Runtime Infrastructure - Feature #6364

implement an option to use Tomcat instead of Jetty as the embedded webserver/servlet container in the FWD server

05/13/2022 09:56 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #6373: direct Java object access to conver...	**New**

History

#1 - 05/13/2022 10:03 AM - Greg Shah

We have a customer that has extensive processes and tooling to deploy part of their application in a Tomcat container. They dislike the idea of trying to get their code working in Jetty but love the idea of deploying this part of their application inside the same JVM as the FWD server. This will allow them to optimize the connection from that code to the converted 4GL which they access via appserver today.

This task is meant to enable Tomcat usage as an option. Some objectives/requirements:

- We can focus on the FWD server. This customer doesn't use the web client, so I don't think we need to worry about the Jetty usage in the FWD client. If there is some benefit to allowing Tomcat in both places, that is OK. But I don't want to spend any significant extra time enabling it on the client.
- The converted code will not run in Tomcat. We are simply using Tomcat in place of the same things used in Jetty.
- We want all the startup/configuration/shutdown to be programmatically managed, just as we do for Jetty today. This means we will still configure things in the directory and **not in filesystem artifacts or .war files**.
- The customer wants to be able to deploy code into Tomcat on the fly. This will include replacing code that is running there (in the Tomcat container) but the converted 4GL code will not be replaced on the fly.

#2 - 05/13/2022 10:06 AM - Greg Shah

Hynek wrote:

> AFAIK Tomcat can be embedded in a Java process similarly as Jetty. Whether all the features FWD currently uses in Jetty embedded are also supported in Tomcat embedded is an unknown.

#3 - 05/13/2022 10:07 AM - Greg Shah

Constantin wrote:

> There are no servlets for SOAP, REST, WebHandler, so these parts need to be refactored to allow it - and there will not be a servlet for each resource, I think a generic servlet for each web service case, which will route the request to the proper handler.

#4 - 05/13/2022 10:08 AM - Greg Shah

> There are no servlets for SOAP, REST, WebHandler, so these parts need to be refactored to allow it - and there will not be a servlet for each resource, I think a generic servlet for each web service case, which will route the request to the proper handler.

This is required to migrate our Jetty handlers to Tomcat? Is there a generic approach we can use for common code in both environments?

#5 - 05/13/2022 10:12 AM - Greg Shah

Separately (via email) we discussed the implications of running the converted code in a Tomcat container. Doing that would bring many complications and for little or no benefit. I think it could be a real source of problems. Some points:

Constantin wrote:

> I'm not that familiar with Tomcat and how .war apps can be deployed, but assuming there is an 'application startup' hook in Tomcat and we have servlet support for the FWD web services, then it should be possible to create a (huge) .war file with the application and deploy it in a Tomcat server.
>
> But there will be some work, both on the FWD web service support and running FWD server from within Tomcat. This is what I can think about now, can't tell more what is possible or not, unless I start digging through Tomcat docs and doing some samples.

Hynek wrote:

> This may be more difficult than it sounds. Especially due all the more advanced Java patterns we do, which are typically not well tolerated in application servers (custom class loader structure, dynamic class creation, AspectJ, jar file deployment assumptions, class loading using non-context class loader, and who knows what else).

Greg wrote:

> I would like to avoid deploying converted applications as a war/running the converted code inside a servlet container. I'm more open to swapping Tomcat in for Jetty and allowing them to deploy their own (custom, not converted) servlets into Tomcat using existing processes/tools.

This task **is not** adding support for converted code running in Tomcat. I'm just storing our discussion here for future reference.

#6 - 05/13/2022 10:58 AM - Hynek Cihlar

Greg Shah wrote:

> There are no servlets for SOAP, REST, WebHandler, so these parts need to be refactored to allow it - and there will not be a servlet for each resource, I think a generic servlet for each web service case, which will route the request to the proper handler.

This is required to migrate our Jetty handlers to Tomcat? Is there a generic approach we can use for common code in both environments?

Do we actually need SOAP, REST, WebHandler? If the task is just to deploy a WAR file in the embedded Tomcat instance, then it should be enough to integrate Tomcat embedded in FWD server and provide some means of configuration by the directory.

#7 - 05/13/2022 11:34 AM - Greg Shah

I don't want to support both Jetty and Tomcat in the same server. Many customers will have REST, SOAP, WebHandler and WebSpeed requirements in addition to their own servlets.

#8 - 05/13/2022 01:34 PM - Hynek Cihlar

Greg Shah wrote:

> I don't want to support both Jetty and Tomcat in the same server.

Adding Tomcat side by side to Jetty would be a fraction of the work compared to a complete replacement of Jetty. This could save some effort.

#9 - 05/13/2022 01:47 PM - Greg Shah

Understood.

This is a long term feature as opposed to something that must be delivered quickly. I don't want to leave technical debt.

#10 - 05/17/2022 10:35 AM - Greg Shah

- Related to Feature #6373: direct Java object access to converted Java code from in-JVM non-converted Java code added

#11 - 08/24/2022 05:39 PM - Greg Shah

The customer that requested this feature is mentioned that they depend upon a peculiar behavior of Tomcat. Tomcat loads all jars in alphabetical order rather than the order they exist in the classpath or filesystem. Evidently, the jars used by this customer will fail in some way if not loaded in that sorted order. We probably need to modify MultiClassLoader to optionally provide the same behavior.