Exploring the PayPal X Google App Engine Toolkit: Implementing Refund Requests

by Bill Day

The first article in this series on the PayPal X Google App Engine (henceforth GAE) Toolkit introduced cloud computing. It also walked you through the PayPal portions of the PicMart photo printing application example provided with the toolkit download. If you missed that first installment, please click here and read it now before proceeding through this article.

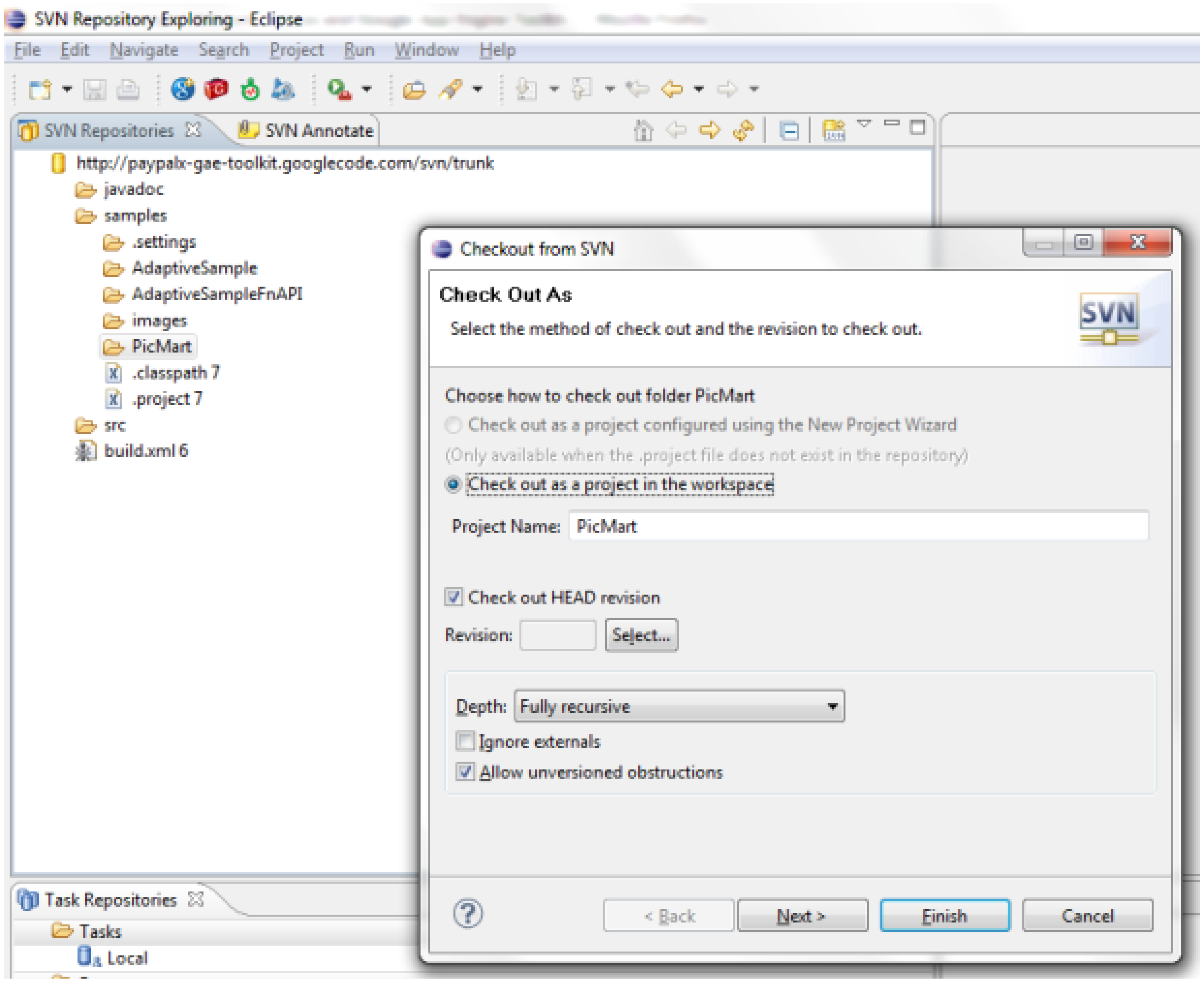
Read it? Good!

