Finolex Academy of Management & Technology, Ratnagiri

Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

PRACTICAL NO. 7

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.

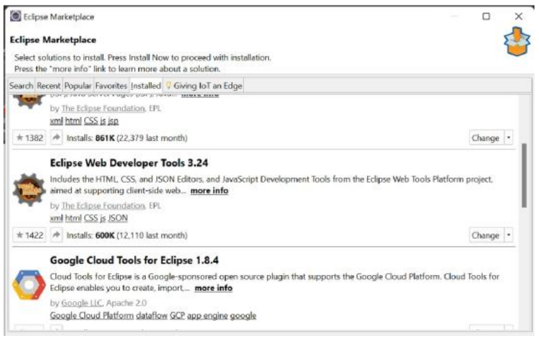
Performance limits. CPU-intensive operations are slow and expensive to perform using GAE. This is because one physical server may be serving several separate, unrelated app engine users at once who need to share the CPU.

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.

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.



Eclipse Marketplace

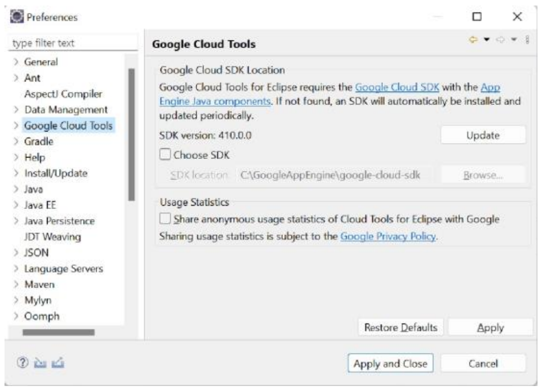
Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

by The Eclipse Foundation, EPL
xml html CSS js jsp
★ 1382 Installs: **861K** (22,379 last month)

Eclipse Web Developer Tools 3.24
Includes the HTML, CSS, and JSON Editors, and JavaScript Development Tools from the Eclipse Web Tools Platform project, aimed at supporting client-side web... **more info**
by The Eclipse Foundation, EPL
xml html CSS js JSON
★ 1422 Installs: **600K** (12,110 last month)

Google Cloud Tools for Eclipse 1.8.4
Cloud Tools for Eclipse is a Google-sponsored open source plugin that supports the Google Cloud Platform. Cloud Tools for Eclipse enables you to create, import... **more info**
by Google LLC, Apache 2.0
Google Cloud Platform dataflow GCP app engine google

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

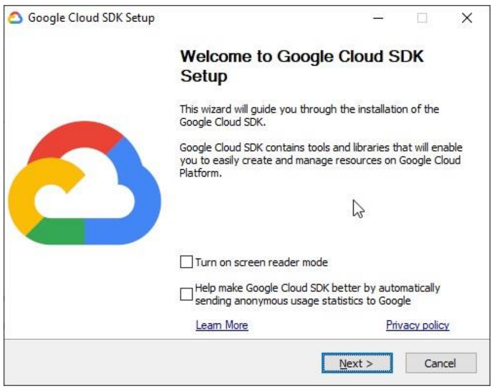


Google Cloud Tools

Google Cloud SDK Location
Google Cloud Tools for Eclipse requires the Google Cloud SDK with the App Engine Java components. If not found, an SDK will automatically be installed and updated periodically.
SDK version: 410.0.0
☐ Choose SDK
SDK location: C:\GoogleAppEngine\google-cloud-sdk

Usage Statistics
☐ Share anonymous usage statistics of Cloud Tools for Eclipse with Google
Sharing usage statistics is subject to the Google Privacy Policy.

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 –



Google Cloud SDK Setup

Welcome to Google Cloud SDK Setup

This wizard will guide you through the installation of the Google Cloud SDK.

Google Cloud SDK contains tools and libraries that will enable you to easily create and manage resources on Google Cloud Platform.

