# **Castor JAXB**

# **Castor JAXB improvments**

This page will contain the documentation for revised Castor JAXB provider implementation.

#### Table of content:

- Castor JAXB improvments
- The ideas
  - General Implementation issues
  - Introduce CastorJAXBContextFactory
  - Marshaller
  - Unmarshaller
  - Future enhancement
  - Functional testing
- The implementation
  - General Implementation issues

# The ideas

All of mentioned here things are being realised through task http://jira.codehaus.org/browse/CASTOR-3160.

### **General Implementation issues**

There is quite a number of methods that will require a implementation.

#### Marshaller:

```
PP PC public < A extends XmlAdapter > A getAdapter(final Class < A > xmlAdapter)
public Node getNode(final Object node)
PP PC public void setAdapter(final XmlAdapter arg0)
PP PC public < A extends XmlAdapter > void setAdapter(final Class < A > arg0, final A
arg1)
```

#### and

PP PC public void marshal(final Object object, final XMLStreamWriter xmlStreamWriter)
(dependency on Castor 1.3.3-SNAPSHOT)
PP PC public void marshal(final Object object, final XMLEventWriter xmlEventWriter)
(dependency on Castor 1.3.3-SNAPSHOT)

and

```
public AttachmentMarshaller getAttachmentMarshaller()
public void setAttachmentMarshaller(final AttachmentMarshaller arg0)
```

#### Unmarshaller:

```
PP PC public < A extends XmlAdapter > A getAdapter(final Class < A > arg0)
public ValidationEventHandler getEventHandler()
PP PC public Schema getSchema()
public UnmarshallerHandler getUnmarshallerHandler()
PP PC public void setAdapter(final XmlAdapter xmlAdapter)
PP PC public < A extends XmlAdapter > void setAdapter(final Class < A > type, final A
xmlAdapter)
public void setEventHandler(final ValidationEventHandler validationEventHandler)
PP PC public void setSchema(final Schema schema)
```

and

```
PP PC public Object unmarshal(final XMLStreamReader xmlStreamReader) (dependency on
Castor 1.3.3-SNAPSHOT)
PP PC public Object unmarshal(final XMLEventReader xmlEventReader) (dependency on
Castor 1.3.3-SNAPSHOT)
PP PC public < T > JAXBElement < T > unmarshal(final XMLStreamReader xmlStreamReader,
final Class < T > type) (dependency on Castor 1.3.3-SNAPSHOT)
PP PC public < T > JAXBElement < T > unmarshal(final XMLEventReader xmlEventReader,
final Class < T > type) (dependency on Castor 1.3.3-SNAPSHOT)
PP PC public < T > JAXBElement < T > unmarshal(final Source node, final Class < T >
type) (dependency on Castor 1.3.3-SNAPSHOT)
PP PC public < T > JAXBElement < T > unmarshal(final Source node, final Class < T >
type) (dependency on Castor 1.3.3-SNAPSHOT)
```

and

```
public AttachmentUnmarshaller getAttachmentUnmarshaller()
public void setAttachmentUnmarshaller(final AttachmentUnmarshaller
attachmentUnmarshaller)
```

JAXBContext:

```
public Validator createValidator() throws JAXBException
public <T> Binder<T> createBinder(final Class<T> domType)
public Binder<org.w3c.dom.Node> createBinder()
public void generateSchema(final SchemaOutputResolver schemaOutputResolver)
```

where PP means patch provided and PC means patch committed.

Besides that it seams that JAXBXmINaming behaviour is incorrect for names written in pascal case for example Name is converted into lowercase name.

The JAXBXmlNaming#toXml should be revised.

### Introduce CastorJAXBContextFactory

As described in javadocs API the JAXBContext reads the jaxb.properties file existing in the given context path package and uses the class specifed in javax.xml.bind.context.factory to create the JAXBContext. The class must simply implement two methods:

```
public static JAXBContext createContext( String contextPath, ClassLoader classLoader,
Map properties ) throws JAXBException
public static JAXBContext createContext( Class[] classes, Map properties ) throws
JAXBException
```

#### Marshaller

- I think that the marshall methods need to handle the JAXBElement, in simple case just check if passed object is instance of the JAXBElement and retrieve it's value, in more complex one use the QName from the element and set that as the root element name. (Done)
- The JAXB Marshaller defines set of properties that will need to be handled. (Done current implementation supports fallowing properties: jaxb.encoding, jaxb.schemaLocation, jaxb.noNamespaceSchemaLocation, jaxb.fragment - except for the jaxb.formatted.output which does not have an equivalent. Moreover any other property will treated as internal Castor property so it is possible to modify the underlying Castor marshaller)

### Unmarshaller

- The unmarshall methods that creates the JAXBElement as a result should also correcity set the QName of the created element. (Done)
- The current implementation uses a single shared unmarshaller instance, this may not be thread safe, especially for methods that unmarshalls to JAXBElement which sets the expected class.

### **Future enhancement**

Some of the functionality could require to be actually implemented in backing Castor framework - for example handling the attachment through MTOM/XOP and swaRef.

### **Functional testing**

I think a little bit of time should be spend on functional testing, and this might get quite tedious. Looking, for example, at the @XmlAttribute annotation, there's a lot of variants to test, requiring POJOs to be annotated slightly different for each test case. That would require us to write a lot of POJOs and wire them up accordingly in the test classes. Let's see whether we can agree on how to go about this (layout, package structures, ...).

# The implementation

#### **General Implementation issues**

Marshaller:

```
public void marshal(final Object object, final XMLStreamWriter xmlStreamWriter)
public void marshal(final Object object, final XMLEventWriter xmlEventWriter)
```

Unmarshaller:

```
public Object unmarshal(final XMLStreamReader xmlStreamReader)
public Object unmarshal(final XMLEventReader xmlEventReader)
public < T > JAXBElement < T > unmarshal(final XMLStreamReader xmlStreamReader, final
Class < T > type)
public < T > JAXBElement < T > unmarshal(final XMLEventReader xmlEventReader, final
Class < T > type)
public < T > JAXBElement < T > unmarshal(final Source node, final Class < T > type)
```