Some configuration tips before we dive into the code

Configuring everything in Eclipse, PayPal, and GAE can be a bit tricky, so let's quickly revisit how to best get everything setup before we proceed into the API and changing the PicMart code.

For this article, we really only need the PicMart portion of the GAE Toolkit project. Pulling it over SVN will automatically pick up the GAE Toolkit JARs you need and plop them into your PicMart project's lib directory. This will enable you to compile and deploy PicMart, and the modified version of it discussed below, to GAE.

If you haven't already downloaded and configured the GAE Toolkit in your Eclipse installation, simplify things for yourself by checking out just the samples/PicMart portion of the Toolkit project from http://paypalx-gae-toolkit.googlecode.com/svn/trunk over SVN as pictured below.



Checkout from SVN

Check Out As

Select the method of check out and the revision to check out.

Choose how to check out folder PicMart

( ) Check out as a project configured using the New Project Wizard

(Only available when the .project file does not exist in the repository)

(•) Check out as a project in the workspace

Project Name: PicMart

[✓] Check out HEAD revision

Revision: Select...

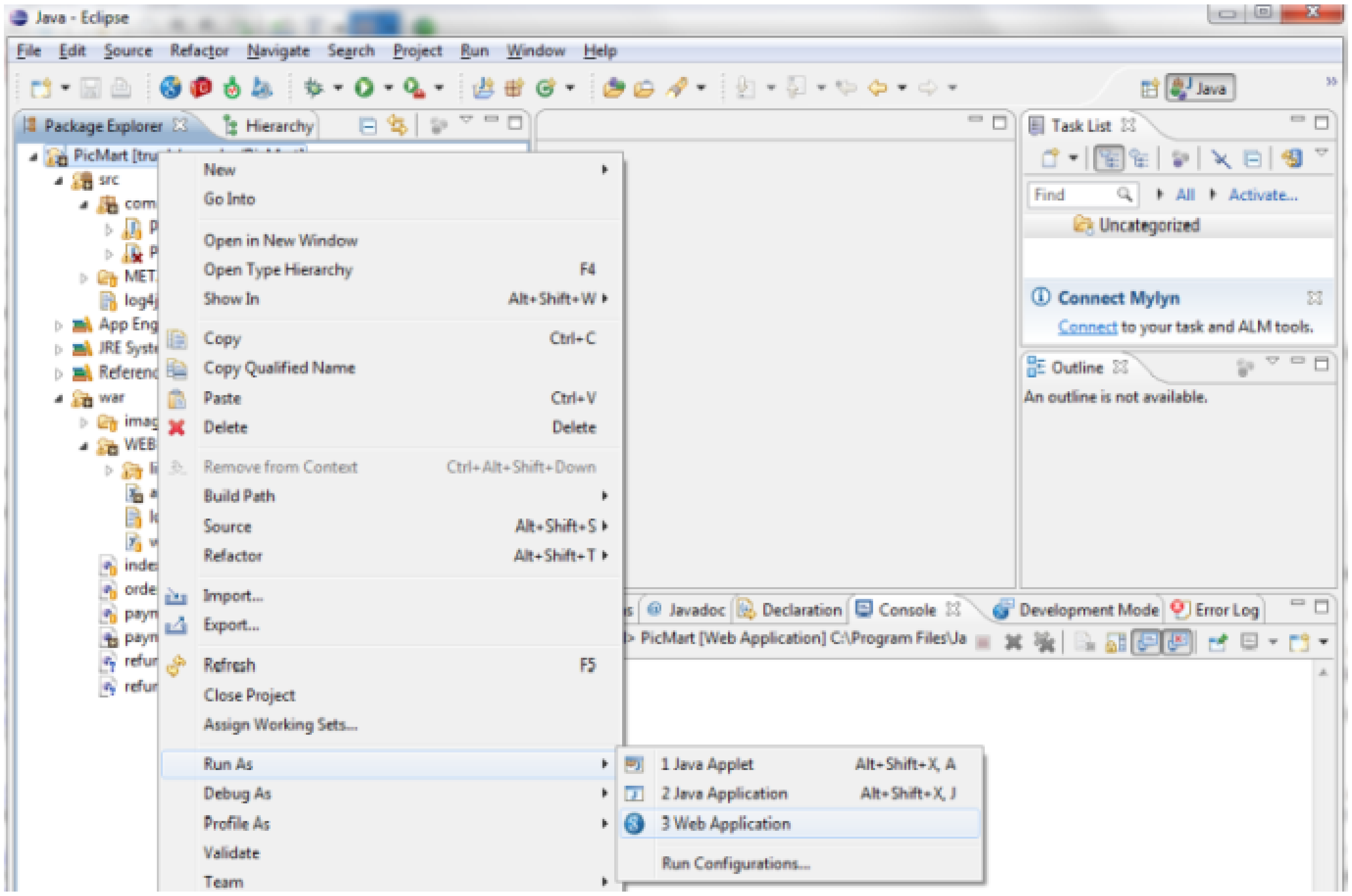
Depth: Fully recursive

[ ] Ignore externals

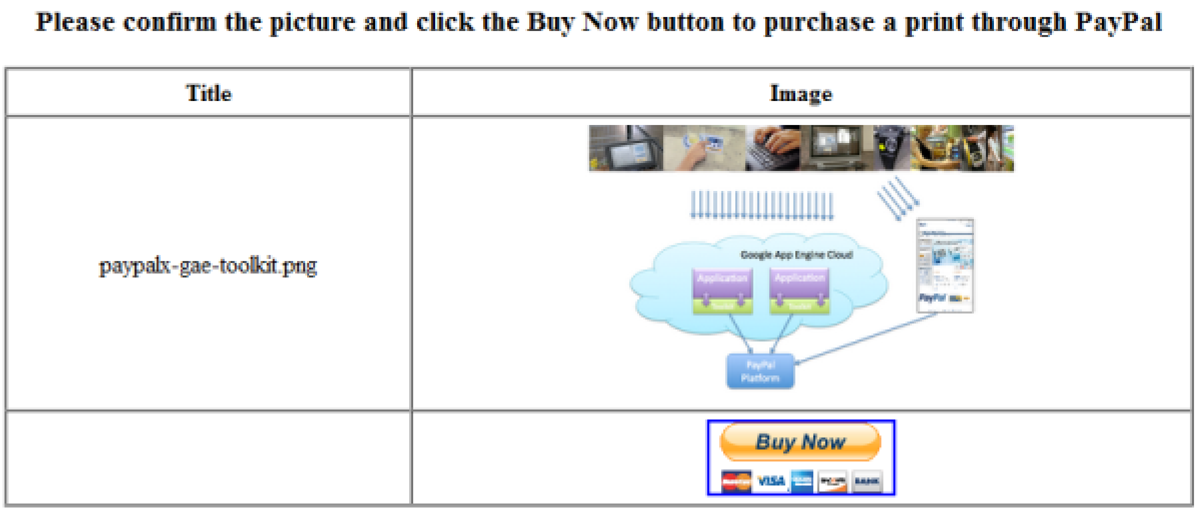
[✓] Allow unversioned obstructions

< Back Next > Finish Cancel

Once you have the PicMart project (whether by following along in the previous article or jumpstarting things as described above), you can verify your local configuration is setup correctly by test deploying on the GAE Eclipse plugin provided Jetty server. Do this by right clicking on the PicMart project in Eclipse and selecting "Run As" -> "Web Application":