- [ ] Turn on screen reader mode
- [ ] Help make Google Cloud SDK better by automatically sending anonymous usage statistics to Google

Learn More Privacy policy

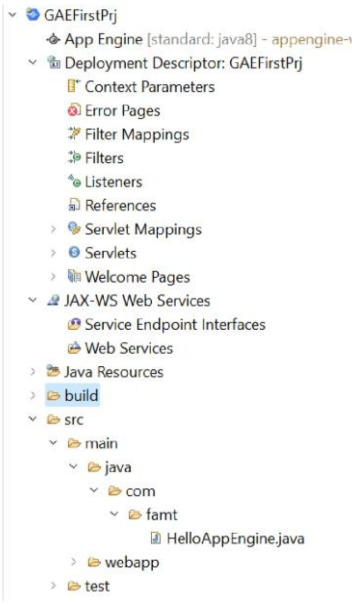
Next > Cancel

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 –



```
GAEFirstPrj
  App Engine [standard: java8] - appengine-
  Deployment Descriptor: GAEFirstPrj
    Context Parameters
    Error Pages
    Filter Mappings
    Filters
    Listeners
    References
    Servlet Mappings
    Servlets
    Welcome Pages
  JAX-WS Web Services
    Service Endpoint Interfaces
    Web Services
  Java Resources
  build
  src
    main
      java
        com
          famt
            HelloAppEngine.java
      webapp
    test
```

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.

Exercise:

1. **Write a java program using Google App Engine for checking entered number is Odd or Even.**

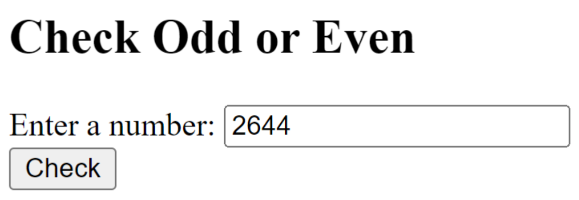
Ans :

Program:

HelloAppEngine.java

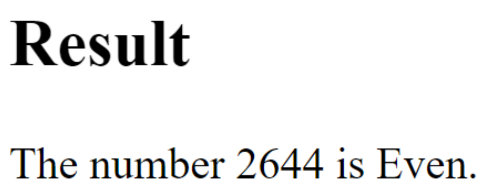
```
package com.example;
import java.io.IOException;
import java.util.Scanner;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(
    name = "HelloAppEngine",
    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.setContentType("text/html");
        response.getWriter().println("<html><body>");
        response.getWriter().println("<h2>Check Odd or Even</h2>");
        response.getWriter().println("<form method='post'>");
        response.getWriter().println("Enter a number: <input type='text'
name='number'><br>");
        response.getWriter().println("<input type='submit'
value='Check'>");
        response.getWriter().println("</form></body></html>");
    }
    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        String numberStr = request.getParameter("number");
        try {
            int number = Integer.parseInt(numberStr);
            String result = (number % 2 == 0) ? "Even" : "Odd";
            response.setContentType("text/html");
            response.getWriter().println("<html><body>");
            response.getWriter().println("<h2>Result</h2>");
            response.getWriter().println("The number " + number + " is " + result + ".");
            response.getWriter().println("</body></html>");
        } catch (NumberFormatException e) {
            response.getWriter().println("Please enter a valid number.");
        }

  }
     }
```

Output:

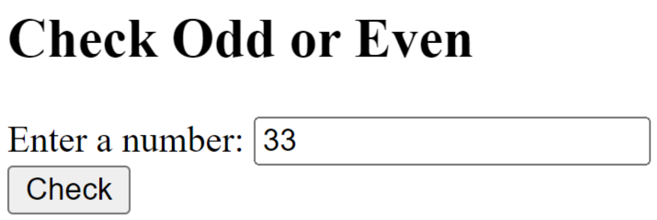

Check Odd or Even

Enter a number: 2644

Check

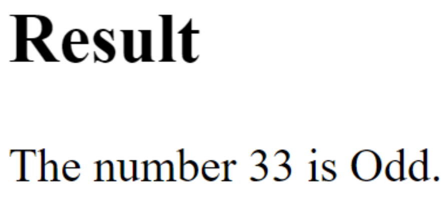

Result

The number 2644 is Even.


