Finolex Academy of Management & Technology, Ratnagiri

Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

PRACTICAL NO. 8

App Development using Cloud Computing

LOB7: Understand use of various tools and techniques to develop cloud applications.
LO7: Implement basic cloud applications using various tools.

Google App Engine is a scalable runtime environment mostly devoted to executing Web applications. It is a PaaS solution that enables users to host their own applications on the same or similar infrastructure as Google Docs, Google Maps, and other popular Google services. These take advantage of the large computing infrastructure of Google to dynamically scale as the demand varies over time.

App Engine provides both a secure execution environment and a collection of services that simplify the development of scalable and high-performance Web applications. These services include in-memory caching, scalable data store, job queues, messaging, and corn tasks. Developers can build and test applications on their own machines using the App Engine software development kit (SDK), which replicates the production runtime environment and helps test and profile applications. Once development is complete, developers can easily migrate their application to App Engine, set quotas to contain the costs generated, and make the application available to the world. The languages currently supported are Python, Java, and Go.

Benefits of GAE

Ease of setup and use. GAE is fully managed, so users can write code without considering IT operations and back-end infrastructure. The built-in APIs enable users to build different types of applications. Access to application logs also facilitates debugging and monitoring in production.

Pay-per-use pricing. GAE's billing scheme only charges users daily for the resources they use. Users can monitor their resource usage and bills on a dashboard.

Scalability. Google App Engine automatically scales as workloads fluctuate, adding and removing application instances or application resources as needed.

Security. GAE supports the ability to specify a range of acceptable Internet Protocol (IP) addresses. Users can allow list specific networks and services and blocklist specific IP addresses.

GAE challenges

Lack of control. Although a managed infrastructure has advantages, if a problem occurs in the back-end infrastructure, the user is dependent on Google to fix it.

Performance limits. CPU-intensive operations are slow and expensive to perform using GAE. This is because one physical server may be serving several separate, unrelated app engine users at once who need to share the CPU.

Limited access. Developers have limited, read-only access to the GAE filesystem.

Java limits. Java apps cannot create new threads and can only use a subset of the Java runtime environment standard edition classes.

Finolex Academy of Management & Technology, Ratnagiri

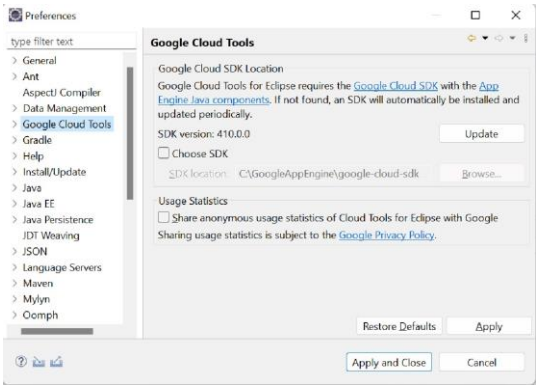
Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

Steps for Configuration of Google SDK with Eclipse –

1. Open the Eclipse IDE for installation of Google SDK plugin. Check “Java Build Path” for “JRE System Library” which must be jdk1.8.0 (jdk8). The higher versions of JDKs are not supported by Google Cloud Tools.
2. To add Google SDK solution; click Help-> Eclipse Marketplace and then search for “Google Cloud Tools for Eclipse 1.8.4” in the window. Click on Install button to install soution to the Eclipse IDE.

3. Check the Google Cloud Tools location by clicking the menu Window->Preferences-> Google Cloud Tools which will display following window –



4. If the “Google Cloud Tools” are not properly configured then you may run GoogleCloudSDKInstaller.exe for installation of Google SDK cloud tools. (You may download it from – https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe
5. The screen looks like –

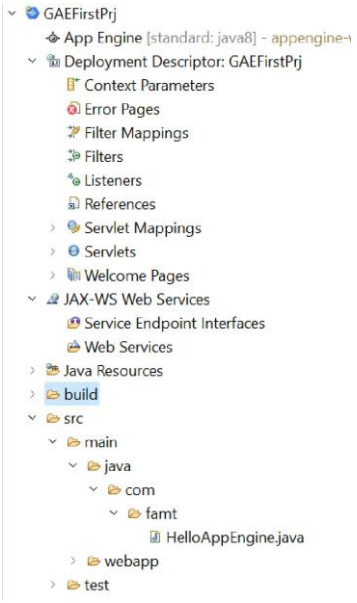
6. Default SDK install doesn’t include some extra components like App Engine extensions for Java, which you can separately install using Google Cloud SDK shell.

