App Engine (https://cloud.google.com/appengine/) › Documentation (https://cloud.google.com/appengine

Using Google Cloud SQL

Python (https://cloud.google.com/appengine/docs/python/cloud-sql) | **Java** | PHP (https://cloud.google.com/appengine/docs/php/cloud-sql) | Go (https://cloud.google.com/appengine/docs/go/cloud-sql)

Google Cloud SQL provides a relational database that you can use with your App Engine application. Cloud SQL is a MySQL database (http://dev.mysql.com/doc/) that lives in Google's cloud. To learn more about Google Cloud SQL, see the Google Cloud SQL documentation (https://cloud.google.com/sql/docs/).

For information on pricing and restrictions imposed by both Cloud SQL and App Engine, see Pricing and Access Limits (https://cloud.google.com/appengine/docs/java/cloud-sql/pricing-access-limits).

Before you begin

1. Create or select a Cloud Platform project in the Cloud Platform Console and then ensure that project includes an App Engine application:

   GO TO APP ENGINE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECTOR/APPENGINE/CRI

   The **Dashboard** opens if an App Engine application already exists in your project. Otherwise, you are prompted to choose the region (https://cloud.google.com/appengine/docs/locations) where you want your App Engine application located.

2. To deploy a Java app to App Engine, you must first setup your environment, see Using Apache Maven and the App Engine Plugin (https://cloud.google.com/appengine/docs/java/tools/maven) for details.

Configure your local environment

You can either use a local MySQL server to test your application or you can connect to Cloud SQL.

1. If you want to test your application with a local MySQL server, install it now.
   If you use Linux on a distribution with `apt-get`, you can run:

   ```
   sudo apt-get install mysql-server
   ```

   For other operating systems, see the MySQL Community Server download page (http://dev.mysql.com/downloads/mysql/).

Setting up

1. In the same project as your App Engine application, create a Second Generation Cloud SQL instance and configure the root user (https://cloud.google.com/sql/docs/create-instance#create-2nd-gen).
2. If you don't want to use the root user to connect, create a user (https://cloud.google.com/sql/docs/create-user).
3. Using the Cloud SDK, get the Cloud SQL instance connection name to use as a connection string in your application code:

   ```
   gcloud sql instances describe [INSTANCE_NAME]
   ```

   Record the value returned for `connectionName`. You can also find this value in the Instance details page of the Google Cloud Platform Console. For example, in the Cloud SDK output:

   ```
   $ gcloud sql instances describe instance1
   connectionName: project1:us-central1:instance1
   ```

4. Add the Cloud SQL instance connection name, database, user, and password to `appengine-web.xml`.

[appengine/cloudsql/src/main/webapp/WEB-INF/appengine-web.xml](https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/appengine/cloudsql/src/main/webapp/WEB-INF/appengine-web.xml)
 (https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/appengine/cloudsql/src/main/webapp/WEB-INF/appengine-web.xml)

SAMPLES/BLOB/MASTER/APPENGINE/CLOUDSQL/SRC/MAIN/WEBAPP/WEB-INF/APPENGINE-WEB.XML)

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <threadsafe>true</threadsafe>
  <use-google-connector-j>true</use-google-connector-j>
  <system-properties>
    <property name="ae-cloudsql.cloudsql-database-url" value="jdbc:google:m
    <property name="ae-cloudsql.local-database-url" value="jdbc:mysql://goo
  </system-properties>
</appengine-web-app>
```

5. Add a JDBC library to your application. For example, if you use Maven, you can add the dependency to the project's **`pom.xml`**:

[appengine/cloudsql/pom.xml](https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/appengine/cloudsql/pom.xml)
 (https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/appengine/cloudsql/pom.xml)

)M/GOOGLECLOUDPLATFORM/JAVA-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/CLOUDSQL/POM.XML)

```
<dependency> <!-- ONLY USED LOCALY -->
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.40</version>  <!-- v5.x.x is Java 7, v6.x.x is Java 8 -->
</dependency>
<dependency>
  <groupId>com.google.cloud.sql</groupId>
  <artifactId>mysql-socket-factory</artifactId>
  <version>1.0.2</version>
</dependency>
```

Code sample overview

The following code sample creates a visitor log in a Cloud SQL instance. It writes visit information to Cloud SQL and then reads and returns the last ten visits:

appengine/cloudsql/src/main/java/com/example/appengine/cloudsql/CloudSqlServlet.java (https://github.com/GoogleCloudPlatform/java-docs-samples/blob/master/appengine/cloudsql/src/main/java/com/example/appengine/cloudsql/CloudSqlSer

NGINE/CLOUDSQL/SRC/MAIN/JAVA/COM/EXAMPLE/APPENGINE/CLOUDSQL/CLOUDSQLSERVLET.JAVA)

