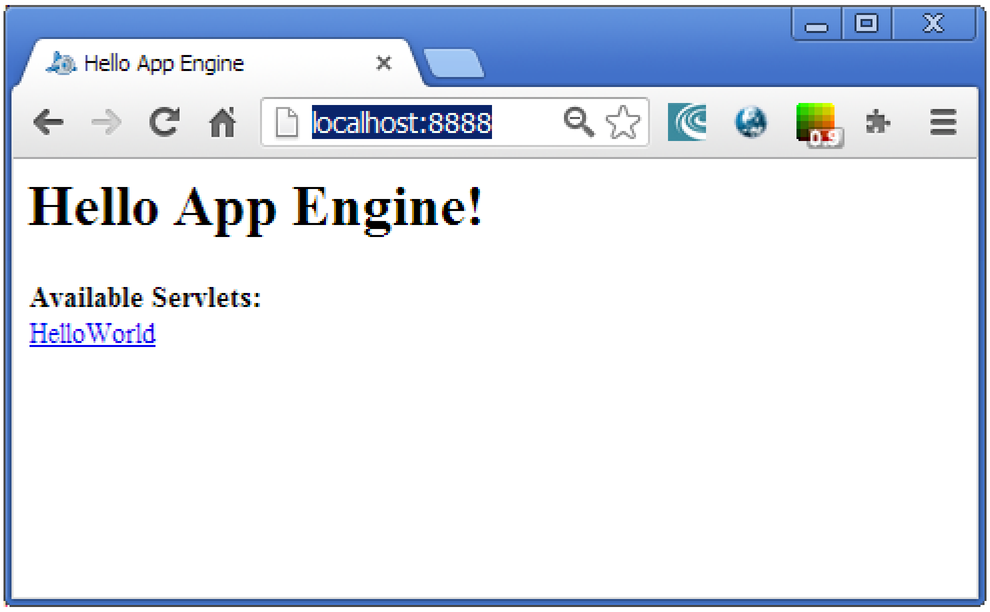
1. Select the project **HelloWorld** in the Eclipse Package Explorer
2. Right-click and select **Run As → Web Application**
3. This will start the Jetty Web Server locally and deploy your web application to it. On successful starting up, you should see the following message in your Eclipse Console View.
4.



```
HelloWorld [Web Application] D:\jdk1.6.0_13\bin\javaw.exe (Jan 6, 2013 5:40:33 PM)
Jan 6, 2013 12:10:35 PM com.google.apphosting.utils.jetty.JettyLogger info
INFO: Logging to JettyLogger(null) via com.google.apphosting.utils.jetty.JettyLogger
Jan 6, 2013 12:10:35 PM com.google.apphosting.utils.config.AppEngineWebXmlReader readAppEngineWebXml
INFO: Successfully processed D:\romin\MindStorm\eclipse-workspace\HelloWorld\war\WEB-INF/appengine-web.xml
Jan 6, 2013 12:10:35 PM com.google.apphosting.utils.config.AbstractConfigXmlReader readConfigXml
INFO: Successfully processed D:\romin\MindStorm\eclipse-workspace\HelloWorld\war\WEB-INF/web.xml
Jan 6, 2013 12:10:35 PM com.google.appengine.tools.development.SystemPropertiesManager setSystemProperties
INFO: Overwriting system property key 'java.util.logging.config.file', value 'D:\appengine\appengine-java-sdk-1.7.4
Jan 6, 2013 12:10:36 PM com.google.apphosting.utils.jetty.JettyLogger info
INFO: jetty-6.1.x
Jan 6, 2013 12:10:38 PM com.google.apphosting.utils.jetty.JettyLogger info
INFO: Started SelectChannelConnector@127.0.0.1:8888
Jan 6, 2013 12:10:38 PM com.google.appengine.tools.development.DevAppServerImpl start
INFO: The server is running at http://localhost:8888/
Jan 6, 2013 12:10:38 PM com.google.appengine.tools.development.DevAppServerImpl start
INFO: The admin console is running at http://localhost:8888/_ah/admin
```

*If you do not see the Console View in your Eclipse IDE, select **Window → Show View → Console** from the main menu*

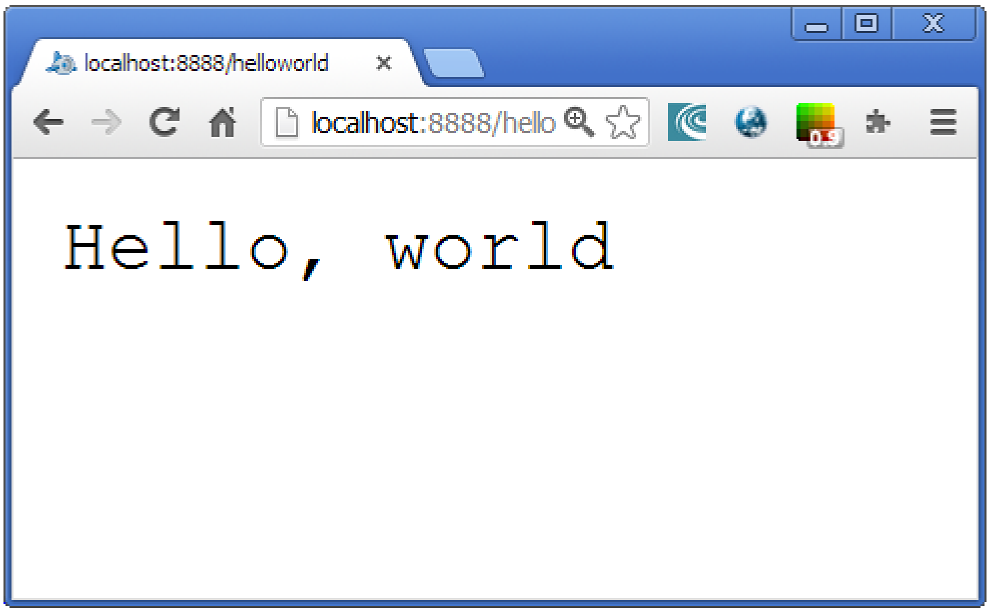
5. The Web Server runs on default port 8888.
6. To visit your application, launch your browser and visit http://localhost:8888. This should bring up the default **index.html** page as shown below:



Hello App Engine!

