Googles Groovy app engine...

Posted At : April 9, 2009 9:32 AM | Posted By : Jon Messer
Related Categories: Groovy

When Google first announced it's app engine, I thought wow that sounds cool, but I wasn't really into learning python (nothing against it, just wan't interested). So I filed it for later investigation. Well after about a year or so, later has become now. And now we don't need to learn python. We can use Java or more interesting for me Groovy.

So after seeing this but before I found this I went through the Java getting started docs, which are great, for Java. So if you want to follow along, sign up (only 10,000 developers will get this early access) install the eclipse plugin for the app engine, download the SDK somewhere (I chose /usr/local) you'll need that path later, and of course you'll need Groovy (specifically the groovy-all jar) and it wouldn't hurt to have the Groovy plugin too. Whew, that seems like a lot, and you don't actually need all of these, but if you just want to get your feet wet within the IDE this is the way. You could also do all this command line stylie, if you choose.

I would recommend that you go through the whole getting started how-to for Java (you don't have to upload it to the cloud though). I am only going to go through the differences I needed to make to integrate a little Groovy. So my goal for this is to use JPA instead of JDO and use Groovy for the domain object. Of course now that I've read this I'll probably go back and use groovlets instead of servlets and JSP, but for now...

First off, right click your project and "Add Groovy Nature" to it (how cool is that). Also locate your groovy-all-*.jar from your {groovy install path}/embeddable directory, drop this into the war/WEB-INF/lib directory of your project.

Ok now let's delete the guestbook/Greeting.java and guestbook/PMF.java files that the getting started had you write. Yes this will break your app it's OK we'll fix it, create Greeting.groovy and EMF.java files in thier place.

Greeting.groovy :

```
package guestbook

import javax.persistence.*
import com.google.appengine.api.users.User;

@Entity
public class Greeting
{
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 Long id

 public User author
 public String content
 public Date date

}
```

EMF.java

```
package guestbook;

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public final class EMF {
 private static final EntityManagerFactory emfInstance = Persistence.createEntityManagerFactory("transactions-optional");

 private EMF() {}

 public static EntityManagerFactory get() {
  return emfInstance;
 }
}
```

We have to update SignGuestbookServlet.java and GuestBook.jsp to use our JPA enable EMF instead of the JDO PMF (friggin acronyms)

SignGuestbookServlet.java (changes only):

```
...

import guestbook.EMF;
import javax.persistence.EntityManager;
```

```
...

        Greeting greeting = new Greeting();
        greeting.date = date;
        greeting.author = user;
        greeting.content = content;

        EntityManager em = EMF.get().createEntityManager();
        try {
            em.persist(greeting);
        } finally {
            em.close();
        }
...
```

GuestBook.jsp (changes only)

```
...
<%@ page import="javax.persistence.EntityManager" %>

...

<%
    EntityManager em = EMF.get().createEntityManager();
    String query = "select from " + Greeting.class.getName();
    List<Greeting> greetings = (List<Greeting>) em.createQuery(query).getResultList();
    if (greetings.isEmpty()) {
%>
<p>The guestbook has no messages.</p>
<%
    } else {
        for (Greeting g : greetings) {
            if (g.author == null) {
%>
<p>An anonymous person wrote:</p>
<%
            } else {
%>
<p><b><%= g.author.getNickname() %></b> wrote:</p>
<%
            }
%>
<blockquote><%= g.content %></blockquote>
<%
        }
    }
    em.close();
%>
...
```

(gawd JSP is ugly)

And finally you have to create a persistence.xml file and put it in war/WEB-INF/classes/META-INF/ directory.

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
        http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd" version="1.0">

    <persistence-unit name="transactions-optional">
        <provider>org.datanucleus.store.appengine.jpa.DatastorePersistenceProvider</provider>
        <properties>
            <property name="datanucleus.NontransactionalRead" value="true"/>
            <property name="datanucleus.NontransactionalWrite" value="true"/>
            <property name="datanucleus.ConnectionURL" value="appengine"/>
        </properties>
    </persistence-unit>

</persistence>
```

At this point I thought everything would just work, but sadly no. The app engine uses datanucleus to "enhance" the byte code to make it persistable, the eclipse plugin does this for you to your java classes, but not your groovy classes. :(

Luckily though, you can use ant to enhance your groovy classes. :)

Just define a build.xml in your project root and add the task definitions from the GAE docs. You'll need to run the datanucleusenhance target, you'll need to set the sdk.dir to match where you save the google SDK

Now you can run the build.xml by right clicking on it and running it as an ant build. The datanucleusenhance target will enhance your groovy class now! So your project should like something like this now :

Now when you right click the project and run as Web Application the plugin will spin up jetty and you should be able to use the guest book with your JPA enable groovy class...

After having written this, I realize it's a little more involved than I remembered, you should probably just follow the groovy how to especially since you can then use Groovlets, but you'll still need to deal with enhancing your presistable domain objects. If you want to just download what I did here is the project file.

The other exciting thing about Google doing Java on app engine is that it's at least remotly possible that we might get some CF love on the app engine...