```
@SuppressWarnings("serial")
public class CloudSqlServlet extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOE
      ServletException {
    String path = req.getRequestURI();
    if (path.startsWith("/favicon.ico")) {
      return; // ignore the request for favicon.ico
    }
    // store only the first two octets of a users ip address
    String userIp = req.getRemoteAddr();
    InetAddress address = InetAddress.getByName(userIp);
    if (address instanceof Inet6Address) {
      // nest indexOf calls to find the second occurrence of a character in a st
      // an alternative is to use Apache Commons Lang: StringUtils.ordinalIndexO
      userIp = userIp.substring(0, userIp.indexOf(":", userIp.indexOf(":") + 1))
    } else if (address instanceof Inet4Address) {
      userIp = userIp.substring(0, userIp.indexOf(".", userIp.indexOf(".") + 1))
    }

    final String createTableSql = "CREATE TABLE IF NOT EXISTS visits ( visit_id
        + "AUTO_INCREMENT, user_ip VARCHAR(46) NOT NULL, timestamp DATETIME NOT
        + "PRIMARY KEY (visit_id) )";
    final String createVisitSql = "INSERT INTO visits (user_ip, timestamp) VALUE
    final String selectSql = "SELECT user_ip, timestamp FROM visits ORDER BY tim
        + "LIMIT 10";

    PrintWriter out = resp.getWriter();
    resp.setContentType("text/plain");
    String url;
    if (System
        .getProperty("com.google.appengine.runtime.version").startsWith("Google
      // Check the System properties to determine if we are running on appengine
      // Google App Engine sets a few system properties that will reliably be pr
      // instance.
      url = System.getProperty("ae-cloudsql.cloudsql-database-url");
      try {
        // Load the class that provides the new "jdbc:google:mysql://" prefix.
    Class.forName("com.mysql.jdbc.GoogleDriver");
   } catch (ClassNotFoundException e) {
    throw new ServletException("Error loading Google JDBC Driver", e);
   }
  } else {
   // Set the url with the local MySQL database connection url when running l
   url = System.getProperty("ae-cloudsql.local-database-url");
  }
  log("connecting to: " + url);
  try (Connection conn = DriverManager.getConnection(url);
    PreparedStatement statementCreateVisit = conn.prepareStatement(createVis
   conn.createStatement().executeUpdate(createTableSql);
   statementCreateVisit.setString(1, userIp);
   statementCreateVisit.setTimestamp(2, new Timestamp(new Date().getTime()));
   statementCreateVisit.executeUpdate();

   try (ResultSet rs = conn.prepareStatement(selectSql).executeQuery()) {
    out.print("Last 10 visits:\n");
    while (rs.next()) {
     String savedIp = rs.getString("user_ip");
     String timeStamp = rs.getString("timestamp");
     out.print("Time: " + timeStamp + " Addr: " + savedIp + "\n");
    }
   }
  } catch (SQLException e) {
   throw new ServletException("SQL error", e);
  }
 }
}
```

Testing in your development environment

To test your app with the local development server:

1. If you are using a local MySQL server, start the MySQL server in your development environment.
2. Start the development server. For example, if you use Maven:

   ```
   mvn appengine:run
   ```

3. The web server is now running and listening for requests on port 8080. To view, visit the following URL:
   http://localhost:8080/ (http://localhost:8080/)

Something go wrong? See Using the Local Development Server (https://cloud.google.com/appengine/docs/java/tools/using-local-server) for more information.

Deploying your app

To upload your app to App Engine, run the following commands:

```
mvn clean package
mvn appengine:deploy
```

For details about deploying to App Engine, see Deploying a Java App (https://cloud.google.com/appengine/docs/java/tools/uploadinganapp).

Except as otherwise noted, the content of this page is licensed under the ***Creative Commons Attribution 3.0 License*** (http://creativecommons.org/licenses/by/3.0/)*, and code samples are licensed under the* ***Apache 2.0 License*** (http://www.apache.org/licenses/LICENSE-2.0)*. For details, see our* ***Site Policies*** (https://cloud.google.com/site-policies)*. Java is a registered trademark of Oracle and/or its affiliates.*

Última actualización: Diciembre 15, 2016.