Available Servlets:
HelloWorld

7. When you click on Hello World link, it will invoke the **Java Servlet** mapped on **/helloworld** url and which will stream out the Hello World response as shown below:



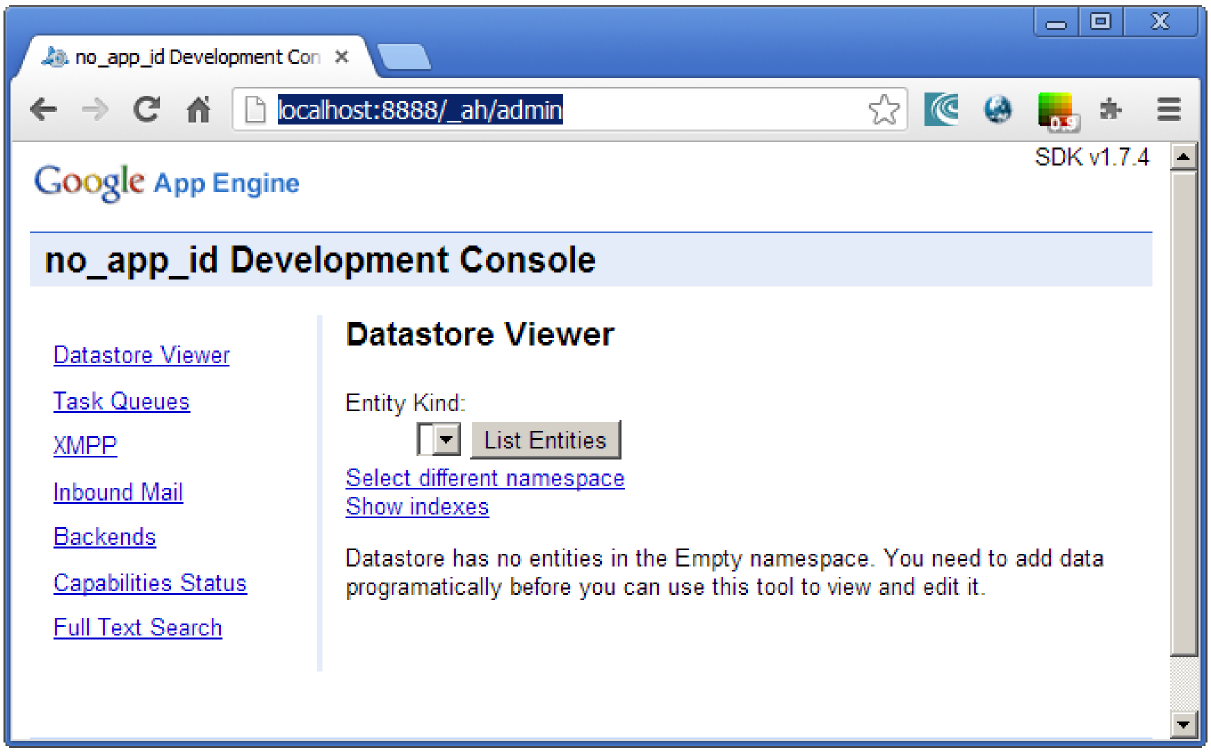
```
Hello, world
```

Admin Console for Local Development Server

The local web server that is launched also contains a simple web Administrative Console application that can help you perform some operations.

The local Admin console is available at http://localhost:8888/_ah/admin

A sample screen is shown below:



no_app_id Development Console

Datastore Viewer

Datastore has no entities in the Empty namespace. You need to add data programatically before you can use this tool to view and edit it.

The local Admin console that we shall come later to in the training allows you to do a few things:

1) Check the contents of your Datastore

2) Do XMPP Testing

We will make use of the Admin console as we move along in the course.

Deploying the Application

Deploying the Application consists of the following two steps:

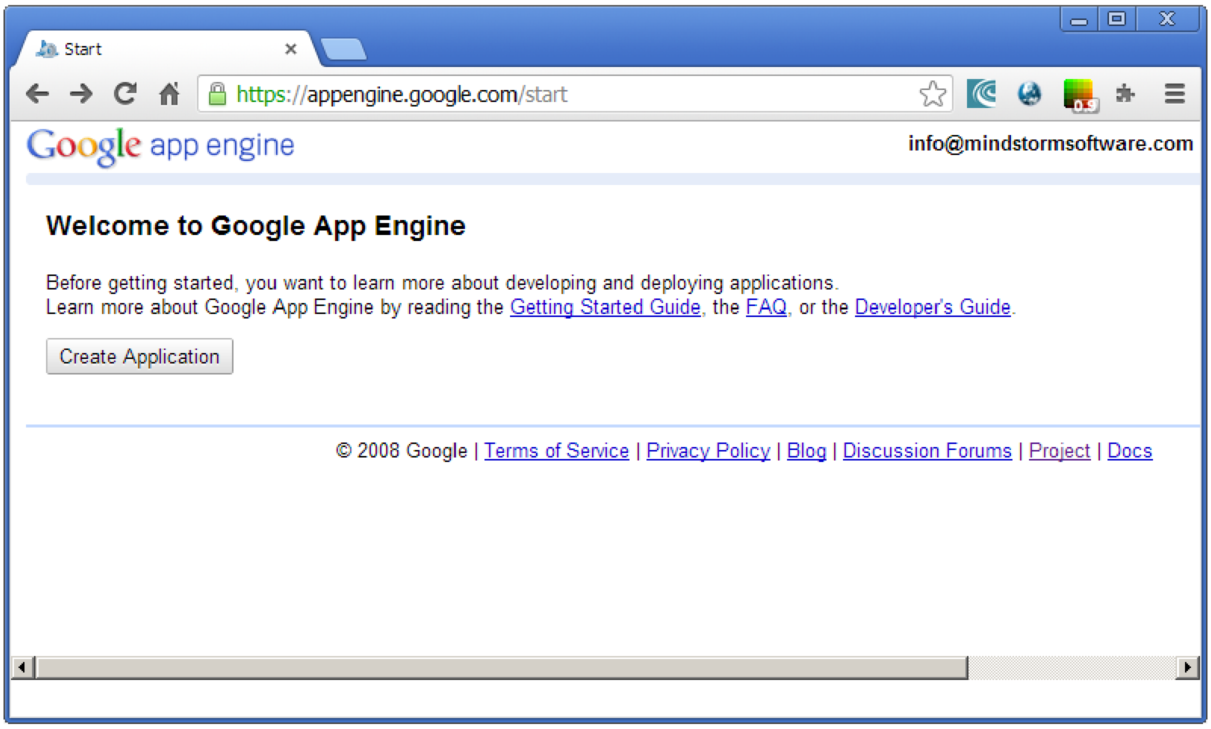
1) Ensure that you have created an Application in your Google App Engine Account and you know the Application ID (appid)
2) Use the Eclipse environment to login to your Google App Engine account and one-click deploy the application

Let us do the above 2 steps now.

Create an Application in your Google App Engine Account

Before you begin deploying your application, ensure that you have signed up for App Engine. Go to http://appengine.google.com and sign up with your Google Account. The verification involves that you provide your mobile number and you will be sent a SMS code for verification before your account is verified.

For e.g. once you login at http://appengine.google.com with your Google Account, you should see a screen similar to the following:

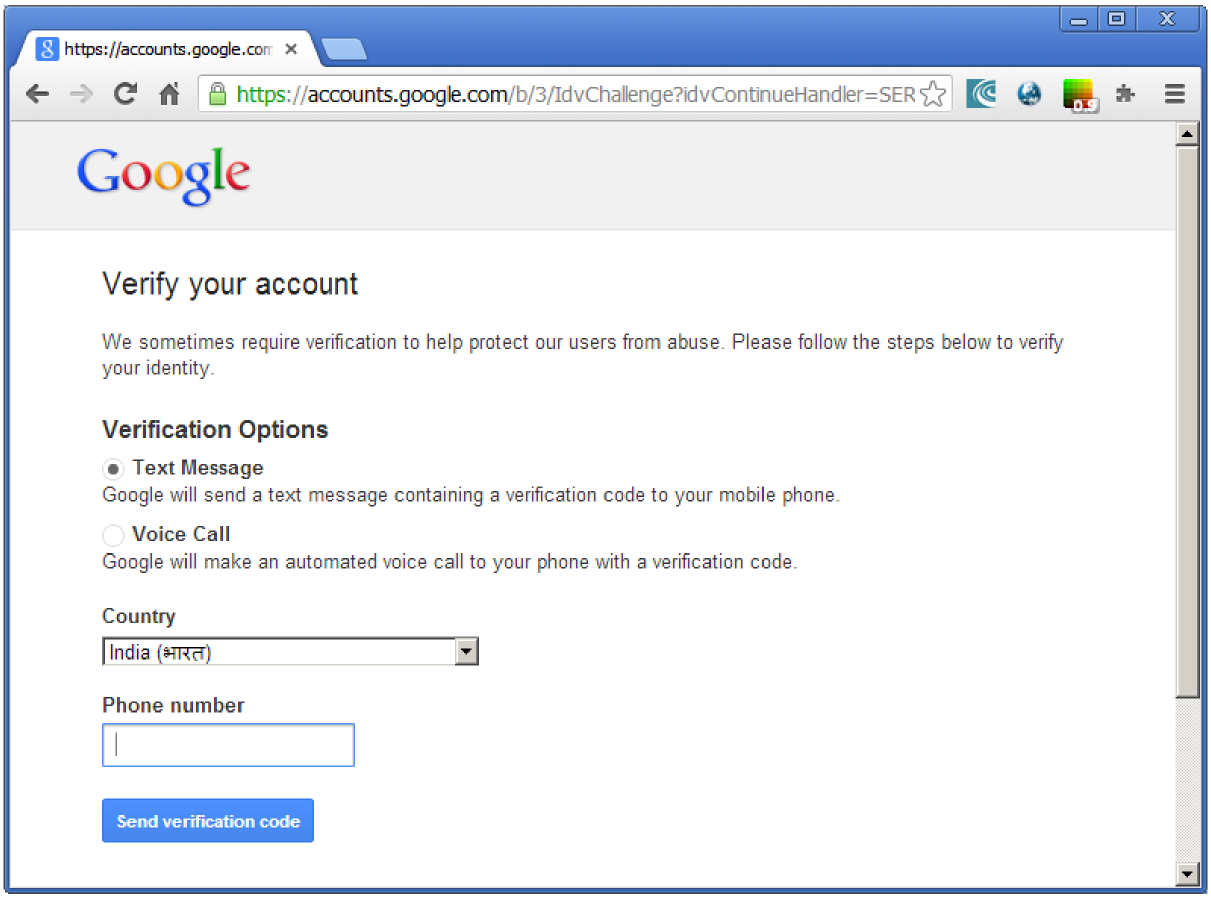

Start
https://appengine.google.com/start
Google app engine
info@mindstormsoftware.com

