Item 262
git_comments:

1. * * Remove the transport protocol from the path and make * it starts with / * @param path * @return trimmed path

git_commits:

1. **summary:** CXF-3160 Reduce Code duplication between http transport variants
   message: CXF-3160 Reduce Code duplication between http transport variants git-svn-id: https://svn.apache.org/repos/asf/cxf/trunk@1042401 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Reduce Code duplication between http transport variants
   description: We still have to much duplicated or very similar code in the http transports. Moving header code from AbstractHttpDestination to Headers Moving invokeDestination from ServletController to AbstractServletController to share this code with http-osgi Moving invoke from ServletDestination to AbstractHttpDestination to share this code with http-osgi Removing doMessage from OsgiDestination Removing invokeDestination from OsgiServletcontroller Ignoring some mock based tests that do not work anymore. Will have to do some work on them There are some small changes in the behaviour for http-osgi as we are now using the servlet code here. It would be great if Sergey or Dan could review this. Idon´t think the differences are important but I am not sure.

jira_issues_comments:

1. **body:** Hi Christian there's definitely a lot of duplication between rt/transports/http and rt/transports/http_osgi. rt/transports/http_osgi needs to be refactored, I was actually trying to get rid of some duplications during the last couple of days, for the changes to make into 2.3.1, but I ended up seeing the demos I was working upon failing as a result so we'd need a bit more time :-) I'd probably consider even removing it altogether and may be adding an OSGI Activator to the http transport code, not sure yet how plausible it is...The biggest issue with the http_osgi is that it basically forces all the bundles to share the same context, ex, /cxf by default or say /services cheers, Sergey
   label: code-design
2. **body:** Hi Sergey, I am quite sure it is possible to at least almost remove the osgi code. I was preparing a larger refactoring first but got stuck at some point. In the other refactoring only the SpringOsgiServlet was left as it sets the context classloader which is not useful outside osgi. You said your examples were failing. Did you test with my patch or with yours? If you see some problems it would be great if you could supply me some sample code that shows them. Perhaps you could also attach your patch so I can check what ideas you pursue. What I found in any case is that ServletTransportFactory checks for the last base path that the controller processed. The OSGI http transport does not do this. Do you know why there is this difference? Btw. in the larger refactoring I tried to reuse the DestinationRegistry from the OSGI code for the other variants. I really like that it is separated from the TransportFactory. I also tried to extract the service listing into a separate class and reuse this for all variants. I would be really interested what you think about doing these two things. Still I would not like to do all these thing in once as they got me stuck. On the mailing list Dan wrote that he considered also using a servlet for the http jetty transport. I am quite sure that we could use mostly the same code for servlet, jetty and osgi. Christian
   label: code-design
3. Christian, I recently (tuesday) found out that the SpringOsgiServlet was setting the context classloader wrong anyway. Thus, that is kind of irrelevant. Basically, it was setting it to the classloader of the CXF-

bundle, not of the application. Thus, it wasn't usable to find any of hte security related things anyway. I added some code to the some of the other classes to get the right classloader, so defintely take that into account. I think in OSGi, the servlet can only be registered on a single context, so it didn't need to take into account multiple contexts. The Servlet one does. That said, it could check, it would just always match. Refactoring out the Service list and such is great. (honestly, I'd love to just move that and the wsdl generation into interceptors, but that's a different story so you could even do a "GET" type thing over JMS to get them like we handle the JAX-RS over JMS things.)