Check Odd or Even

Enter a number: 33

Check


Result

The number 33 is Odd.

2. **Write a java program using Google App Engine for checking entered number is Prime or not.**

Ans:

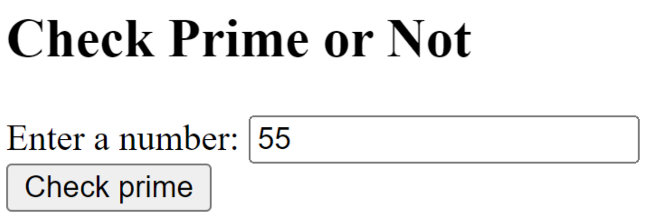
Program:

HelloAppEngine.java

```
package com.example;
import java.io.IOException;
import java.util.Scanner;
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.setContentType("text/html");
              response.getWriter().println("<html><body>");
              response.getWriter().println("<h2>Check Prime or 
Not</h2>");
              response.getWriter().println("<form method='post'>");
              response.getWriter().println("Enter a number: <input 
type='text' name='number'><br>");
              response.getWriter().println("<input type='submit' 
value='Check prime'>");
              response.getWriter().println("</form></body></html>");
          }
          @Override
          public void doPost(HttpServletRequest request, 
HttpServletResponse response) throws IOException {
              String numberStr = request.getParameter("number");
              try {
                  int number = Integer.parseInt(numberStr);
                  boolean isPrime = checkPrime(number);
                  String result = (isPrime) ? "Prime" : "Not Prime";
                  response.setContentType("text/html");
                  response.getWriter().println("<html><body>");
                  response.getWriter().println("<h2>Result</h2>");
                  response.getWriter().println("The number " + number + " 
is " + result + ".");
                  response.getWriter().println("</body></html>");
              } catch (NumberFormatException e) {
                  response.getWriter().println("Please enter a valid 
number.");
              }
          }
              private boolean checkPrime(int number) {
                  if (number <= 1) {
                  }
                  for (int i = 2; i <= Math.sqrt(number); i++) {
                      if (number % i == 0) {
                          return false;
                      }
                  }
                  return true;
              }  }
```

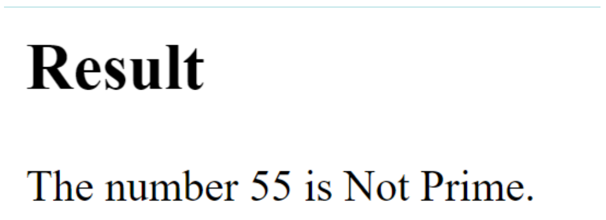
Output:



Check Prime or Not

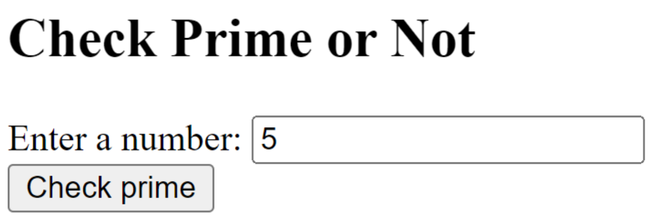
Enter a number: 55

Check prime



Result

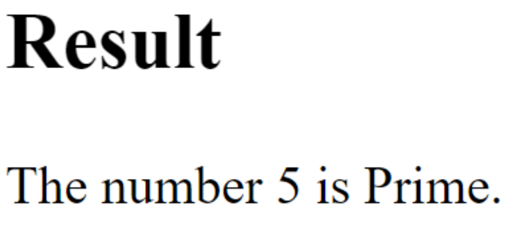
The number 55 is Not Prime.



Check Prime or Not

Enter a number: 5

Check prime



Result

The number 5 is Prime.