Welcome to Google App Engine

Before getting started, you want to learn more about developing and deploying applications.
Learn more about Google App Engine by reading the Getting Started Guide, the FAQ, or the Developer's Guide.

Create Application
© 2008 Google | Terms of Service | Privacy Policy | Blog | Discussion Forums | Project | Docs

Click on **Create Application**. This will bring up the SMS Verification page as shown below:


https://accounts.google.com/b/3/IdvChallenge?idvContinueHandler=SER

Verify your account

We sometimes require verification to help protect our users from abuse. Please follow the steps below to verify your identity.

Verification Options

● Text Message
Google will send a text message containing a verification code to your mobile phone.

○ Voice Call
Google will make an automated voice call to your phone with a verification code.

Country: India (भारत)

Phone number:

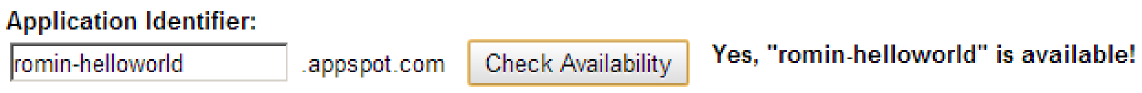
Enter your Mobile Phone number and it will send you the verification code that you will then be prompted to enter to complete the verification process. **Do note that if your mobile number has been validated once, you cannot use it to associate it with another Google Account.**

Every verified Account will have a quota of 10 Applications and each application is provided the free quota as per the quota limits provided by the App Engine environment.

To deploy our application, we will need to create one application. To do that, login to http://appengine.google.com and go to My Applications. This will bring up the list of your current applications in the account and if you have not setup any applications, this list will be empty.

To create an application, follow the steps given below:

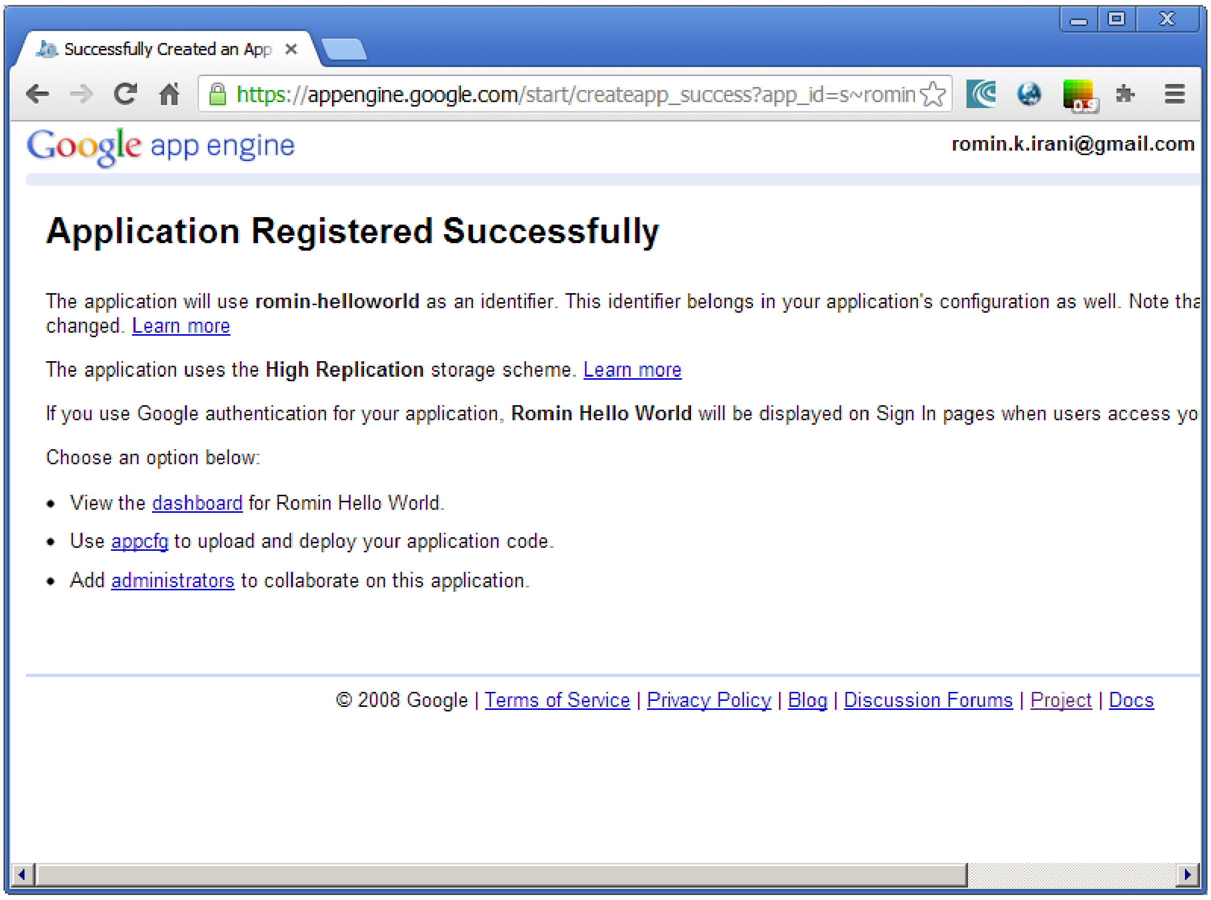
1. Click on **Create Applications** button.
2. Choose a name for your application. This will have to be a unique name and Google will verify that the name is unique. **This name is also the Application ID or Application Identifier or App Id for your application.** You can click on **Check Availability** to verify if the App Id is available for you.