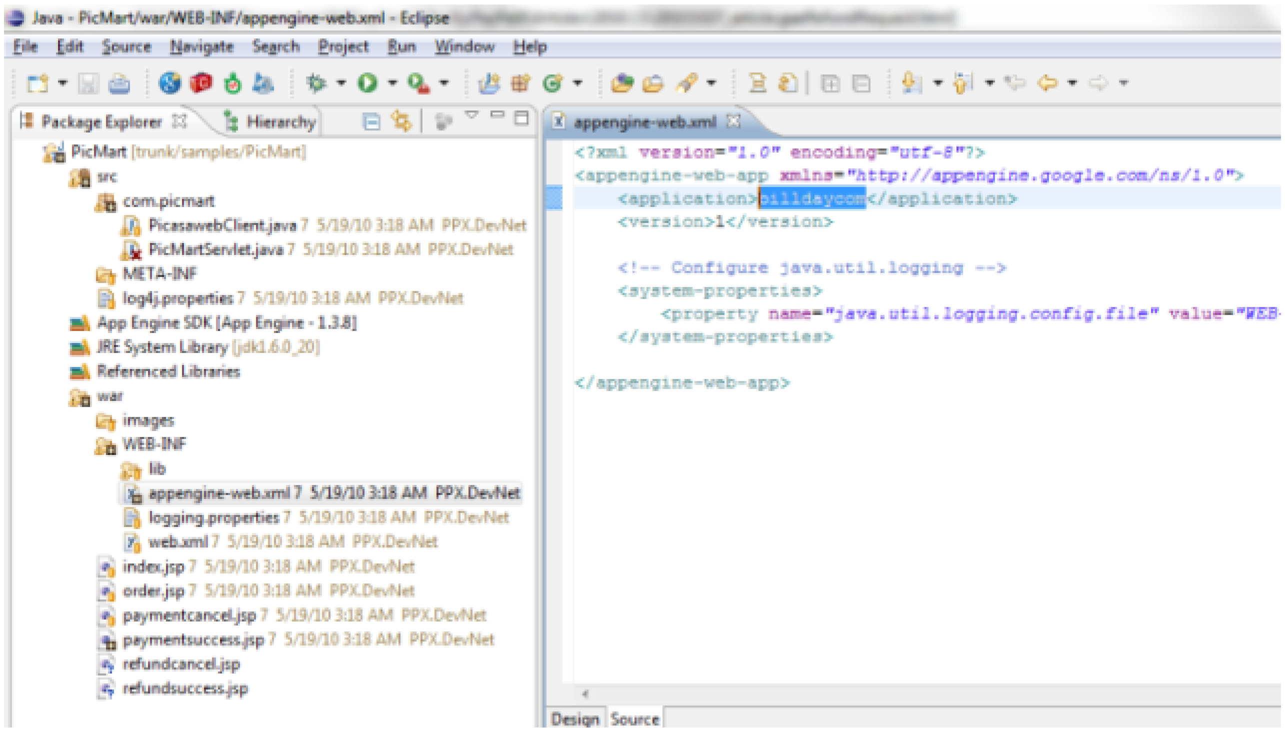
If everything works correctly, you should see INFO: The server is running at `http://localhost:8888/` in your Eclipse console and you can then connect to that URL in your browser and see the application up and running. Verify it's working by clicking on the image at the upper right; if this loads the purchase screen as shown below, the app is running.



Please confirm the picture and click the Buy Now button to purchase a print through PayPal

Title	Image
paypalx-gae-toolkit.png	
	Buy Now

Finally, there are two additional configuration actions you'll need to verify you have everything setup to modify, compile, and deploy the updated PicMart GAE app. If you don't already have a PayPal Sandbox test buyer account, you will need to create one. Click here to login to the Sandbox. You'll use this test buyer account to make test purchases in PicMart. You will also need to change the default GAE application ID for PicMart to an application ID that you created. You do this by editing `war/WEB-INF/appengine-web.xml` and changing the `picmart-app` text to your app ID, for instance in my case to `billdaycom`.


