TODO

This document describes a list of features missing in JaxMe and suggests how to implement them. It aims to guide developers, which are new to JaxMe development and would like to start working on one of these features.

1. Wildcard attributes

Wildcard attributes are attributes not explicitly mentioned in the schema. A typical approach would be, to specify a certain namespaces attributes explicitly in the schema and leave all others unspecified. For example, XML Schema elements like "xs:schema", "xs:element", "xs:complexType", and so on, are all accepting arbitrary attributes, which are neither in the empty namespace (which is an attributes default namespace, if a prefix is absent) nor in the XML Schema namespace http://www.w3.org/2001/XMLSchema".

A wildcard attribute is specified like this:

```
        <xs:anyAttribute namespace="##other" processContents="lax"/>
```

To implement wildcard attributes in JaxMe, the following procedure is suggested:

1. Modify the class JAXBPropertySG and add a constructor taking an instance of XSWildcard as input. Add a method to the XSObjectFactory invoking the constructor. (Note that the methods signature must be distinguishable from the similar constructor required for xs:any elements.)
2. Call the constructor from within JAXBComplexTypeSG.initAttributes().
3. The generated complex types are extended with an additional Map. The keys of the Map are instances of javax.xml.namespace.QName. The values might be instances of org.xml.sax.Attribute.

   Such a Map is generated for any instance of xs:anyAttribute. This is done in JAXBComplexTypeSG.generateProperties().
4. Change the method JAXBComplexTypeSG.getXMLHandlersAddAttributeMethod() to fill the Map.
5. Change the method JAXBComplexTypeSG.generateXMLSerializersAttributes() to create SAX events filling the attributes.
6. Write a Unit test verifying your implementation.

2. xs:any elements

JAXB specifies in detail, how support for wildcard elements ought to look like. For simplicity, we omit these details here, and assume, that a wildcard element is stored as a DOM document. (Note, that this is indeed supported by the JAXB RI as well.)

Before working on the implementation, take the class build/jm/test/jaxme/src/org/apache/ws/jaxme/test/misc/address/impl/AddressTypeHandler as an example. In particular have a view at the methods startElement(String,String,String,Attributes) and endElement(String,String,String). Note that the startElement() method creates instances of JMHandler and the endElement() method processes the results by invoking the JMHandlers getResult() method.

Likewise, see the method marshalChilds() from the same class to learn how serialization works. Note in particular, how the Data object is used for namespace handling.

1. Create a generic implementation of JMHandler, which processes SAX events and creates a DOM document. This should be fairly simple by deriving a subclass from org/apache/ws/jaxme/util/DOMBuilder or using an instance of that class internally.
2. Create a generic implementation of JMXmlSerializer that takes a DOM document as input and serializes it to SAX event. Most possibly you will need to copy code from org/apache/ws/jaxme/util/DOMSerializer, derive a subclass or do similar stuff. The important difference to this class is the proper handling of namespace prefixes by means of the Data object.
3. Extend the class JAXBPropertySG by a constructor taking an instance of XSWildcard as an argument. Note, that the constructors signature must be distinguishable from the similar constructor required for wildcard attributes. Extend the XSObjectFactory with a method invoking the constructor.
4. Change the class JAXBParticleSG to invoke this constructor.
5. Change the methods extendXMLHandlersStartElementMethod() and extendXMLHandlersEndElementMethod() from JAXBGroupSG to create or process your instance of JMHandler.
6. Change the method getXMLSerializersMarshalChildsMethod() from JAXBGroupSG to serialize the wildcard elements, if present. Use your generic JMXmlSerializer to achieve that.
7. Write a Unit test that verifies your implementation.