Hands-on: Introduction to Datastore

Updated: March 20, 2015

This exercise leads you through creating an App Engine application that integrates Datastore API.

Contents

What You Will Learn

You will learn:

- How to use the Cloud Datastore in your Java or Python App Engine application
- How to use the Datastore Viewer to view the data

What You Will Do

You will create an App Engine application in either Java or Python. You will then integrate the Datastore API to understand how it works.

Get Set Up

1. If you need to deploy your application to App Engine production, you will need to have an application ID (project ID) in order to deploy your app to production App Engine. If you already have a

project to test, you can skip this section. Otherwise, create a new project as follows:

a. Visit Google Developers Console in your web browser, and click **Create Project**.
b. Supply an appropriate project name and accept the project ID that is auto-generated for you. Please note down the application id that has been generated. You will need that if you deploy the application.
c. Click **Create**
d. Enable billing.

Creating the Application

Before we move on with the instructions and code, here is what we are going to do:

1. We are going to create a **Conference** entity, which has attributes and data types as follows:
   a. Conference Title (**title)** : String
   b. City in which the conference will be held (**city)** : String
   c. Maximum number of Attendees allowed (**maxAttendees**) : Integer
   d. Conference Start Date (**startdate**) : Date
   e. Conference End Date (**enddate**) : Date
2. We will create a HTML Web Form (**Create Conference Web Form)** that will provide a simple data entry mechanism for the above Conference entity attributes. On submission of the form, a handler will extract out the HTTP request parameter and create the **Conference** entity in the datastore.
3. We will then redirect the form back to the original Web Form, where we shall use the Datastore Query API to retrieve all the current Conference entities present in the Datastore.

If you are creating the application on your own, please follow the respective sections for your language (Java or Python).

We have the final Project ZIP files available for both Java and Python. You can use them if you want but we do encourage you to try it out on your own. The files are given below:

- Java Datastore Basics Project ZIP file
- Python Datastore Basics Project ZIP file

You can also clone the projects directly from GitHub:

```
git clone https://github.com/GoogleCloudPlatformTraining/cp300-gae-datastore-java.git
```

OR

```
git clone https://github.com/GoogleCloudPlatformTraining/cp300-gae-datastore-python.git
```

Java Instructions

If you have cloned the working project from Git, you will have the Maven project present in **cp300-gae-datastore-java**. You do not need to create any other files.

If you plan to create the application from scratch, follow these steps:

1. Use App Engine Maven to create a blank Java Project via the command line / terminal given below:

   mvn archetype:generate -Dappengine-version=1.9.18 -Dapplication-id=**your-app-id** -Dfilter=com.google.appengine.archetypes:appengine-skeleton-archetype -DgroupId=com.mycompany.app -DartifactId=**cp300-gae-datastore-java**

   Choose all the defaults. This will create a blank Java App Engine project in the **cp300-gae-datastore-java** folder.
2. If you had cloned the project from Git, then you can go directly to building and running the app locally. If you created the project from scratch above, then follow the next steps given below:

3. Go ahead and create a Java servlet that will handle the **/create** request. The Java source file is shown below:

   CreateConference.java

```
package com.mycompany.app;

import java.io.IOException;
import java.io.PrintWriter;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.logging.Logger;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.appengine.api.datastore.DatastoreService;
import com.google.appengine.api.datastore.DatastoreServiceFactory;
import com.google.appengine.api.datastore.Entity;
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;

@SuppressWarnings("serial")
public class CreateConference extends HttpServlet {

  public static final Logger _LOG =
Logger.getLogger(CreateConference.class.getName());

  // Get a handle to the DatastoreService
  public static DatastoreService datastore =
DatastoreServiceFactory.getDatastoreService();

    public void doPost(HttpServletRequest req, HttpServletResponse resp) throws
IOException {

    PrintWriter writer = resp.getWriter();

    // Get the values sent by the form
    String title = req.getParameter("title");
    String city = req.getParameter("city");
    String maxAttendees = req.getParameter("maxAttendees");
    String confStartDate = req.getParameter("startdate");
    String confEndDate = req.getParameter("enddate");

    try {
       // Convert maxAttendees to a Long before saving it
        Long mL = Long.parseLong(maxAttendees);

        // Convert the startDate and endDate to Dates
        // The format is 2013-04-26

        Date startDate;
        try {
          startDate = new SimpleDateFormat("yyyy-MM-dd",
Locale.ENGLISH).parse(confStartDate);
        } catch (ParseException e) {
          startDate = null;
        }
        Date endDate;
        try {
          endDate = new SimpleDateFormat("yyyy-MM-dd",
Locale.ENGLISH).parse(confEndDate);
        } catch (ParseException e) {
          endDate = null;
        }

        // Create an entity of kind "Conference"
        Entity confEntity = new Entity("Conference");

        confEntity.setProperty("title", title);
        confEntity.setProperty("city", city);
        confEntity.setProperty("maxAttendees", mL);
        confEntity.setProperty("startdate", startDate);
        confEntity.setProperty("enddate", endDate);

        // Save the entity in the datastore
        Key confKey = datastore.put(confEntity);

           // Get a string of the conference entity's key
           String confKeyString = KeyFactory.keyToString(confKey);
```

```
            _LOG.info("Conference Entity with Key = " + confKeyString + "
created");

      resp.sendRedirect("/");
    } catch (Exception e) {
      resp.setContentType("text/html");
      writer.println("Exception in saving entity. Reason : " + e.getMessage());
    }
  }

  /*
   * If a user comes to this page as a GET, redirect to the Create Conference
page
   */

  public void doGet(HttpServletRequest req, HttpServletResponse resp) throws
IOException {
    resp.sendRedirect("/createconference");
  }
}
```