7. To install java component use – "gcloud components install app-engine-java" command. After running this command, it will start installation in new command prompt and complete it with the help of Internet.

Creating a Google App Engine Standard Java Project –

Create a project using **File->New->Project->Google Cloud Platform->Google App Engine Standard Java Project** and click on next for assigning **project name and java package**. Click next and add **App Engine API** from next window and click Finish.

The project explorer shows following structure –



To run the Google App Engine project click Run->Run As->App Engine which will after compilation; opens the welcome page in the default browser and displays the Welcome message.

Finolex Academy of Management & Technology, Ratnagiri

Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

Exercise:

1. Write a java program using Google App Engine for and display the message “Hello App Engine”.

   Step 1: Install Google Cloud Tools for Eclipse Open Eclipse IDE. Click on the Help tab at the top.
   Choose Eclipse Marketplace from the dropdown.
   In the Eclipse Marketplace window, search for Google Cloud Tools.
   Click Go to search for the plugin.
   Find Google Cloud Tools for Eclipse in the search results, then click Install.
   Follow the on-screen instructions to complete the installation.
   Once installed, Eclipse will prompt you to restart. Click Yes to restart Eclipse.
   Step 2: Verify Installation
   After Eclipse restarts, click on the Window tab.
   Select Preferences from the dropdown.
   In the preferences window, navigate to Google Cloud Tools.
   Check if the SDK Version is 4.10.0.0 or higher. If not, click Update to install the latest version.
   Step 3: Configure Apache Tomcat Server Runtime
   Click on the Window tab.
   Choose Preferences.
   In the preferences window, navigate to Server > Runtime Environments.
   Look for Apache Tomcat v9.0 in the list. If not present, click Add.
   Select Apache Tomcat v9.0 and specify the installation folder (e.g., C:\Program Files\Apache Tomcat).
   Click Finish to save the configuration.
   Step 4: Install JDK and JRE
   Download JDK 1.8 and JRE 1.8 from Oracle's official website.
   Install both JDK and JRE in a directory (e.g., C:\Program Files\Java).
   Make sure both are correctly installed.
   Step 5: Create a Google App Engine Standard Java Project
   In Eclipse, go to Google Cloud Platform in the top menu.
   Select Create New Project from the dropdown.
   Choose Google App Engine Standard Java Project.
   Assign a name to your project and specify a Java package.
   Click Next.
   Select App Engine API for the project.

Click Finish to create the project.
Step 6: Add Apache Tomcat as Server Runtime

Right-click on your project in the Project Explorer.
Choose Build Path > Configure Build Path.
In the Java Build Path window, click the Libraries tab.
Click Add Library.
Choose Server Runtime from the list and click Next.
Select Apache Tomcat v9.0 and click Finish.
Step 7: Verify and Set JRE System Library
Right-click on the project in Project Explorer and choose Properties.
Go to Java Build Path > Libraries.
Ensure that JRE System Library 1.8 is listed. If not:
Click Add Library.
Select JRE System Library, then click Next.
Choose Alternate JRE.
Click Search to find JDK 1.8 and select it.
Click Finish and then Apply.
Step 8: Run the Application on App Engine
Right-click on the project and select Run As > Run Configurations.
In the Run Configurations window, right-click on App Engine Standard and select New Configuration.
Under the Server tab, click Define a new server.
Choose Google from the server types and select App Engine Standard.
Provide a name for the server and specify the port number (e.g., 8080).
Click Finish.
Step 9: Deploy the Application
After setting up the server, click Run to deploy the project to App Engine.
Eclipse will use the Google Cloud SDK to deploy your project to Google Cloud.
Once deployed, the application will be accessible through the specified server URL.