Java - PicMart/war/WEB-INF/appengine-web.xml - Eclipse
File Edit Source Navigate Search Project Run Window Help
Package Explorer
Hierarchy
appengine-web.xml
Design
Source

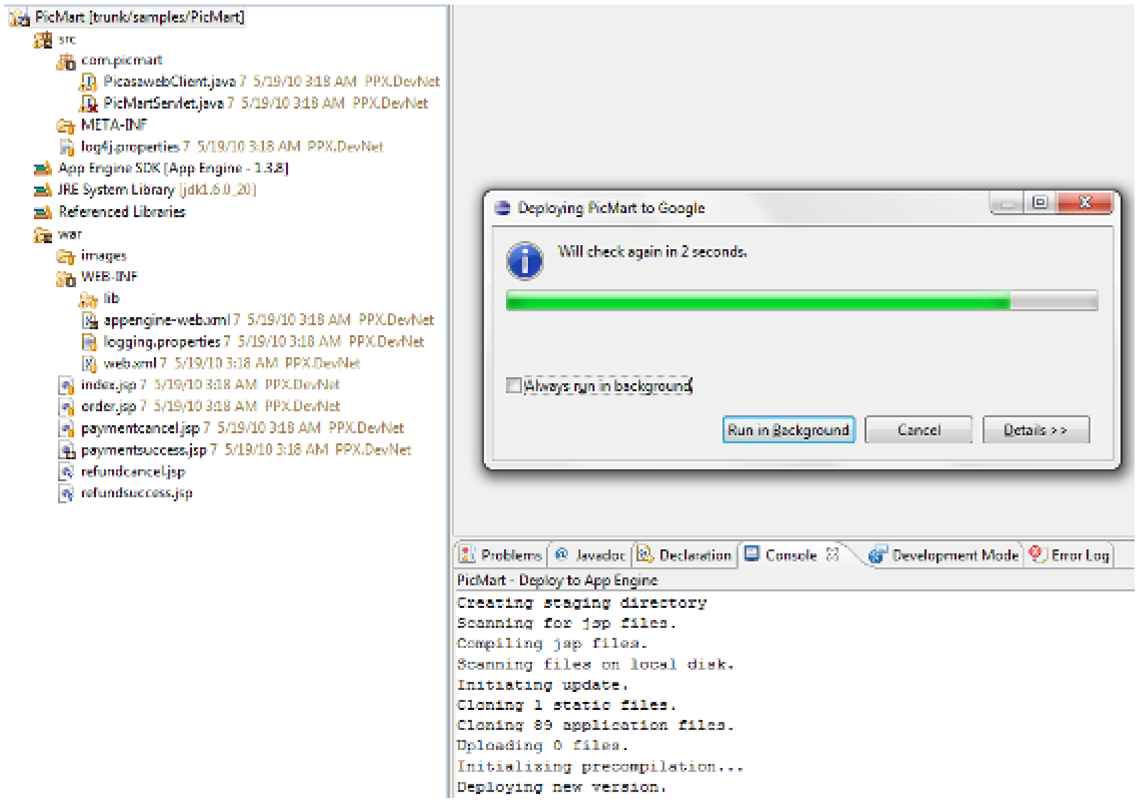
```
PicMart [trunk/samples/PicMart]
  src
    com.picmart
      PicasawebClient.java 7  5/19/10 3:18 AM  PPX.DevNet
      PicMartServlet.java 7  5/19/10 3:18 AM  PPX.DevNet
    META-INF
    log4j.properties 7  5/19/10 3:18 AM  PPX.DevNet
  App Engine SDK [App Engine - 1.3.8]
  JRE System Library [jdk1.6.0_20]
  Referenced Libraries
  war
    images
    WEB-INF
      lib
      appengine-web.xml 7  5/19/10 3:18 AM  PPX.DevNet
      logging.properties 7  5/19/10 3:18 AM  PPX.DevNet
      web.xml 7  5/19/10 3:18 AM  PPX.DevNet
    index.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    order.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    paymentcancel.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    paymentsuccess.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    refundcancel.jsp
    refundsuccess.jsp
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
    <application>billdaycom</application>
    <version>1</version>

    <!-- Configure java.util.logging -->
    <system-properties>
        <property name="java.util.logging.config.file" value="WEB-
    </system-properties>

</appengine-web-app>
```

