Hands-on: Memcache

Updated: March 20, 2015

This exercise leads you through creating an App Engine application that illustrates usage of memcache.

Contents

What You Will Learn

You will learn:

- How to use the Memcache API in your Java or Python App Engine application
- Use Memcache Viewer to observe the memcache entries

What You Will Do

You will create an App Engine application in either Java or Python. You will then integrate the Memcache 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 will write a handler for the App Engine application. This handler will take a request parameter named **rssfeedURL**. This parameter will contain the value for the RSS Feed that we want to fetch in our application.
2. We will read the request parameter in the handler and will not actually make the network call to get the feed. What we shall do is the following:
    a. Look up the value for the rssFeedURL in memcache via its key i.e. 'rssFeedURL'.
    b. If the key is present, we shall read its value and return it back. This means we returned the results directly from memcache itself.
    c. If the key is not present, we shall assume that the call is made to the network and then place the key and its value in memcache, such that it is available in the cache should a call be made subsequently to get the feed value. We shall also attach an expiration time of 60 seconds to the cache entry. What this means is that once the 60 seconds have expired, memcache will automatically remove this entry.

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 Memcache Basics Project ZIP file
- Python Memcache Basics Project ZIP file

You can also clone the projects directly from GitHub:

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

OR

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

Java Instructions

If you have cloned the working project from Git, you will have the Maven project present in **cp300-gae-memcache-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-memcache-java**

   Choose all the defaults. This will create a blank Java App Engine project in the **cp300-gae-memcache-java** folder.

2. You will now need to add the RSSFeedServlet.java and modify the web.xml file as explained below. If you had cloned the project from Git, these files are already present.

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

RSSFeedServlet.java

```
package com.mycompany.app;
import java.io.IOException;
import java.util.Date;
import java.util.logging.Logger;
import javax.servlet.http.*;

import com.google.appengine.api.memcache.Expiration;
import com.google.appengine.api.memcache.MemcacheService;
import com.google.appengine.api.memcache.MemcacheServiceFactory;

@SuppressWarnings("serial")
public class RSSFeedServlet extends HttpServlet {
  private static final Logger LOG = Logger.getLogger(RSSFeedServlet.class.getName());

  public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    //Get a Handle to the Memcache Service
    MemcacheService cache = MemcacheServiceFactory.getMemcacheService();
    //Read the request Parameter
    String rssFeedURL = req.getParameter("rssFeedURL");
    String result;

    //Check if Key is present in the Cache.
    //If not present, add it to the Cache with Expiration of 60 seconds
    //else if available, simply return the result from the Cache
    if (!cache.contains(rssFeedURL)) {
      result = "Loaded into cache at " + (new Date());
      cache.put(rssFeedURL, result,Expiration.byDeltaSeconds(60));
      LOG.info(result);
      }
    else {
```

```
    result = "From Cache : " + cache.get(rssFeedURL);
    LOG.info(result);
    }
   resp.getWriter().println(result);
   }
 }
```

Add the following servlet entries to the WEB-INF/web.xml file

```
<servlet>
  <servlet-name>RSSFeedServlet</servlet-name>
  <servlet-class>com.mycompany.app.RSSFeedServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>RSSFeedServlet</servlet-name>
  <url-pattern>/feed</url-pattern>
</servlet-mapping>
```

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-memcache-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:8080/feed?rssFeedURL=http://myurl1.rss. It should print out

   ```
   Loaded into cache at  <Date Time>
   ```

2. Visit the above URL again within a minute. It should give you the value from memcache.

   ```
   From Cache : Loaded into cache at Mon Jan 12 09:31:44 UTC 2015
   ```

3. Wait for a minute or more and try out the same URL. What value does it give now ?
4. Try different URLs.

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: helloworld.application
```

The **helloworld.py** source code is shown below:

```
import webapp2
import time
from google.appengine.api import memcache

class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.write('Hello World')

class FeedPage(webapp2.RequestHandler):
    def get(self):

        #Read the request parameter
        rssFeedURL = self.request.get("rssFeedURL")
        result = ''

        # Check if the key is present in the Cache
        data = memcache.get(rssFeedURL)
        #If key is not present, put the entry in the Cache with expiry of 60 seconds
        if (data is None):
            result = 'Loaded into cache at ' + time.asctime(time.gmtime(time.time())
            memcache.add(rssFeedURL,result,60)
        else:
           # Return the value from the Cache itself
           result = 'From Cache : ' + data
        self.response.write(result)
application = webapp2.WSGIApplication([
    ('/', MainPage),
    ('/feed', FeedPage)
], debug=True)
```

Run the Python Application locally via your IDE or if you prefer the terminal/ command line via 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/feed?rssFeedURL=http://myurl1.rss. It should print out

   ```
   Loaded into cache at  <Date Time>
   ```

2. Visit the above URL again within a minute. It should give you the value from memcache itself.

   ```
   From Cache : Loaded into cache at Mon Jan 12 09:31:44 UTC 2015
   ```

3. Wait for a minute or more and try out the same URL. What value does it give now?
4. Try different URLs.

Deploying the Application and Observing Memcache Data

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

1. Deploy the application to App Engine. The application will be available over http://<APP_ID>.appspot.com
2. Invoke a few requests to your application in the format given below: http://<APP_ID>.appspot.com/feed?rssFeedURL=<some_url>. Try it out with different URLs. Some examples are given below:
   a. /feed?rssFeedURL=http://myurl1.rss
   b. /feed?rssFeedURL=http://myurl2.rss
   c. /feed?rssFeedURL=http://myurl3.rss
3. Do invoke the same rssFeedURL and notice that that it will be served from memcache.
4. In the Admin Console, go to the Data section in the left hand navigation panel, select Memcache Viewer.
5. Review the memcache data.
   How many items are in memcache?
6. In the Memcache Viewer, flush the cache.
7. Back in your application, invoke the URL again and test it out.

Additional Resources

Developer Documentation:

- Using Memcache (Python)
- Memcache Java API Overview
- Memcache Python API Overview

Compare and Set -- Blog post about using compare and set when putting values in memcache

Effective Memcache -- article about best practices for memcache (highly recommended reading)

Best Practices for App Engine Memcache

Summary

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