Code :-

a) HelloAppEngine.java

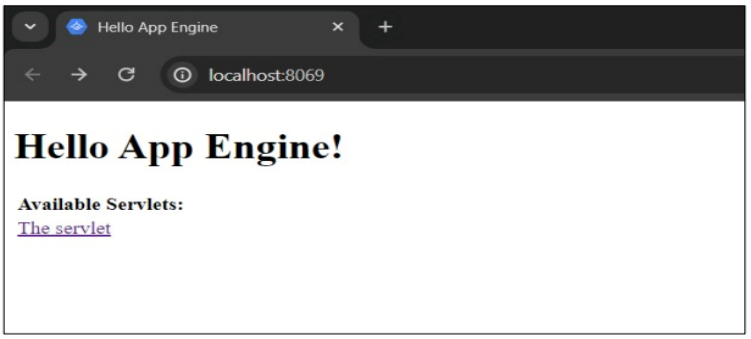
```
import java.io.IOException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

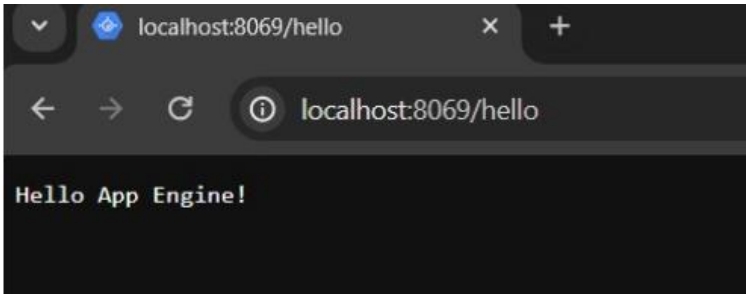
```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(
name = "HelloAppEngine",
urlPatterns = {"/hello"} )
public class HelloAppEngine extends HttpServlet {
@Override
public void doGet(HttpServletRequest request, HttpServletResponse
response)
throws IOException {
response.setContentType("text/plain");
response.setCharacterEncoding("UTF-8");
response.getWriter().print("Hello App Engine!\r\n"); } }
```

b)HelloAppEngineTest.java