4. Hi Christian I've tried to push the updateDests method to a super class so that it can be shared between the two servlet controllers, and I nearly got there but it was late before 2.3.1 and I also thought that may be we should step back a bit and see if we can have just a single http transport but with an osgi activator acting more dynamically.... But what actually broke my demos was the minor update I did to make sure updateDests() was non-conditional on all the paths, same way as in ServletController. Not sure why, possibly because two updateDests differ a lot now between the two controllers cheers, Sergey
5. I also tried to have only one updateDests method in my larger patch and also had problems with it. The problem is that I do not know what specific problems the methods try to solve in each case so I was unsure how to proceed. In general I came to the conclusion that it is probably the best thing to do the refactoring in several steps and make sure that each step is working before doing the next. So how do we proceed with this thing? I propose that I apply my patch if you have no problems with it and we start the next refactoring from this new base. Christian
6. **body:** Moved the check for restful services to OsgiRegistry. Removed dependency of OsgiServletController to OsgiServlet to avoid circular dependency. Replaced OsgiDestination with AbstractHttpDestination where possible. This allows us to better reuse the code later.
   label: code-design
7. Hi Christian the changes look fine and reasonable - difficult to see if they can affect the existing demos. They look ok. Dan may have something else to say. You may want to try the changes later on againt some ServiceMix demos (say a soap cxf-osgi and the old cxf-jaxrs demo), by replacing the CXF bundle ServiceMix ships in (system folder) with the one containing your changes. cheers, Sergey
8. I have committed my second patch now. The tests seem to run fine. @Dan I have a question about the change in the class loader inside OsgiServletController that you did. Could you explain what classes or ressources were not reachable before and how it works now? What I also wonder is why you set the classloader for the normal services and the queryhandlers but not for the jaxrs services? I also do not yet really understand what the queryhandlers are used for. Can anyone shed some light on this? I just did some research and found the JavascriptQueryHandler and the WSDLQueryhandler. Are they the only known implementations? Anyway I wonder if we could treat the service listing like a queryhandler. I am even wondering if the destinations could also be selected in this way. Currently we are having different code for queryhandlers, service listing, soap and jaxrs services. I could imagine the qh.isRecognizedQuery(baseUri, ctxUri, ei) we use for the queryhandlers could be generally used for destinations. So all four cases could be either different kinds of destinations or queryhandlers. What do you think about this? Another thing I don´t understand is what the method udatedests is good for. I guess it saves the combination of basepath of the servlet and relative path that was configured on the endpoint. What I don´t understand though is why it is called everytime and not only when doing initializations. The method also seems to behave different in the jetty case. I guess because there can be several base paths.
9. Hi Christian I was involved a bit awhile back into the discussions surrounding query handlers. Example, CXF offers a default WSDLQueryHandler and someone can write a custom one and which will be used instead of it. But generally, the query handlers are for handling HTTP queries, those starting with '?' and thus it would make sense IMHO to keep them dealing with HTTP queries only. I would not 'overload' them. As far as service listings are concerned - please note that it is possible to configure CXFServlet to use alternative path segments for service listings, ex, "/listings" instead of '/services'. There could be a jaxrs system test verifying it, but mentioning it just in case. It would be nice to make it easy for users to customize the way the service listings are presented, such that a user handler or custom servlet extending CXFServlet can be presented with all the information needed to build a view. cheers, Sergey
10. Christian, The situation was that when using WS-Security, we couldn't use Thread.getContextClassloader().getResourse("/resources/security.properties") or similar to get the properties files that the application needed to configure the WS-Security Crypto objects. The thread context classloader was a classloader just of the CXF bundle, not the application, so the application specific resources were not available. I was just testing with a "normal" CXF/JAX-WS service so I may have only encountered that path. Dan
11. **body:** Dan, so at runtime the cxf bundle has no access to the classes of the application? I guess this is quite normal in osgi. I somehow doubt that messing with the classloader will really make things better.

Isn´t there a more osgi like way to solve this problem? At some point the application will configure the service endpoint in it´s spring config. At this point the classloader of the service provider should be available. So we could easily inject the properties into the endpoint config. Isn´t the spring osgi extender already tweaking the classloaders for us here? Btw. I have done some more refactoring that I almost had committed as the build worked nicely on my machine. Then I realized that I did not change any osgi spring configs. So they did not really work anymore. Still the build worked. So am I right that we have no system tests for osgi in the build? If I am correct then I think we should add at least some basic tests to make sure the configs are correct. Christian
label: code-design

12. Here is my next step in the refactoring. * moved the OsgiDestinationFactory to the http project as DestinationFactory and DestinatinFactroyImpl * replaced the destination management code in TransportFactories by calls to the registry * Replaced specialized destination classes by AbstractHttpDestination where possible
13. Seems I had an error in my commit. The build fails to compile. I will fix this as soon as I come home from work. Strange ... I am sure a full build worked ;-(
14. Closing the issue as he main work is done. I will open another issue as soon as I plan more refactorings