

JAXB Wizard and Code-Seeder Palette User's Guide

Copyright © 2009, 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Using the JAXB Wizard and Code-Seeder Palette	5
Performing a Typical JAXB Binding Process	6
▼ To Perform a Typical JAXB Binding Process	6
The JAXB Wizard	7
Code-Seeder Palette (JAXB) Wizard Editor Palette Actions	8

Using the JAXB Wizard and Code-Seeder Palette

The Java Architecture for XML Binding (JAXB) provides a fast and convenient way to bind XML schemas to Java representations, making it easy for Java developers to incorporate XML data and processing functions in Java applications. As part of this process, JAXB provides methods for constructing, unmarshalling, and marshalling XML content and data using Java language objects. This allows you to leverage the flexibility of platform-neutral XML data in Java applications without having to work with or even know XML programming techniques. The result is highly portable XML data joined with highly portable Java code that can be used to create flexible and portable applications and Web services.

When using integration solutions, data in web services is received and sent through JAX-WS or JAX-RPC. When the data is received it is converted to Java classes, which makes it easy to access when contained in the message. But not all data enters the system through web services. For example, it could enter through Java-based integration applications, where data is read from JMS or from the file system. In either case, there is a similar need to be able to access the data through Java classes. The JAXB Wizard enables you to generate Java classes from an XSD or DTD inside the IDE. The Code-Seeder Palette's functionality allows you to generate template code in order to marshal, unmarshal, construct JAXB Objects to and from many sources. Sources can be JMS Message, String, or File.

What You Need to Do

The following topics provide instructions on how access and use data to generate Java classes from an XSD or DTD inside the IDE.

- [“Performing a Typical JAXB Binding Process” on page 6](#)
- [“The JAXB Wizard” on page 7](#)

Additional Resources

Below are links to additional information, tutorials, and a screencast to help you work with the JAXB Wizard and Code-Seeder Palette:

- [Using JAXB to Process XML Payloads \(http://webcast-west.sun.com/interactive/09B01880/index.html\)](http://webcast-west.sun.com/interactive/09B01880/index.html)

This is a 14-minute screencast. It might take some time to download.

- [Binding WSDL to Java with JAXB \(http://www.netbeans.org/kb/docs/websvc/jaxb.html\)](http://www.netbeans.org/kb/docs/websvc/jaxb.html)

This is a short JAXB tutorial.

- [JAXB Sample \(http://wiki.netbeans.org/NB6JAXBSample1\)](http://wiki.netbeans.org/NB6JAXBSample1)
- [Frequently Asked Questions \(http://wiki.netbeans.org/FaqJaxb\)](http://wiki.netbeans.org/FaqJaxb)

Performing a Typical JAXB Binding Process

The JAXB APIs and tools are shipped in the taxi subdirectory of the Java WISED. This directory contains sample applications, a JAXB binding compiler (SC), and implementations of the runtime binding framework APIs contained in the `javax.xml.bind` package. For more information on binding between XML Schema and Java Classes, refer to the following tutorial: <http://download.oracle.com/javaee/5/tutorial/doc/bnazf.html>.

▼ To Perform a Typical JAXB Binding Process

The following steps demonstrate ONE particular way to bind XML schemas and Java representations. Depending on your requirements, the binding process can be achieved in several different ways.

1 Generate JAXB classes.

An XML schema is used as input to the JAXB binding compiler to generate JAXB classes based on that schema.

2 Compile JAXB classes.

All of the generated classes, source files, and application code must be compiled.

3 Unmarshal XML documents.

XML documents written according to the constraints in the source schema are unmarshalled by the JAXB binding framework.

Note – JAXB also supports the unmarshalling of XML data from sources other than files/documents, such as DOM nodes, string buffers, and SAX Sources.

4 Generate the content tree.

The unmarshalling process generates a content tree of data objects instantiated from the generated JAXB classes; this content tree represents the structure and content of the source XML documents.

5 Validate source XML (optional).

The unmarshalling process optionally involves validation of the source XML documents before generating the content tree.

Note – If you modify the content tree in Step 6 (below), you can also use the JAXB Validate operation to validate the changes before marshalling the content back to an XML document.

6 Process content.

The client application can modify the XML data represented by the Java content tree by means of interfaces generated by the binding compiler.

7 Marshal the content tree out to one or more XML output documents.

The content may be validated before marshalling.