Application Identifier: romin-helloworld .appspot.com

Yes, "romin-helloworld" is available!

3. Give a title to your application.
4. Select default values for **Authentication Options and Storage Options.**

5. Finally, click on **Create Application.** This will create the application and when successful, it will display a page as shown below:



Successfully Created an App
https://appengine.google.com/start/createapp_success?app_id=s~romin
Google app engine
romin.k.irani@gmail.com

Application Registered Successfully

The application will use **romin-helloworld** as an identifier. This identifier belongs in your application's configuration as well. Note tha
changed. Learn more

The application uses the **High Replication** storage scheme. Learn more

If you use Google authentication for your application, **Romin Hello World** will be displayed on Sign In pages when users access yo

Choose an option below:

- View the dashboard for Romin Hello World.
- Use appcfg to upload and deploy your application code.
- Add administrators to collaborate on this application.

© 2008 Google | Terms of Service | Privacy Policy | Blog | Discussion Forums | Project | Docs

6. When you deploy your application, it will be available at a standard URL in the form of **http://<appid>.appspot.com** **.** For e.g. if your appid is myapp1, then the application will be available on http://myapp1.appspot.com
7. If you go back to your list of applications, it will show that application in the list of applications. It will indicate that currently no version is deployed, which is what we will be doing next.

romin-helloworld	Romin Hello World	High Replication	None Deployed

The application created shows that "None Deployed". This is because we have not yet deployed any code to it, we have simply created a container with a unique name.

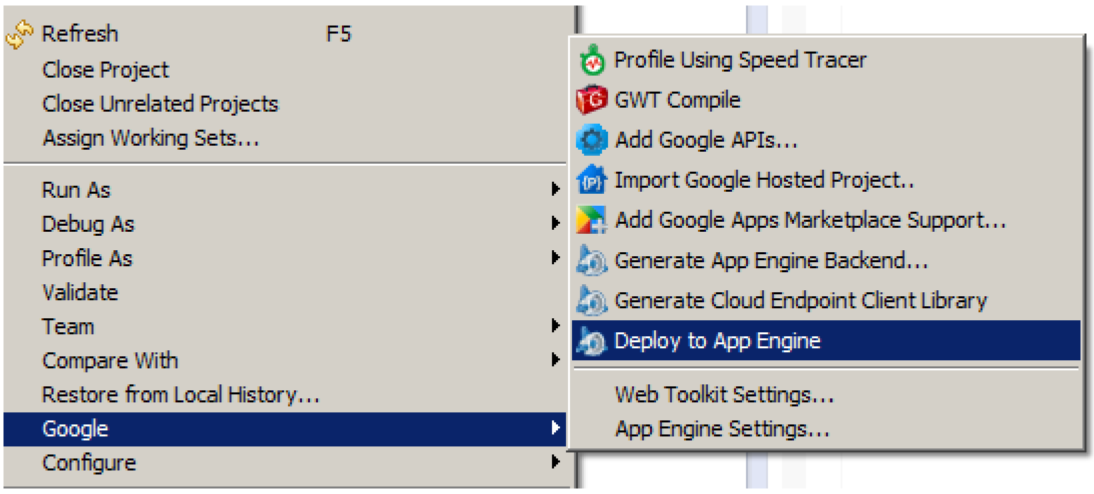
It is time now to do the second step i.e. to upload the code to it from your local Eclipse project.

Login and Deploy your application code from Eclipse

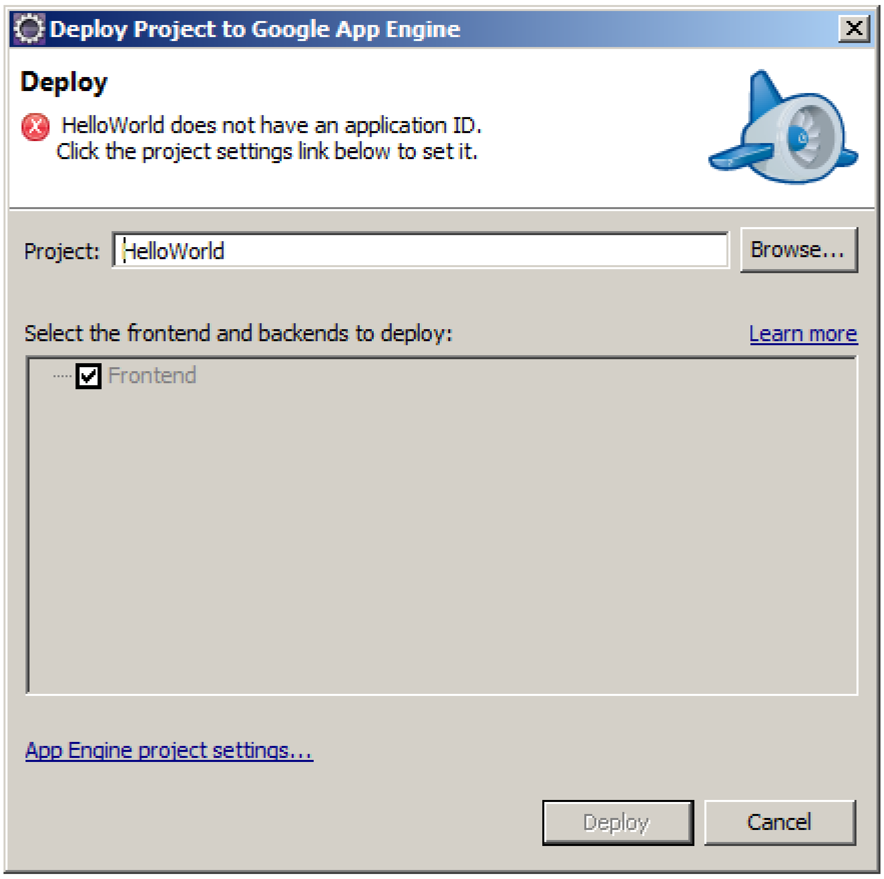
Follow the steps given below:

1. Login to your Google Account from Eclipse. You will see in the right bottom corner, the following : Sign in to Google…
2. Click on the link and follow the instructions to login to your Google App Engine Account. If it prompts you to request for your permission, please allow it by clicking on **Allow Access.**

3. Once you are successfully logged in, it will display your account id in the right bottom corner and which means that you are logged in. Eclipse will remember this across restarts too, so it is not required to login every time.
4. Now that we are logged in, it is time to deploy the application. Select the project **HelloWorld** in the Eclipse Package Explorer
5. Right-click and select **Google → Deploy to App Engine** as shown below:



6. This will bring up the dialog below that will complain that you do have **not set an application ID.**


Deploy Project to Google App Engine

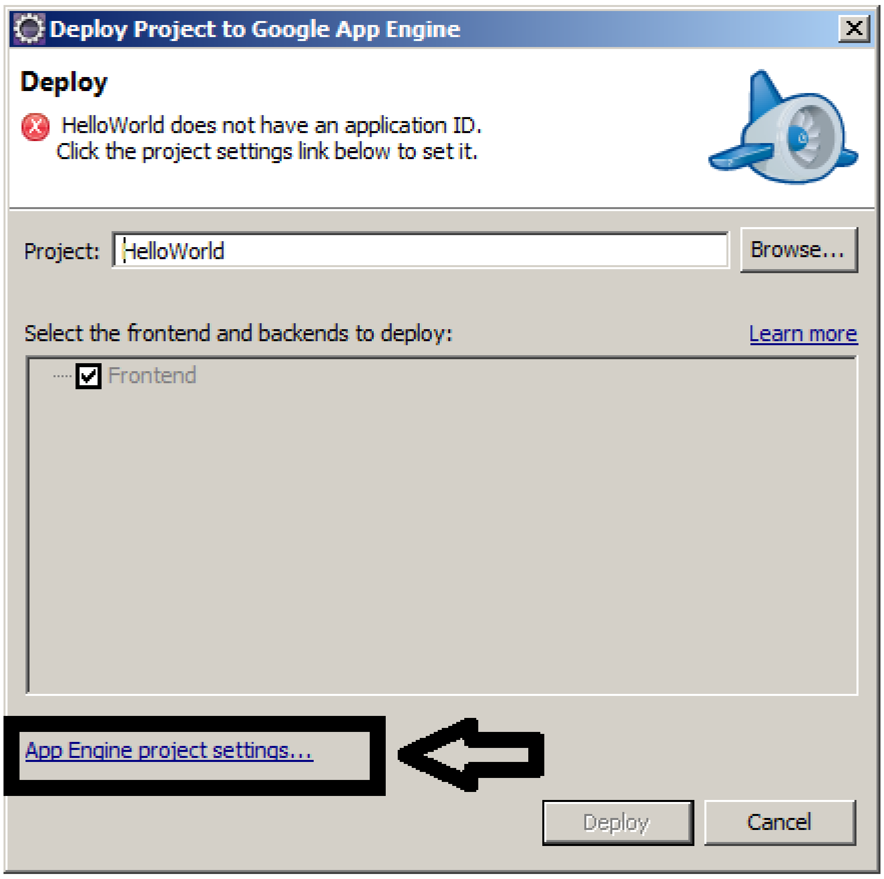
Deploy

HelloWorld does not have an application ID.
Click the project settings link below to set it.

Project: HelloWorld

Select the frontend and backends to deploy:

7. This means that we need to assign an application ID from our currently configured Applications in our account at http://appengine.google.com
8. If you have not yet created an Account, do so at http://appengine.google.com and sign up for the App Engine Service. Every account is given 10 applications and you can use any of the application Ids over here.
9. Assume that your application ID is myapp1 that is present in the list of applications listed under your account at http://appengine.google.com
10. Now click on **App Engine project settings** as shown below:



Deploy Project to Google App Engine

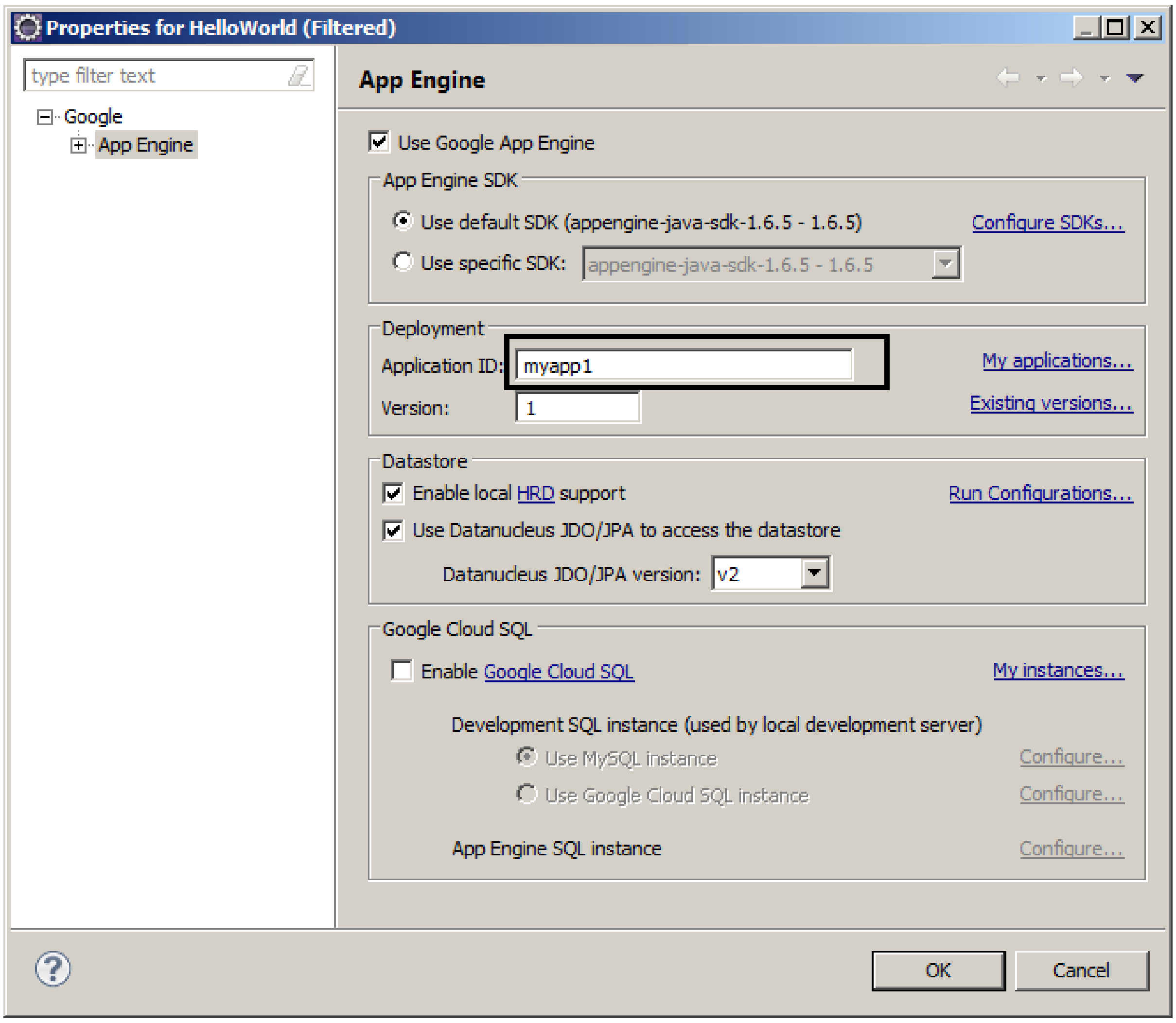
Deploy

HelloWorld does not have an application ID.
Click the project settings link below to set it.

Project: HelloWorld

Browse...
Select the frontend and backends to deploy:
Learn more
Frontend
App Engine project settings...
Deploy
Cancel

11. Enter the Application ID as shown in the dialog below:



[✓] Use Google App Engine

App Engine SDK

[●] Use default SDK (appengine-java-sdk-1.6.5 - 1.6.5) Configure SDKs...

[○] Use specific SDK: appengine-java-sdk-1.6.5 - 1.6.5

Deployment

Application ID: myapp1 My applications...

Version: 1 Existing versions...

Datastore

[✓] Enable local HRD support Run Configurations...

[✓] Use Datanucleus JDO/JPA to access the datastore

Datanucleus JDO/JPA version: v2

Google Cloud SQL

[ ] Enable Google Cloud SQL My instances...