If you've done everything correctly above, you should now be able to deploy PicMart to the Google cloud by selecting the project in Eclipse, clicking on the "Deploy App Engine Project" jet engine icon from the toolbar, and then filling in your Google account email and password in the popup. Deployment may take a minute or two, and will give you updates such as:



```
PicMart [trunk/samples/PicMart]
  src
    com.picmart
      PicasawebClient.java 7  5/19/10 3:18 AM  PPX.DevNet
      PicMartServlet.java 7  5/19/10 3:18 AM  PPX.DevNet
    META-INF
    log4j.properties 7  5/19/10 3:18 AM  PPX.DevNet
  App Engine SDK [App Engine - 1.3.8]
  JRE System Library [jdk1.6.0_20]
  Referenced Libraries
  war
    images
    WEB-INF
      lib
      appengine-web.xml 7  5/19/10 3:18 AM  PPX.DevNet
      logging.properties 7  5/19/10 3:18 AM  PPX.DevNet
      web.xml 7  5/19/10 3:18 AM  PPX.DevNet
    index.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    order.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    paymentcancel.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    paymentsuccess.jsp 7  5/19/10 3:18 AM  PPX.DevNet
    refundcancel.jsp
    refundsuccess.jsp
```

```
PicMart - Deploy to App Engine
Creating staging directory
Scanning for jsp files.
Compiling jsp files.
Scanning files on local disk.
Initiating update.
Cloning 1 static files.
Cloning 89 application files.
Uploading 0 files.
Initializing precompilation...
Deploying new version.
```

Once you see "Deployment completed successfully" in the Eclipse console, you're set! You can go to your running copy of PicMart at http://billdaycom.appspot.com/picmart, replacing billdaycom in that URL with your own app ID.

API packages and refund related classes

Now let's take a deeper look at the GAE toolkit API. The GAE Toolkit API is broken out into five Java packages:

- `com.paypal.adaptive.api.requests` - allows you to initiate payments, request and cancel preapprovals, and make refund requests in addition to getting foreign currency conversion rates.

- `com.paypal.adaptive.api.requests.fnapi` - provides functional API wrapper classes to handle simple, chained, and parallel payments, preapproved versions of each, and refund operations.
- `com.paypal.adaptive.api.responses` - this package contains the responses received after the server processes the various request operations.
- `com.paypal.adaptive.core` - contains classes supporting the other packages in the toolkit; you can jump to the pertinent classes here when you encounter a reference to them in the other packages.
- `com.paypal.adaptive.exceptions` - provides the various payment incomplete, PayPal error, and similar transaction problem exceptions that can be thrown during processing of your requests.
-

The previous GAE article showed you how PicMart makes a ParallelPay request and receives the resulting

PayResponse. But what if the customer using the application changes their mind and wants to request a full refund after they've submitted their purchase?

The requests, functional API, and responses packages provide the classes needed to make refund requests and then process responses back from the server. To request a complete refund, for example, you call `RefundCompletePayment`'s `makeRequest` method after you've set up the required parameters including API credentials (see the previous article for more on that). If all goes well, you receive back a `RefundResponse` after the request has been processed server-side. If something goes wrong during the request execution, one of several exceptions will be thrown indicating that a required parameter is missing, the request was made previously, or some other problem occurred (see the documentation for a complete list of possibilities).

Extending our example PicMartServlet to process a refund request

Let's use the refund portion of API discussed above to add logic to the PicMartServlet sample. The new code gives a consumer who's just made a PicMart purchase a chance to change their mind and cancel the purchase for a refund. Please note that the modified and new source code files are included in the (unzip this archive, replacing your existing PicMartServlet in src/com.picmart with the one in the ZIP and placing the two provided JSPs in your PicMart/war directory).

In order to enable refund processing, we'll first add a new parameter refund to the Servlet params. If refund is set to 1, then it indicates to the Servlet that it should process a refund request. This parameter needs to be declared with the new line of code below, added to the Servlet's doGetmethod, just after the opening tryblock, at the end of the other String declarations:

```
String refund = req.getParameter("refund");
```

Now add this code further down in the same try block, just before the final catch-all else:

```
} else if(refund != null && refund.equals("1")){          // process complete refund corresponding to the passed in payKey and track
```

Note that the above is based on code from the separate toolkit example AdaptiveRequests.java, modified for PicMart and its requirements, so you can refer to that file to see another instance of its use in different circumstances.