To summarize, using JAXB involves two discrete sets of activities:

- Generate and compile JAXB classes from a source schema, and build an application that implements these classes
- Run the application to unmarshal, process, validate, and marshal XML content through the JAXB binding framework

These two steps are usually performed at separate times in two distinct phases. Typically, there is an application development phase in which JAXB classes are generated and compiled, with a binding implementation being built. That is followed by a deployment phase in which the generated JAXB classes are used to process XML content in an ongoing *live* production setting.

The JAXB Wizard

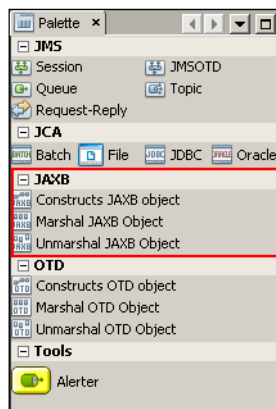
The JAXB wizard enables you to generate and compile JAXB classes for a schema, while not requiring a knowledge of the JAXB XML Binding Compiler (xjc) Tool. The wizard also helps in code completion for generated classes. JAXB code generation and compilation is made part of the project's compilation and build task. Once the JAXB Binding is created, the code seeder Palette actions will help you in using the JAXB generated classes in constructing, marshalling, and unmarshalling

Code-Seeder Palette (JAXB) Wizard Editor Palette Actions

The Code-Seeder Palette (JAXB) Wizard contains the following palette actions.

- “Constructs JAXB Object” on page 8
- “Marshal JAXB Object” on page 11
- “Unmarshal JAXB Object” on page 13

FIGURE 1 Palette Menu



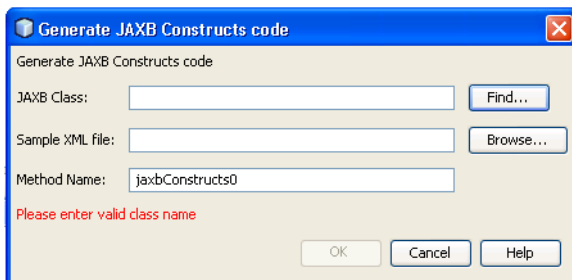
Constructs JAXB Object

The **Constructs** action generates code to instantiate JAXB objects. This code generation is useful in automating the generation of a series of getter and setter methods, especially when the JAXB object contains a series of nested objects. The code that is generated will not only instantiate the JAXB Object, but will also populate all the nested objects with initial values. This instantiation and population of values and objects can be done by passing a sample XML file. The Sample XML file needs to conform to the XSD/Schema used to generate the JAXB class. When sample XML file is not provided, all the nested JAXB members objects are instantiated.

▼ To Generate Code Using the Constructs JAXB Object

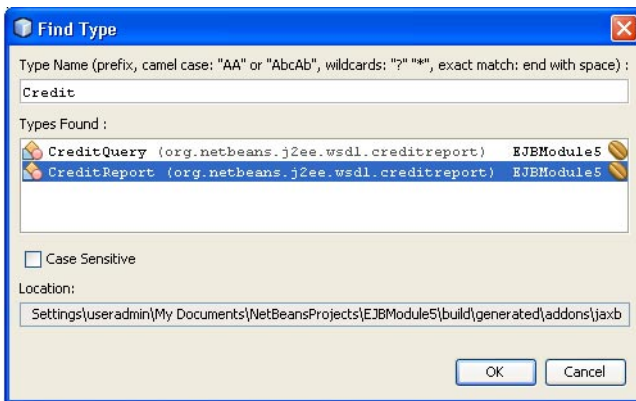
- 1 Drag and drop the Constructs JAXB Object icon into the IDE editor of an existing project. The Generate JAXB Constructs code dialog box is displayed.

FIGURE 2 JAXB Constructs Object



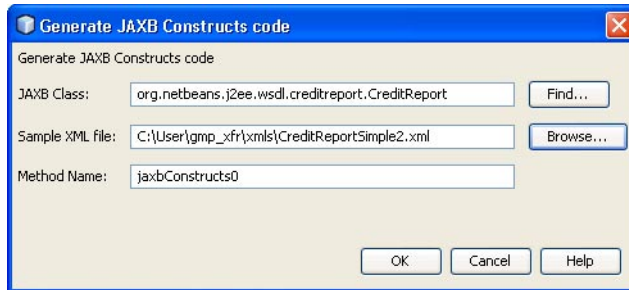
- 2 Enter the JAXB class name with the package name to be instantiated/initialized. You can also use the Find button to search for the class (as shown below).

FIGURE 3 Find Class Type



- 3 Enter a sample XML file (if available) for the initialization code to use. Update the auto generated method name if required.