```
import java.io.IOException;
import org.junit.Assert;
import org.junit.Test;
public class HelloAppEngineTest {
@Test
public void test() throws IOException {
MockHttpServletResponse response = new MockHttpServletResponse();
new HelloAppEngine().doGet(null, response);
Assert.assertEquals("text/plain", response.getContentType());
Assert.assertEquals("UTF-8", response.getCharacterEncoding());
Assert.assertEquals("Hello App Engine!\r\n",
response.getWriterContent().toString()); }
 }
```

Output :-

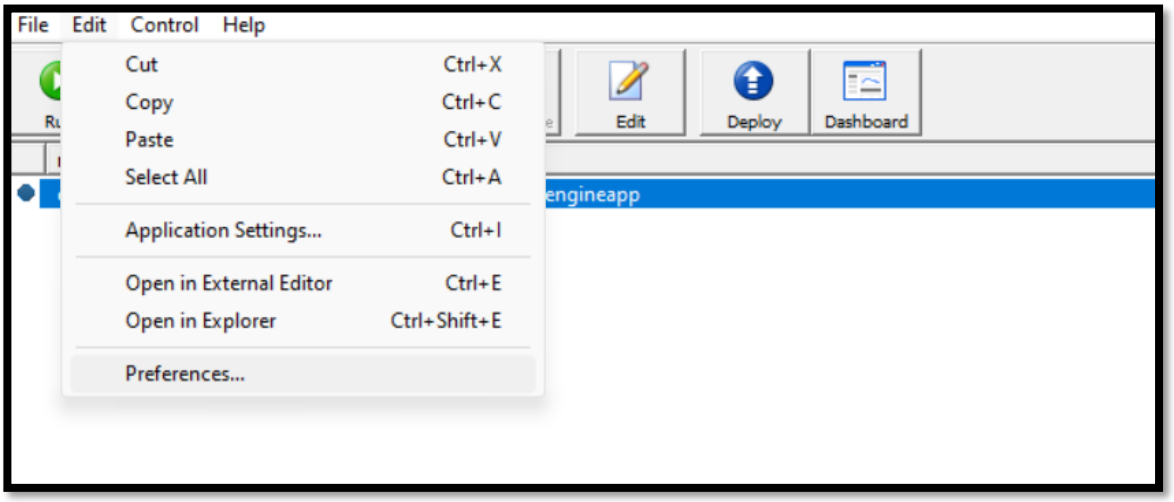