Add the following servlet and welcome-file-list entry to the WEB-INF/web.xml file:

```
   <servlet>
      <servlet-name>CreateConference</servlet-name>
      <servlet-class>com.mycompany.app.CreateConference</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>CreateConference</servlet-name>
      <url-pattern>/create</url-pattern>
    </servlet-mapping>

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

Create the **index.jsp** file as shown below:

```
<%@ page import="java.util.*" %>
<%@ page import="com.google.appengine.api.datastore.*" %>
<%@ page import="com.google.appengine.api.datastore.Query.*" %>
<%@ page import="java.text.DateFormat" %>

<%
    // Get a handle to the datastore service
    DatastoreService datastore = DatastoreServiceFactory.getDatastoreService();
    // Create the Query to get all the conferences
    Query confQuery = new Query("Conference");
    confQuery.addSort("title", SortDirection.ASCENDING);
    // Submit the query
     PreparedQuery results = datastore.prepare(confQuery);
  %>
  <!DOCTYPE html>
<html>
  <h1>Conference List</h1>
  <body>
       <table border="1">
         <tr><th>Conference Title</th><th>City</th><th>Max
Attendees</th><th>Start Date</th><th>End Date</th></tr>
         <%
                  if (results != null) {
              for (Entity conference : results.asIterable()) {
                   String title = conference.getProperty("title").toString();
                          String city =
conference.getProperty("city").toString();
                   String maxAttendees =
conference.getProperty("maxAttendees").toString();
                  // Get the start and end date.
                 // Check the dates exist, and then convert them to a printable
format.
                String startDateString = "";
                Date startDate = (Date) conference.getProperty("startdate");
                if (startDate != null) {
                 startDateString =
DateFormat.getDateInstance(DateFormat.MEDIUM)

.format(startDate);
                        }
                String endDateString = "";
                Date endDate = (Date) conference.getProperty("enddate");
                if (endDate != null) {
                    endDateString =
DateFormat.getDateInstance(DateFormat.MEDIUM)

.format(endDate);
                }
             %>
                 <tr>
           <td><%=title%></td>
             <td><%=city%></td>
             <td><%=maxAttendees%></td>
             <td><%=startDateString%></td>
             <td><%=endDateString%></td>
            </tr>
         <%
                   }
                  }
          %>
</table>

<h1>Schedule new Conference</h1>

  <form action="/create" method="post">
         <p><b>What is the title of your conference? </b><input name="title"
size="30"/></p>

         <p><b>Where would you like to hold your conference? </b><input
name="city" size="30">
             </input></p>

         <p><b>What is the maximum number of attendees? </b>
         <input name="maxAttendees" value="5" /><i>Must be an integer</i></p>

         <p><b>What date does your conference start? </b><input name="startdate"
type="date">

         <p><b>What date does your conference end? </b><input name="enddate"
type="date"></p>
```

```
         <p><input type=submit value="Schedule conference" id=scheduleconference
/></p>
  </form>
  </body>
</html>
```

To build and run the Java App Engine project, execute the following Maven commands from the command line / terminal. These commands should be executed from the **cp300-gae-datastore-java** folder i.e. where your pom.xml file is present:

1. To build the project

   ```
   mvn package
   ```

2. To run the application in the local devserver :

   ```
   mvn appengine:devserver
   ```

To test out the application locally, try the following in your local browser:

1. Visit http://localhost:<PORT_NUMBER>.
2. This should bring up the Create Conference Form. Notice that since we have not created any data records, the list of conferences is empty.
3. Go ahead and create a conference. Fill out all the values (do not leave any empty values). Click on Schedule Conference to submit the form.
4. Once the conference has been successfully created, the browser will be redirected back to the same page and the newly created record will be shown in the list.
5. Observe that since the AppEngine Datastore is an eventually consistent one, there could be a delay before you see the records
6. Visit the DevServer Admin application by visiting http://localhost:8080/_ah/admin. Click on Datastore Viewer and see the Conference entity records in the list.

Python Instructions

Here is the **app.yaml** for the application:

