



CASTOR XML SOURCE CODE GENERATOR USER DOCUMENT

Author: Arnaud Blandithor:



1	Abstract	3
1	Why a Source Code Generator?.....	4
1.1	XML Data Binding.....	4
1.2	The role of the source generator.....	5
2	Usage, options & XML Schema support.....	6
2.1	Usage	6
2.2	Source Generator Options	6
2.2.1	Command Line Options.....	7
2.2.2	Collection types:.....	8
2.2.3	Advanced options	8
2.3	XML Schema support.....	15
2.3.1	Built-in types	16
2.3.2	Structure.....	16

Structuquirement2..Tf



7/11/2001

1 Why a Source Code Generator?

1.1 XML Data Binding

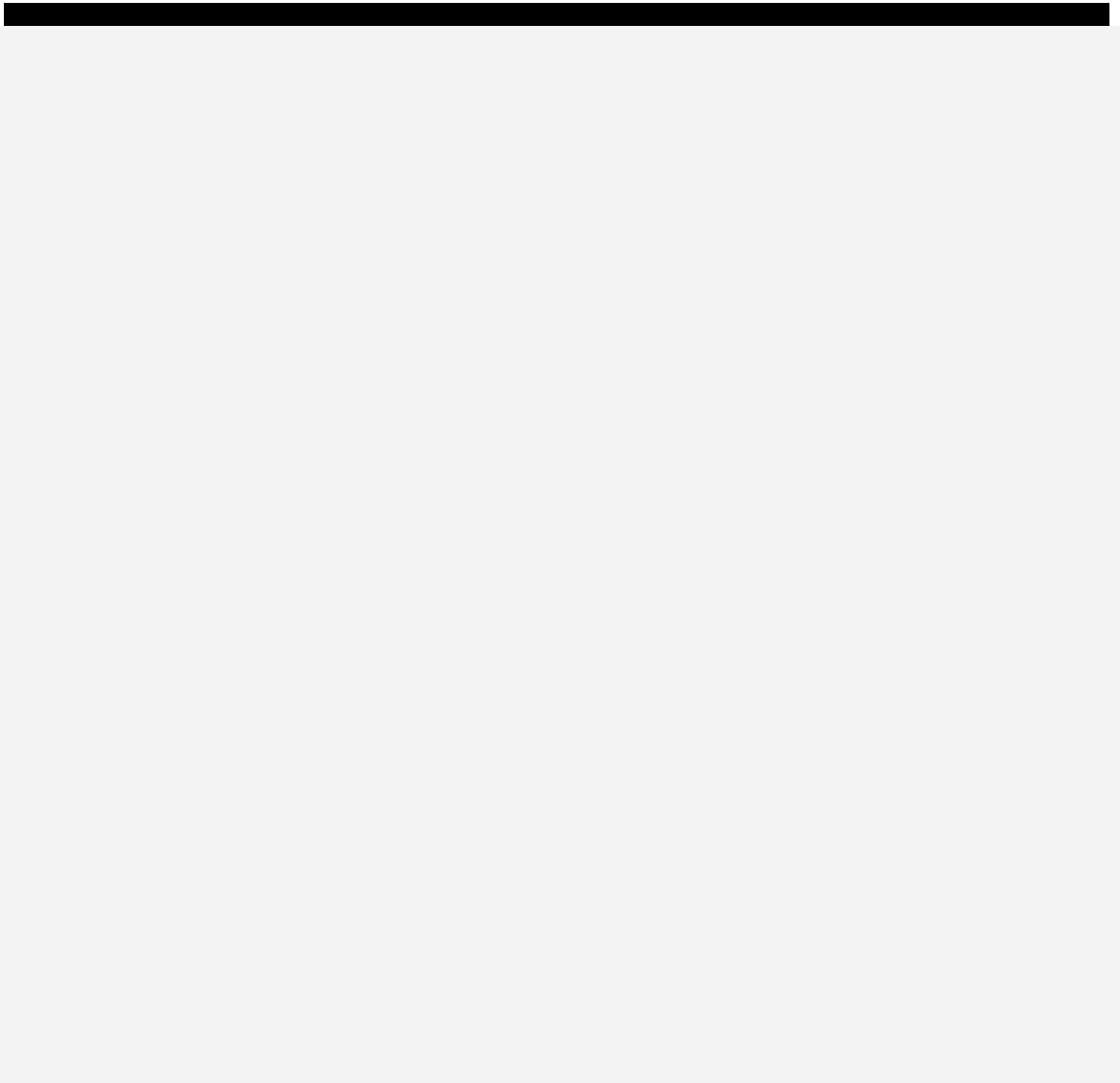
Many current applications which manipulates XML documents rely on XML Schemas which define the structure, the content and even the meaning of these XML documents.