2. Write a program using google app Engine installer and display the “Hello World” Message.

Output :-

Giving python preferences

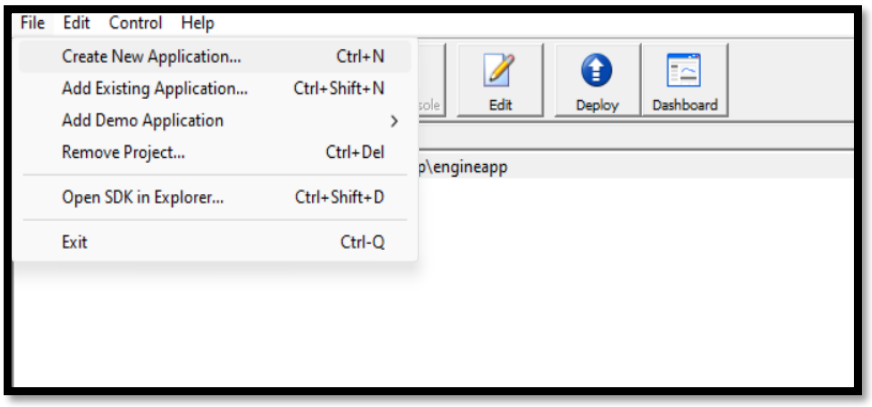


Finolex Academy of Management & Technology, Ratnagiri

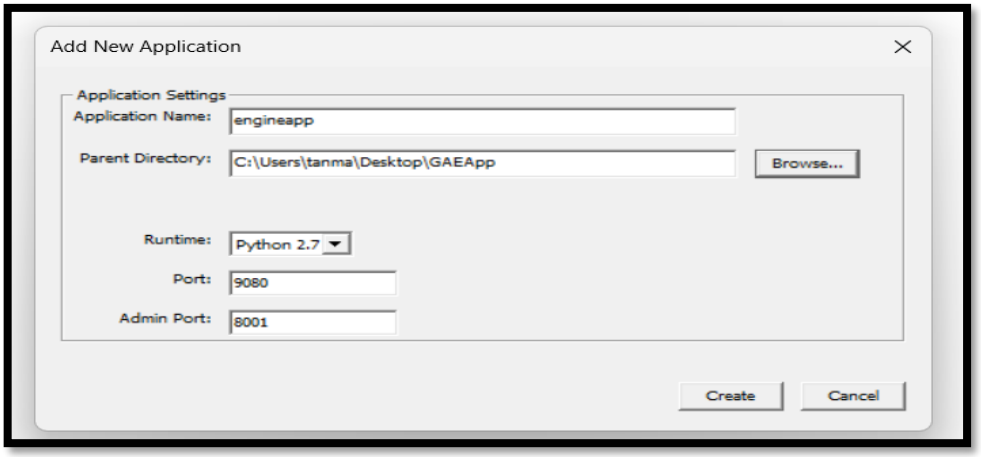
Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

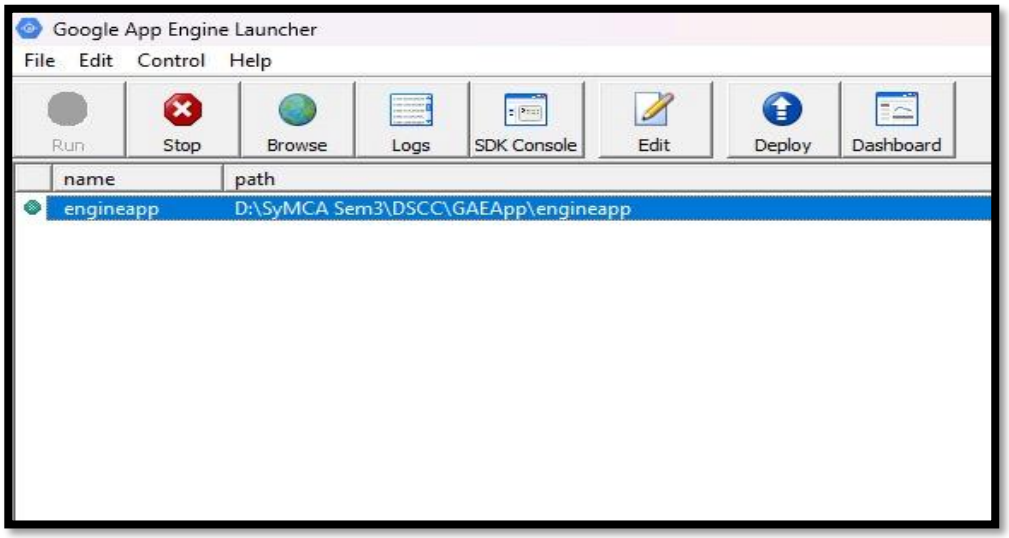
Step 1 :- Open the google app Engine installer and creating new application.

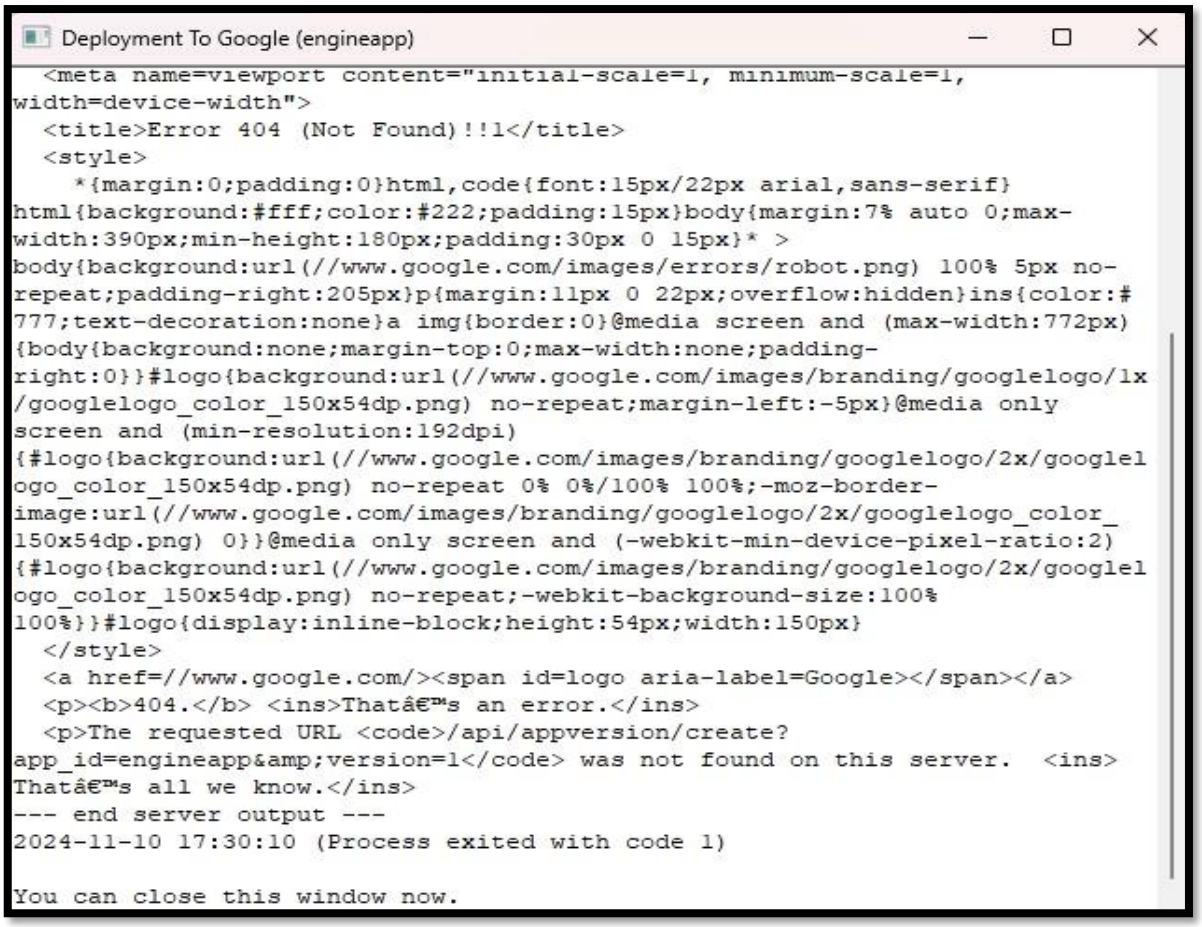


Step 2 :- giving path to the parent directory .



Step 3 :- running engine.





Output on chrome