JAX-WS 2.2 MR Proposals

1 wsa:EndpointReferenceType mapping changes

JAX-WS default mapping is broken when schema constructs extend wsa:EndpointReferenceType.

For e.g.:

```
<xs:complexType name="EndpointUpdateType">
  <xs:complexContent>
    <xs:extension base="wsa:EndpointReferenceType">
      <xs:attribute name="updateType" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

public class EndpointUpdateType extends W3CEndpointReference
{...}
```

The problem is that W3CEndpointReference is a final class so the generated EndpointUpdateType code won't compile. W3CEndpointReference class is designed to be immutable as the typical web service developer need not be concerned with its contents. Hence, generating “EndpointUpdateType” class that extends W3CEndpointReference is not useful. Hence, we propose to bind wsa:EndpointReference and its subtypes to W3CEndpointReference.

2 Additional Constructor in Generated Class

The enhancement https://jax-ws.dev.java.net/issues/show_bug.cgi?id=537 asks for an additional constructor with one argument representing the WSDL location.

For e.g.:

```
public class StockQuoteService extends javax.xml.ws.Service {
    ...
    public StockQuoteService(String wsdlLocation) {
        super(wsdlLocation, new QName("http://example.com/stocks", "StockQuoteService"));
    }
    ...
}
```

This is useful to many applications where they want to create the service by passing a local WSDL, but they don't have to worry about passing in a service Qname.

3 Web Service Feature Annotations

Java EE clients need not write additional code to configure web service features like MTOM, Addressing on the injected proxies. That can be done if the feature annotations like @MTOM, @Addressing, @RespectBinding can be specified with @WebServcieRef, @WebServiceRefs annotations.

For e.g.:

```
public class MyClient {
    @Addressing
    @WebServiceRef(StockQuoteService.class)
    StockQuoteProvider stockQuoteProvider;  // proxy is configured with addressing
```

```
        ...
}
```

At present, @Addressing, @MTOM, @RespectBinding can only be applied to @Target({ElementType.TYPE}). However, @WebServiceRef can be applied to @Target({ElementType.TYPE, ElementType.METHOD, ElementType.FIELD}). The web service feature annotations @Target program elements will be changed to match @WebServiceRef @Target program elements.

4 Request/Response wrapper and exception beans optional

When a service is developed starting from java, tools like wsgen generate wrapper and exception bean classes. They also need to be bundled as part of the application. JAX-WS specification specifies these artifacts for portability, but the programming model itself doesn't use any of these classes. Hence, we propose that the applications need not package these classes and a JAX-WS runtime can take care by generating them dynamically. But if the application bundles the bean classes, then those classes must be used.

5 @XmlElement on doc/lit wrapper parameters

https://jax-ws.dev.java.net/issues/show_bug.cgi?id=496 requests a way to set "nillable" attribute on the wrapper children. Similary, there are enhancement requests to set "required" attribute on the wrapper children. @WebParam doesn't have any elements to set these. Moreover, these requirements fall in JAXB area. If doc/lit wrapper parameters are annotated with @XmlElement, then a necessary schema can be generated. Hence JAXB 2.2 allows @XmlElement be specified with the parameters. If both @WebParam, and @XmlElement are present on a parameter, @XmlElement on the parameter will be taken for the wrapper bean field annotation.

For example:

```
float getPrice(
    @WebParam(name="tickerSymbol")
    @XmlElement(name="ticker", targetNamespace="tns", nillable=true)
    String sym);

@XmlRootElement(name="getPrice", targetNamespace="...")
@XmlType(name="getPrice", targetNamespace="...")
@XmlAccessorType(AccessType.FIELD)
public class GetPrice {
    @XmlElement(name="ticker", targetNamespace="tns", nillable=true)
    public String tickerSymbol;
}

// TODO: is it illegal to specify @XmlElement on header parameters ??
```

6 Miscellaneous Changes

- Section 3.3 defines how a class implicitly defines service endpoint interface (SEI). This needs to be clarified so that only those methods that are not final or static will be considered for SEI(other requirements in 3.3 must be met to become a SEI method).
- JAX-WS doesn't support mapping of additional(which are not defined in the wsdl:portType)

headers, headerfaults that are defined in wsdl:binding. But there are two customizations in section 8.7.6 which were intended for additional headers and headerfaults. Hence, we should remove jaxws:parameter, jaxws:exception customizations from section 8.7.6

- JAX-WS runtimes override the default namespace for JAXB beans to SEI's targetNamespace during the runtime. This intended behaviour was not captured in the specification and it will be clarified. JAX-WS specification wanted elements and types to be qualified(to be in some namespace and JAXB doesn't define any namespace by default).
- Some other minor editorial updates