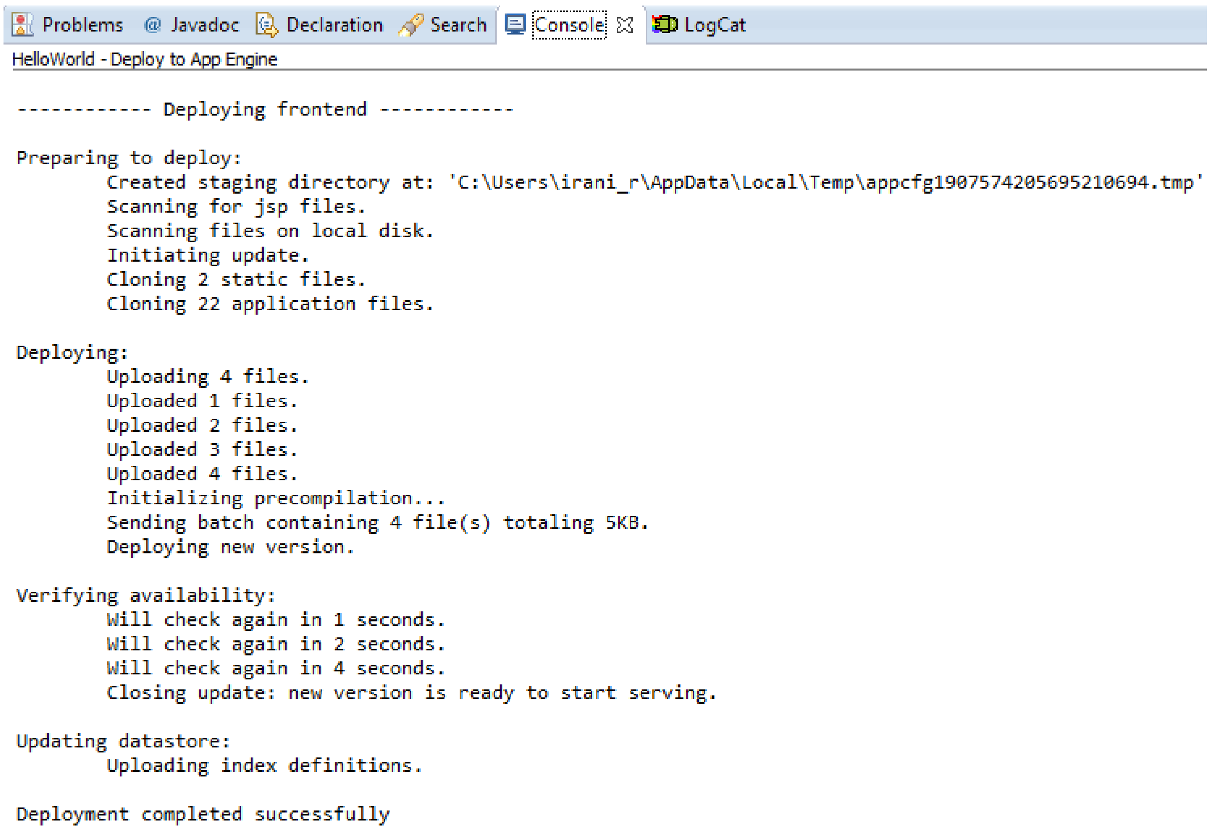
Development SQL instance (used by local development server)

[●] Use MySQL instance Configure...

[○] Use Google Cloud SQL instance Configure...

App Engine SQL instance Configure...

12. Click on OK to move forward and then **Deploy** button to deploy the application. It will take a while since this will package and upload the web application and load it to the Google Servers.
13. On successful deployment, you should see a message in the Console as shown below:



```
------------ Deploying frontend ------------

Preparing to deploy:
        Created staging directory at: 'C:\Users\irani_r\AppData\Local\Temp\appcfg1907574205695210694.tmp'
        Scanning for jsp files.
        Scanning files on local disk.
        Initiating update.
        Cloning 2 static files.
        Cloning 22 application files.

Deploying:
        Uploading 4 files.
        Uploaded 1 files.
        Uploaded 2 files.
        Uploaded 3 files.
        Uploaded 4 files.
        Initializing precompilation...
        Sending batch containing 4 file(s) totaling 5KB.
        Deploying new version.

Verifying availability:
        Will check again in 1 seconds.
        Will check again in 2 seconds.
        Will check again in 4 seconds.
        Closing update: new version is ready to start serving.

Updating datastore:
        Uploading index definitions.

Deployment completed successfully
```

14. Now you can access your application directly in the browser with the following URL: http://<app-id>.appspot.com . So assuming that that your app-id is myapp1, the url will be http://myapp1.appspot.com and you should see the same Hello World response.
15. Congratulations! You have just deployed your first application to Google App Engine.

Modify and Deploy Workflow

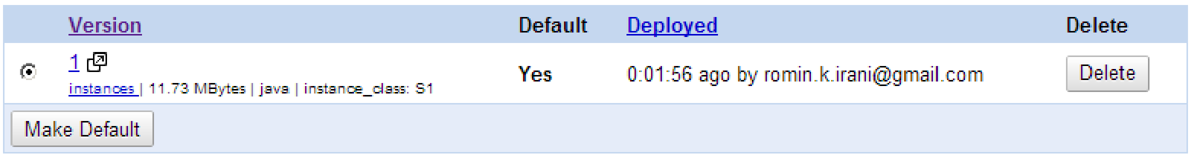
As you work on your application, the process of development and deployment is the same. The process is:

1) Make changes locally and test it out in your local Development Server. You might need to stop and start the Development Server locally for changes to take effect at times, especially loading of Java classes if the source code is changed.
2) Deploy the changes to Cloud, by clicking on the Deploy to App Engine button.

Optional

You can optionally even change the version of your application. For e.g. when deploying the application, the dialog asks you for the application Id and version. If you keep the version as same, it will keep the same version and overwrite the code of that version in the cloud. If you increment or change the version to something else, it will upload the application under that version. At a time you can only have one version as the default version.

3) To view the different versions and swap between them (if required), you should login to your account at http://appengine.google.com and visit the Administrative Console for your application. Go to **Main → Versions** and you will see the current list of versions that are deployed for your application. An example screenshot is shown below:



	Version	Default	Deployed	Delete
●	1 instances | 11.73 MBytes | java | instance_class: S1	Yes	0:01:56 ago by romin.k.irani@gmail.com	Delete

Make Default

war/WEB-INF/appengine-web.xml

Open up the appengine-web.xml now and you will find that the application ID is now set. An example is shown below. You should see your application ID in the <application> element.

```
<application>romin-helloworld</application>
<version>1</version>
```