Base Language - Feature #3868

customer servlet integration/support

01/11/2019 10:43 AM - Greg Shah

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 01/11/2019 10:52 AM - Greg Shah

A customer has multiple OpenEdge applications which have custom written servlets that use the Java open client to call converted code. This allows them to write completely new clients and expose some interface that is easy in java but hard in OpenEdge. As part of the conversion project, these will be moved in-process in the FWD server and be hosted in the Jetty servlet container.

Some questions:

1. What can we do to make it simple to configure/load these servlets? I want to avoid a huge amount of configuration/work in order to deploy these things. I also want it to be consistent with the rest of our approach, so I guess that means directory level configuration.

2. What security implications are there? What do we need to do to ensure these are well secured and fully integrated with the FWD security model?

3. What session state support is needed such that this will integrate with converted code?

4. What can we do to make it easy to call converted code? I realize we already have StandardServer.invoke() for top-level session invocation. This may work well for a single long-running invocation. The problem is that this is too costly for an approach where there are multiple simple/short calls. We don't want to incur the session setup/takedown for every request. This is highly related to item 3 above, but there may also be a other requirements needed to make this invocation facility useful for a full range of capabilities.

Is there anything else I am missing that should be considered?

#3 - 12/03/2019 05:16 PM - Constantin Asofiei

3809e rev 11414 contains changes to add GenericWebServer APIs to load a WAR file in a standalone Jetty server, in the same JVM as the FWD server.

The problem with the approach is when the WAR file has libraries (in WEB-INF/lib) which already exist in FWD or the main converted app (in deploy/lib), but as a different version. An example of such a 'version collision' is jasper, which in FWD we have:

```
castor-core-1.3.3.jar
castor-xml-1.3.3.jar
jasperreports-6.5.1.jar
```

while a customer app (one of its WAR web apps) use an earlier version of jasper.

This version collision impacts the jasperreports_extension.properties, too. The only way I could bypass this was to remove the castor and jasper files from deploy/lib and this jasperreports_extension.properties from p2j.jar. The customer scenario is in #4436.

We need a way to force Jetty WebAppContext to use its own jasper libraries, instead of the FWD libraries; the documentation is here: https://www.eclipse.org/jetty/documentation/current/jetty-classloading.html and I think we need to add a custom class-loader for this.

#4 - 12/04/2019 01:56 AM - Hynek Cihlar

Constantin Asofiei wrote:

> We need a way to force Jetty WebAppContext to use its own jasper libraries, instead of the FWD libraries; the documentation is here: https://www.eclipse.org/jetty/documentation/current/jetty-classloading.html and I think we need to add a custom class-loader for this.

Constantin, pick a class that should load from WEB-INF/lib and trace it being loaded in WebAppClassLoader.loadClass. In particular check for any ClassNotFoundException.

#5 - 01/31/2020 11:44 AM - Greg Shah

My understanding is that what remains open in this task is the conflict with jasper and castor. Is that correct?

Hyenk: We may need to you look at this. I don't want to derail your spreadsheet widget work right now, but perhaps this should be a low priority item that you can spend a little time on over the next month?

#6 - 01/31/2020 12:21 PM - Hynek Cihlar

- Assignee set to Hynek Cihlar

Greg Shah wrote:

> Hyenk: We may need to you look at this. I don't want to derail your spreadsheet widget work right now, but perhaps this should be a low priority item that you can spend a little time on over the next month?

OK.

#7 - 03/13/2020 10:58 AM - Constantin Asofiei

Greg Shah wrote:

> 1. What can we do to make it simple to configure/load these servlets? I want to avoid a huge amount of configuration/work in order to deploy these things. I also want it to be consistent with the rest of our approach, so I guess that means directory level configuration.

This was implemented in FWD via a server init hook, which just calls GenericWebServer.initializeWebApp(webAppName, warFile, servletName) for each WAR app.

> 2. What security implications are there? What do we need to do to ensure these are well secured and fully integrated with the FWD security model?
> 3. What session state support is needed such that this will integrate with converted code?

The servlet's business logic can access the converted code via appserver calls. The servlet's threads are not setup to use the FWD server context. They are configured in web.xml via fwd.processalias, fwd.keystorefile and fwd.keystorepass to authenticate with FWD as a process account. Once authenticated, a token is saved in the servlet session, and that will be used to switch FWD context temporarily (for that servlet thread), when i.e. connecting to the appserver or executing appserver requests.

> 4. What can we do to make it easy to call converted code? I realize we already have StandardServer.invoke() for top-level session invocation. This may work well for a single long-running invocation. The problem is that this is too costly for an approach where there are multiple simple/short calls. We don't want to incur the session setup/takedown for every request. This is highly related to item 3 above, but there may also be a other requirements needed to make this invocation facility useful for a full range of capabilities.

I haven't tested calling converted 4GL code directly from the servlet, but as long as the context for the servlet session's FWD process account is properly set, it should be possible to do this (considering that the proper top-level block is setup). Some parts from FWD's multi-threading support may help (I refer that we already have a way of establishing the top-level block).

#11 - 07/03/2023 01:44 PM - Constantin Asofiei

- % Done changed from 0 to 100

- Status changed from New to Test

- Assignee changed from Hynek Cihlar to Constantin Asofiei

The fix is committed to trunk rev 14637 via 7426a.

Any customer application using a jasper version other than FWD's version needs this patch integrated in its web application; in this example, this is integrated via an existing filter in the web app:

```
   public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOExceptio
n, ServletException
   {
      ExtensionsRegistry reg = new DefaultExtensionsRegistry()
      {
         @Override
         public <T> List<T> getExtensions(Class<T> extensionType)
         {
            List<T> result = super.getExtensions(extensionType);
            Iterator<T> iter = result.iterator();
            while (iter.hasNext())
            {
               T ext = iter.next();
               if (ext instanceof FwdJasperExtensionRegistryFactory)
               {
                  iter.remove();
               }
            }
            return result;
         }

         protected List<ClassLoaderResource> loadExtensionPropertyResources()
         {
            List<ClassLoaderResource> res = super.loadExtensionPropertyResources();
            Iterator<ClassLoaderResource> iter = res.iterator();
            while (iter.hasNext())
            {
               ClassLoaderResource clr = iter.next();
               if (clr.getUrl().getFile().indexOf("p2j.jar") >= 0)
               {
                  iter.remove();
               }
            }

            return res;
         }
      };
      ExtensionsEnvironment.setSystemExtensionsRegistry(reg);
      ExtensionsEnvironment.setThreadExtensionsRegistry(reg);
```