UserTypes

Proposal

Xmlbeans provides support for types built into XML Schema. It maps these types to a specific type hierarchy as well as, for many of these types, providing convenience functions to access values as more typical Java types. For example, the `xsd:string` simple type is mapped to `XmlString` but objects with string properties will also have a convenience method to retrieve contents as a standard `java.lang.String`.

This proposal details a mechanism to allow schema code generated by Xmlbeans to provide custom convenience methods for derived simple types. The developer would provide information in the xsdconfig file indicating which simple types to generate custom convenience methods for, the Java type that will be generated, and a static handler that will convert a `org.apache.xmlbeans.SimpleValue` to the indicated custom type.

Use

Suppose a schema contains the following derived simple type:

```
<xs:schema targetNamespace="http://example.com/schema/1.0"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:example="http://example.com/schema/1.0"
        elementFormDefault="qualified">

 ...

<xs:simpleType name="uuid">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}"/>
  </xs:restriction>
</xs:simpleType>

 ...

</xs:schema>
```

Here, it would be useful to have the `uuid` derived type converted to a `java.util.UUID`. To accomplish this the xsdconfig file accompanying the schema would contain the following:

```
<xb:config xmlns:xb="http://xml.apache.org/xmlbeans/2009/03/xbean/config"
        xmlns:example="http://example.com/schema/1.0">

<xb:usertype name="example:uuid" javaname="java.util.UUID">
  <xb:staticHandler>com.example.UUIDType</xb:staticHandler>
</xb:usertype>

</xb:config>
```

The class `com.example.UUIDType` would be expected to have the following static methods:

```
/**
 * Sets the value of obj onto the given simple value target.
 */
public static void encodeUuid(java.util.UUID obj, org.apache.xmlbeans.SimpleValue target);

/**
 * Returns an appropriate Java object from the given simple value.
 */
public static java.util.UUID decodeUuid(org.apache.xmlbeans.SimpleValue obj);
```

When the `example:uuid` simple type is used as an element of a complex type, the static handler would then be invoked by a convenience method returning a `java.util.UUID`. The static handler does not throw exceptions. Any provided static handler must be able to parse any information provided to it in the `SimpleValue` class. Thus, the schema itself should limit possible inputs to those that can be validly interpreted by the static handler.

The `xget` and `xset` methods will remain available to set properties using the normally generated type extending `XmlObject`.

Generated Code

In the initial implementation, the generated code would use the Java reflect API to find and invoke the static handler. This option allows users to build the schema separately from the code that contains the static handler. However, because the use of reflection incurs a performance penalty an option to generate code that calls the static handler directly would likely be necessary for many library users.

Static Handler Methods

The expected method signature on the static handler indicated for a specific type must be public, static, and have a method name of "encode"/"decode" plus the unqualified name of the user type. The unqualified name is required to allow multiple user types to be handled by the same static handler class.

Assuming the user type is named `common:uuid` and maps to the Java type `java.util.UUID` the method signatures must be:

```
public static void encodeUuid(java.util.UUID obj, org.apache.xmlbeans.SimpleValue target)
    throws XmlValueNotSupportedException, XmlValueOutOfRangeException;

public static java.util.UUID decodeUuid(org.apache.xmlbeans.SimpleValue obj)
    throws XmlValueNotSupportedException, XmlValueOutOfRangeException;
```

Assuming the user type is `myns:foobar` and maps to the Java type `com.example.Foobar` the method signatures must be:

```
public static void encodeFoobar(com.example.Foobar obj, org.apache.xmlbeans.SimpleValue target)
    throws XmlValueNotSupportedException, XmlValueOutOfRangeException;

public static com.example.Foobar decodeFoobar(org.apache.xmlbeans.SimpleValue obj)
    throws XmlValueNotSupportedException, XmlValueOutOfRangeException;
```

As XMLBeans is designed around a "start-from-Schema" use case, the encode and decode methods must be written such that they are able to handle any value that is valid against the schema. Because validation is an optional operation when loading XML there may be cases when a decoder would be called with invalid input. In such cases either `org.apache.xmlbeans.impl.values.XmlValueNotSupportedException` or `XmlValueOutOfRangeException` should be thrown from the decode method. Such exceptions should not be thrown in cases where the XML input has been successfully validated against the schema (this is just another way of stressing that the decoder must accept all valid input according to the XML Schema). The encoder should never throw such an exception.

External Links:
Koi Web Designer jewellery