Here's how our new code works: When a consumer requests a refund (we'll see exactly where they do that in the next section), the Servlet param refund is set to 1. This causes the above code to be executed. This code requests a refund, then assuming no exceptions were thrown during the refund process, it takes the reader to a new refundsuccess.jsp page we'll discuss in a moment. Note how we pull the pay key and transaction ID of the purchase-to-be-refunded out of the Servlet parameters, and that we again need to use the credentials mentioned in the first article in this series. Also note that the code is setup as of now to run in the development sandbox; you'll need to change ServiceEnvironment to PRODUCTION to deploy an application "live" after testing and development are completed.

The glue that binds our application together

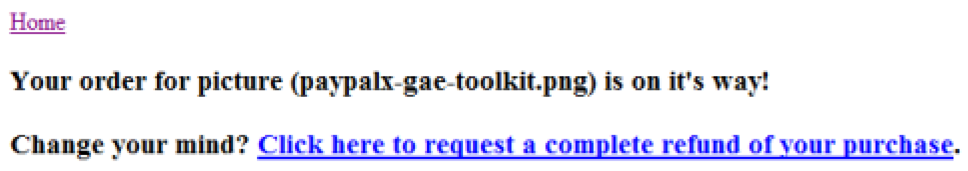
Now that we have modified our Servlet to process refund requests, we need give our customers an entry point in the application from which they can initiate refunds. We do that by changing the paymentsuccess.jsp order success output page to give the customer a refund option. Let's add the following line right before the closing BODY and HTML tags in that JSP:

```
<h3>Change your mind? <a href="/picmart?refund=1&id=<%= request.getParameter("id")
%>&payKey=<%= request.getParameter("payKey") %>&title=<%=
request.getParameter("title") %>">Click here to request a complete refund of your
purchase</a>.</h3>
```

Now when a customer makes a purchase, they'll be taken to a new success page which congratulates them but also gives them the option to change their mind and request a refund. If they click on the refund link, the Servlet is called with the refund param set to 1, which brings us into the new Servlet code shown previously and processes the refund.

Notice that the refund Servlet code in turn loads a refundsuccess.jsp page after a refund is completed. An example refund success JSP is included in .

Assuming you followed all the steps above including unzipping and placing the Servlet and JSPs into the correct locations, you can now re-select "Deploy App Engine Project", enter your password to deploy to GAE, and after deployment completes a refund-enabled PicMart should be running in the Google cloud at your app ID location. Load and work through an application purchase and you should now see the link to request a refund added to the payment success screen:

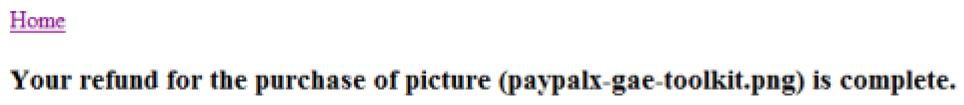


Home

Your order for picture (paypalx-gae-toolkit.png) is on it's way!

Change your mind? Click here to request a complete refund of your purchase.

Clicking the refund link above processes the refund request and takes the consumer to a refund success page:



Home

Your refund for the purchase of picture (paypalx-gae-toolkit.png) is complete.

What else can you dream up to do with the Toolkit?

We've looked at the GAE Toolkit API and how you can modify a provided photo purchase application example to add new functionality. We also discussed a number of potential configuration problems so you can avoid them and get started developing PayPal-enabled cloud applications asap.

Now it's your turn. Why not see what you can do with the PayPal X GAE Toolkit? Read through the Javadocs and get started now. Code, compile, cloud deploy, and change the world!

About the author

Bill is Founder & Technical Guru of Day Web Development, where he provides technical writing & editing, specification development, and platform & technology evangelism services. Recent projects include writing source code level documentation, articles, partner how-to materials, and blog entries for O'Reilly Media, NVIDIA, Nokia, and Digital Reasoning Systems. In addition to being an active writer and blogger for the PayPal X Developer Community, Bill is also a contributor to the GeekDad.com blog. Learn more and contact Bill via his LinkedIn profile and BillDay.com.