FIGURE 4 Providing Sample XML File



- 4 Assuming the XML file conforms to the schema used in generating JAXB classes, a method will be created (as shown below).

FIGURE 5 Successful Method Created

```

private org.netbeans.j2ee.wsdl.creditreport.CreditReport jaxbConstructs() {
    // creditReport
    org.netbeans.j2ee.wsdl.creditreport.CreditReport creditreport =
        new org.netbeans.j2ee.wsdl.creditreport.CreditReport();

    // firstName
    creditreport.setFirstName("Joe");
    // lastName
    creditreport.setLastName("Doe");
    // dob
    creditreport.setDob("07/04/1959");
    // ssn
    creditreport.setSsn("123456789");
    // score
    creditreport.setScore("789");
    // latestAddress1
    creditreport.setLatestAddress1("800 Royal Oaks Dr.");
    // latestAddress2
    creditreport.setLatestAddress2("");
    // city
    creditreport.setCity("Monrovia");
    // state
    creditreport.setState("CA");
    // country
    creditreport.setCountry("USA");
    // postalCode
    creditreport.setPostalCode("91016");
    // liability
    creditreport.setLiability(new BigInteger("50000"));
    // liquidAssests
    creditreport.setLiquidAssests(new BigInteger("25000"));
    // immovableAssests
    creditreport.setImmovableAssests(new BigInteger("500000"));
    // currency
    creditreport.setCurrency("USD");
    return creditreport;
}

```

If the sample XML file is not provided, the code generated will not populate the default values (as shown below).

FIGURE 6 Unpopulated Method

```

private org.netbeans.j2ee.wsdl.creditreport.CreditReport JAXBConstructs1() {
    // creditReport
    org.netbeans.j2ee.wsdl.creditreport.CreditReport creditreport =
        new org.netbeans.j2ee.wsdl.creditreport.CreditReport();

    // firstName
    creditreport.setFirstName("");
    // lastName
    creditreport.setLastName("");
    // dob
    creditreport.setDob("");
    // ssn
    creditreport.setSsn("");
    // score
    creditreport.setScore("");
    // latestAddress1
    creditreport.setLatestAddress1("");
    // latestAddress2
    creditreport.setLatestAddress2("");
    // city
    creditreport.setCity("");
    // state
    creditreport.setState("");
    // country
    creditreport.setCountry("");
    // postalCode
    creditreport.setPostalCode("");
    // liability
    creditreport.setLiability(new BigInteger(""));
    // liquidAssests
    creditreport.setLiquidAssests(new BigInteger(""));
    // immovableAssests
    creditreport.setImmovableAssests(new BigInteger(""));
    // currency
    creditreport.setCurrency("");
    return creditreport;
}

```

Marshal JAXB Object

The **Marshal** palette icon action generates template code to marshal a JAXB object. This provides the ability to convert a Java object tree back into XML data. There is no difference between marshalling a content tree that is created manually using the factory methods and marshalling a content tree that is the result of an unmarshal operation. Clients can marshal a Java content tree back to XML data to a `java.io.OutputStream` or a `java.io.Writer`. You must know the Class name of the object you are trying to marshal. You can search for the class using the Find button. The template code can be generated to marshal a JAXB object to a String, OutputStream, File, Writer, or byte array.

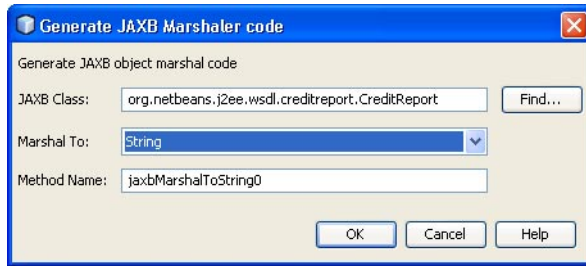
Note – The Code-Seeder currently generates code using a single package name in the context path. You may want to edit the code if more than one package name is needed in the context path.

▼ To Generate Code Using the Marshal JAXB Object

- 1 To marshal an existing JAXB class object, drag and drop the JAXB Marshal palette icon to IDE editor.

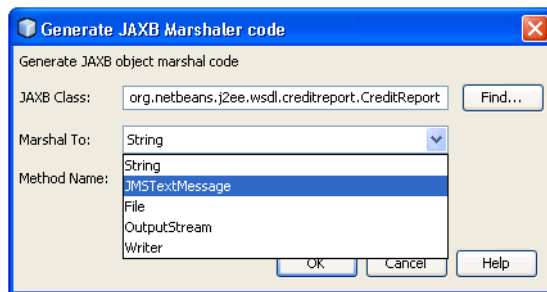
The **Generate JAXB Marshal** codedialog box is displayed.

