Item 264
git_comments:

1. Validating RPC/Lit, make sure we don't try a root element name thing
2. force xsi:type so types can be validated instead of trying to use the RPC/lit element names that aren't in the schema

git_commits:

1. **summary:** Merged revisions 831978 via svnmerge from
   message: Merged revisions 831978 via svnmerge from https://svn.apache.org/repos/asf/cxf/trunk ..... r831978 | dkulp | 2009-11-02 12:30:52 -0500 (Mon, 02 Nov 2009) | 1 line [CXF-1277] Enable schema validation for RPC/Lit ..... git-svn-id: https://svn.apache.org/repos/asf/cxf/branches/2.2.x-fixes@831980 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** JAXB schema validation fails for RPC
   description: If I turn on schema validation in CXF in an RPC server, it always fails. This is because JAXB tries to validate the whole input message, and there's no schema in the wsdl for these things. It seems to me that there could be schema: they are just form-unqualified elements of the elements for the messages. Is there some reason-of-standardization why we can't just include this in wsdl we generate? If so, we could fabricate it for runtime validation, I guess.

jira_issues_comments:

1. I'm not sure that is possible. With RPC, it's VERY conceivable to to have multiple operations with the same part name, but different types. Example, the default for JAX-WS: public Foo echoFoo(Foo f) public Bar echoBar(Bar b) whould result in BOTH operations having an unqualified "arg0" element, but the "type" for one op would be foo and the "type" for the other would be Bar. To do this, we'd basically have to have internally created schemas for each Message object. That might work. Not really sure though.
2. **body:** The question then is: what is the status of schema validation as a product feature. We don't document it very clearly, but it seems possible that there should be a way to turn it on that either (a) carefully does nothing for RPC services, (b) carefully validates only below the level of the parts, or (c) depends on all the schema you are discussing.
   label: documentation
3. Managed to get JAXB to do this properly. When writing, if you set the "declared type" for the JAXBElement to Object.class, it writes out an xsi:type that the validator uses so the element name is irrelevant. When reading, I wrapper the XMLStreamReader with one that fakes an xsi:type (if a type isn't on the wire) so, again, the validator can figure out what to do.