In order to deal with the XML 'constraints' defined in the schema, applications need some tools to create and manipulate XML documents that are instanomuo pr2d mns



7/11/2001

When enabled, all properties will be treated as bound properties. Foh. h class



2.2.3.5



2.3.1 Built-in types

2.3.1.1 Primitive Datatypes

The bold



7/11/2001



XML Schema Datatypes	Facets	Java corresponding types
gDay	pattern enumeration whiteSpace max/min Exclusive max/min Inclusive	

XML Schema Datatypes	Facets	Java corresponding types
NCName	length max/min Length pattern enumeration whiteSpace	java.lang.String
ID	length max/min Length pattern enumeration whiteSpace	java.lang.String
IDREF	length max/min Length pattern enumeration whiteSpace	java.lang.Object
IDREFS	length max/min Length enumeration whiteSpace	java.util.Vector of IDREF
<i>ENTITY</i>	length max/min Length pattern enumeration whiteSpace	
<i>ENTITIES</i>		

XML Datatypes	Facets	Java
integer	totalDigits fractionDigits pattern enumeration whiteSpace max/min max/min	primitive int 2.2.3.6)
nonPositiveInteger	totalDigits fractionDigits pattern enumeration whiteSpace max/min max/min	c

XML Schema Datatypes	Facets	Java corresponding types
nonNegativeInteger	totalDigits fractionDigits pattern enumeration whiteSpace max/min Exclusive max/min Inclusive	primitive int type by default (see 2.2.3.6)

Derived datatypes



7/11/2001



7/11/2001

-



7/11/2001

- {name}
- {target namespace}
- {attribute use pairs}
- {annotation}

Unsupported:

- {attribute wildcard}

•



- § 3.8.2 XML Representation of Model Group Schema Components.

Unsupported features appear in *italics*:

```
<all
```



2.3.2.8 Schema Component: Wildcard

Wildcards are currently not supported.

- § 3.10.1 Wildcard Details

Unsupported:

- {namespace constraint}
- {process contents}
-

-



7/11/2001



7/11/2001

2.3.2.14 Conclusion – Comments

Castor can support –



7/11/2001

Note: if the 'choice' is inside a Model Group and that Model Group parent is a Model Group Definition or a complexType then the value of 'Compositor' will be only 'Choice'.

```
InvoiceType for the top-level complexType.  
InvoiceTypeChoice for the nested 'choice' inside the ComplexType.  
InvoiceTypeChoiceItem for the items inside the nested 'choice'.  
Person for the top-level group.  
PersonChoice for the nested choice.
```

Inside the class 'InvoiceTypeChoiceItem', you'll find a reference to Item1, Item2 and Item3.

2.4 Requirements



7/11/2001

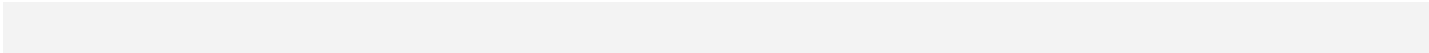
```
<?xml version="1.0"?>
```



7/11/2001



7/11/2001



```
        </xsd:complexType>
    </xsd:element>

    <!-- A U.S. Zip Code -->
    <xsd:element name="zip-code">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[0-9]{5}(-[0-9]{4})?" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>

    <!-- State Code
```



7/11/2001



3.2 The generated code

To simplify this example we now focus on the item element.



7/11/2001

```
//-----/  
// - Methods -/  
//-----/
```



```

    * @param reader
    **/

    public static test.Item unmarshal(java.io.Reader reader)
        throws
org.exolab.castor.xml.MarshalException,org.exolab.castor.xml.ValidationException
    {
        return (test.Item) Unmarshaller.unmarshal(test.Item.class, reader);
    } //-- test.Item unmarshal(java.io.Reader)

    /**
    **/

    public void validate()
        throws org.exolab.castor.xml.ValidationException
    {
        org.exolab.castor.xml.Validator. rg 90.75hiss,nullr);

```

```

    } //-- void validate((

```



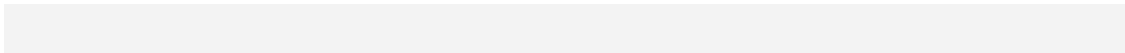

7/11/2001

```
# Defines the default XML parser to be used by castor
```

```
<element name="foo">  
  <complexType>  
    <simpleContent>
```



7/11/2001





7/11/2001



7/11/2001

Castor XML