FIGURE 7 JAXB Marshal Object



- 2 Enter the JAXB object's class name or use the Find button to search for the class. (as in the Construct's [Figure 3](#) section)
- 3 Select the Marshal To option. You can generate the code from a marshal JAXB object to one of String, Writer, OutputStream, JMSTextMessage, or File.

FIGURE 8 Select Marshal To



The desired method will be generated (as shown below).

FIGURE 9 Marshal Generated Method

```
private String jaxbMarshalToString(org.netbeans.j2ee.wsdl.creditreport.CreditReport
    jaxbObj) throws javax.xml.bind.JAXBException, java.io.IOException {
    java.io.StringWriter sw = new java.io.StringWriter();
    javax.xml.bind.JAXBContext jaxbCtx =
        javax.xml.bind.JAXBContext.newInstance(
            jaxbObj.getClass().getPackage().getName());
    javax.xml.bind.Marshaller marshaller = jaxbCtx.createMarshaller();
    marshaller.setProperty(javax.xml.bind.Marshaller.JAXB_ENCODING, "UTF-8"); //NO18N
    marshaller.setProperty(javax.xml.bind.Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
    marshaller.marshal(jaxbObj, sw);
    sw.close();
    return sw.toString();
}
```

Unmarshal JAXB Object

The *Unmarshal* palette icon action will generate template code to unmarshal a JAXB object. This allows for any global XML element declared in the schema to be unmarshalled as the root of an instance document. The JAXBContext object allows the merging of global elements across a set of schemas (listed in the contextPath). This means that a client application is able to unmarshal XML documents that are instances of any of the schemas listed in the contextPath. You must know the Class name of the object you are trying to unmarshal. Since each schema in the schema set can belong to distinct namespaces, the unification of schemas to an unmarshalling context should be namespace-independent. You can search for the class using the Find button. The template code can be generated to unmarshal a JAXB object from a String, InputStream, File, Reader, or byte array.

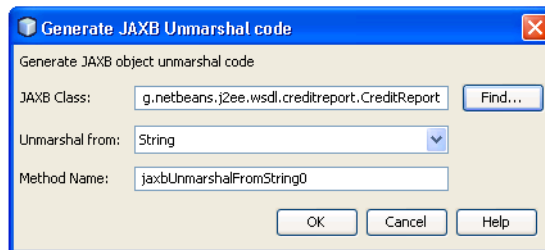
Note – The Code-Seeder currently generates code using a single package name in the context path. You may want to edit the code if more than one package name is needed in the context path.

▼ To Generate Code Using the Unmarshal JAXB Object

- 1 To unmarshal an existing JAXB class object, drag and drop the JAXB Unmarshal palette icon into the IDE editor.

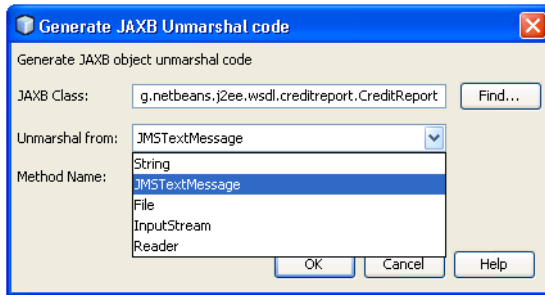
The **Generate JAXB Unmarshal code** dialog box is displayed.

FIGURE 10 Unmarshal Option



- 2 Enter the JAXB object's class name or use the Find button to search for the class.
- 3 Select the Unmarshal From option. You can generate the code to unmarshal a JAXB object from one of String, Reader, InputStream, JMSTextMessage, or File.

FIGURE 11 Select Unmarshal From



The Unmarshal Method will be generated as shown below.

FIGURE 12 Unmarshal Method

```
private org.netbeans.j2ee.wsdl.creditreport.CreditReport jaxbUnmarshalFromString(
    String str) throws javax.xml.bind.JAXBException {
    org.netbeans.j2ee.wsdl.creditreport.CreditReport ret = null;
    javax.xml.bind.JAXBContext jaxbCtx =
        javax.xml.bind.JAXBContext.newInstance(
            org.netbeans.j2ee.wsdl.creditreport.CreditReport.class);
    javax.xml.bind.Unmarshaller unmarshaller = jaxbCtx.createUnmarshaller();
    ret = (org.netbeans.j2ee.wsdl.creditreport.CreditReport)
        unmarshaller.unmarshal(new java.io.StringReader(str)); //NOI18N
    return ret;
}
```