```
application: your-app-id
version: 1
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: /.*
   script: datastore101.application
```

```
libraries:
- name: webapp2
   version: latest
- name: jinja2
   version: latest
```

Create **datastore.py** as shown below:

```
import jinja2
import os
import webapp2
import datetime

from google.appengine.ext import ndb

jinja_environment = jinja2.Environment(

loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
                                     extensions=['jinja2.ext.autoescape'],
autoescape=True)


class Conference(ndb.Model):

    title = ndb.StringProperty()
    city = ndb.StringProperty()
    startdate = ndb.DateProperty(indexed=False)
    enddate = ndb.DateProperty(indexed=False)
    maxAttendees = ndb.IntegerProperty()


class MainPage(webapp2.RequestHandler):

    def get(self):

        conference_values =
        {"conferences":Conference.query().order(Conference.title)}
        template = jinja_environment.get_template('index.html')
        self.response.out.write(template.render(conference_values))


class CreateConference(webapp2.RequestHandler):
    def post(self):

        conference = Conference()
        conference.title = self.request.get('title')
        conference.city = self.request.get('city')
        conference.startdate =
datetime.datetime.strptime(self.request.get('startdate'), '%Y-%m-%d').date()
        conference.enddate = datetime.datetime.strptime(self.request.get('enddate'),
'%Y-%m-%d').date()
        conference.maxAttendees = int(self.request.get('maxAttendees'))
        conference.put()
        self.redirect('/')

application = webapp2.WSGIApplication([
     ('/', MainPage),
     ('/create', CreateConference),
], debug=True)
```

Create **index.html** file as shown below:

```
<!DOCTYPE html>

{% autoescape true %}

<html>
  <h1>Conference List</h1>
  <body>
      <table border="1">
            <tr><th>Conference Title</th><th>City</th><th>Max Attendees</th><th>Start
                           Date</th><th>End Date</th></tr>

      {% for conference in conferences %}

      <tr>

           <td>{{conference.title}}</td>

              <td>{{conference.city}}</td>

              <td>{{conference.max_attendees}}</td>

              <td>{{conference.start_date}}</td>

              <td>{{conference.end_date}}</td>
              </tr>

      {% endfor %}

    </table>

  <h1>Schedule new Conference</h1>

  <form action="/create" method="post">

      <p><b>What is the title of your conference? </b><input name="title"
size="30"/></p>

      <p><b>Where would you like to hold your conference? </b><input name="city"
size="30"/></p>

      <p><b>What is the maximum number of attendees? </b>

         <input name="maxAttendees" value="5" /><i>Must be an integer</i></p>

         <p><b>What date does your conference start? </b><input name="startdate"
type="date">

         <p><b>What date does your conference end? </b><input name="enddate"
type="date"></p>

         <p><input type=submit value="Schedule conference" id=scheduleconference /></p>

  </form>

  </body>

</html>

{% endautoescape %}
```

Run the Python Application locally via your IDE or if you prefer via the terminal/ command line using the following:

```
dev_appserver.py /<your_python_app_dir>
```

To test out the application locally, try the following in your local browser:

1. Visit http://localhost:<PORT_NUMBER>. (The default is 8080)

2. This should bring up the Create Conference Form. Notice that since we have not created any data records, the list of conferences is empty.
3. Go ahead and create a conference. Fill out all the values (do not leave any empty values). Click on Schedule Conference to submit the form.
4. Once the conference has been successfully created, the browser will be redirected back to the same page and the newly created record will be shown in the list.
5. Observe that since the AppEngine Datastore is an eventually consistent one, there could be a delay before you see the records
6. Visit the DevServer Admin application by visiting http://localhost:8000. Click on Datastore Viewer and see the Conference entity records in the list.

Deploying the Application and Using Datastore Viewer

Optionally, if you want, you can deploy the application to App Engine and test it out. You can view Datastore records in the Admin Console.

1. Deploy the application to App Engine. The application will be available over http://<APP_ID>.appspot.com
2. Create a few Conference Records.
3. In the Admin Console, go to the Data section in the left hand navigation panel, select Datastore Viewer.
4. Review the Datastore records for Conference entity.

Additional Resources

- Developer Documentation:
  - Storing Data (Java) (Python)
  - Datastore Overview (Java) (Python)
  - Entities, Properties and Keys (Java) (Python)

Articles:

- Life of a Datastore write

Summary

In this exercise, you learned how to use Datastore API in your application. You also learned how to use the Admin Console to view current Datastore records.