Document Serialization Techniques

iam@jxta.org
Version 1.0
5/21/03

XML is a fine mechanism for representing object graphs that are tree-like. The JXTA StructuredDocument is a simple façade API for these types of Data Structures. This document describes a technique and API for serializing Java Objects to and from Elements in a Structured Document. From there they can be turned to/from String representations of XML Documents.

This document introduces the DocumentSerializable API that is extremely useful for creating, serializing and deserializing objects into JXTA Documents and implicitly into XML Documents.

The DocumentSerializable API

The goal of this technique is have a simple API for the serialization of simple typed fields (String, int, boolean, etc) into tagged elements as well as the nested serialization of complex fields. The technique is based upon a simple Interface, DoocumentSerializable and a utility class DocumentSerializableUtilities, both in the package jxta.net.util. documentSerializable.

This can be used to facilitate application programmers in the creation of platform independent data types that can be placed in Message Elements or serialized to files. It can also be used in the core JXTA implementations to simplify the code, increasing the likelihood of correctness.

Using The DocumentSerializable API

The DocumentSerializable Interfaces has two methods for which any implementing object can place its contents into a JXTA Element or can restore its contents from a JXTA Element.

Interface DocumentSerializable

```
void initializeFrom(Element element) throws DocumentSerializationException
void serializeTo(Element element) throws DocumentSerializationException
```

This DocumentSerializableUtilities class has methods for adding/getting typed data elements to any JXTA Document Element corresponding to standard typed data primitives (boolean, int, long, double, String.

```
public class Foo implements DocumentSerializable {
    private String name;
    private double value;

    public Foo(String name, double value) {
        this.name = name;
        this.value = value;
    }

    public Foo() { }

    public void serializeTo(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.addString(element, "name", name);
        DocumentSerializableUtilities.addDouble(element, "value", value);
    }

    public void initializeFrom(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.getString(element, "name", null);
        DocumentSerializableUtilities.getDouble(element, "getValue", 0);
    }
}
```

Similarly there are methods for adding/getting) adding/getting nested DocumentSerializable Objects. Applying this technique to classes that have a public no-arg( ) constructor provides a powerful mechanism for easily converting any tree like Data Structure to/from a JXTA Document.

Complete Example of Tree Nested Serialization Technique

```
import net.jxta.document.*;
import net.jxta.util.*;
import net.jxta.util.documentSerializable.*;

public class SerializeExample  {

    public static void main(String arg[]) {
        try {
            Foo foo = new Foo("Pi", 3.14159);
            Bar bar = new Bar("Spunky", 30, foo);

            String barAsText = DocumentSerializableUtilities.toXmlString(bar, "bar");
            System.out.println(barAsText);          // Or place it in a JXTA Message , msg.setString("bar", barAsText);

            // Now create a new Bar from the String representation of the original Bar

            Bar bar2 = (Bar) DocumentSerializableUtilities.getDocumentSerializableFromXml(barAsText, Bar.class);

            System.out.println("Got Bar from String (name = " + bar2.getName() + ")");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output

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

<!DOCTYPE bar>

<bar>
    <name>
        Spunky
    </name>
    <age>
        30
    </age>
    <foo>
        <name>
          Pi
        </name>
        <value>
          3.14159
        </value>
    </foo>
</bar>

Got Bar from String (name = Spunky)
```

```
import net.jxta.document.*;
import net.jxta.util.documentSerializable.*;

public class Bar implements DocumentSerializable {
    private String name;
    private int age;
    private Foo foo;

    public Bar(String name, int age, Foo foo) {
        this.name = name;
        this.age = age;
        this.foo = foo;
    }

    public Bar() {}

    public String getName() { return name; }
    public int getAge() { return age; }
    public Foo getFoo() { return foo; }

    public void serializeTo(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.addString(element, "name", name);
        DocumentSerializableUtilities.addInt(element, "age", age);
        DocumentSerializableUtilities.addDocumentSerializable(element, "foo", foo);
    }

    public void initializeFrom(Element element) throws DocumentSerializationException {
        name = DocumentSerializableUtilities.getString(element, "name", null);
        age = DocumentSerializableUtilities.getInt(element, "age", -1);
        foo = (Foo) DocumentSerializableUtilities.getDocumentSerializable(element, "foo", Foo.class);
    }
}
```

```
import net.jxta.document.*;
import net.jxta.util.documentSerializable.*;

public class Foo implements DocumentSerializable {
    private String name;
    private double value;

    public Foo(String name, double value) {
        this.name = name;
        this.value = value;
    }

    public Foo() { }

    public String getName() { return name; }
    public double getValue() { return value; }

    public void serializeTo(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.addString(element, "name", name);
        DocumentSerializableUtilities.addDouble(element, "value", value);
    }

    public void initializeFrom(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.getString(element, "name", null);
        DocumentSerializableUtilities.getDouble(element, "getValue", 0);
    }
}
```

An Alternative Example

The implementation of the getXXX() methods in DocumentSerializableUtilities involve a hashtable lookup.

```
public class DocumentSerializableUtilities {
    public static String getString(Element element, String tagName, String defaultValue) {
        Element childElement = getChildElement(element, tagName);          // Hastable Lookup

        if (childElement != null)
            return getString(childElement);
        else
            return defaultValue;
    }
}
```

For objects that are deserialized in tight loops where performance is a key factor, an alternative way to deserialize is:

```
package iamx.jxta.examples.serialize2;

import net.jxta.document.*;
import net.jxta.util.documentSerializable.*;
import java.util.*;

public class Bar implements DocumentSerializable {
    private String name;
    private int age;
    private Foo foo;

    public Bar(String name, int age, Foo foo) {
        this.name = name;
        this.age = age;
        this.foo = foo;
    }

    public Bar() {}

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }
    public Foo getFoo() { return foo; }
    public void setFoo(Foo foo) { this.foo = foo; }

    public void serializeTo(Element element) throws DocumentSerializationException {
        DocumentSerializableUtilities.addString(element, "name", name);
        DocumentSerializableUtilities.addInt(element, "age", age);
        DocumentSerializableUtilities.addDocumentSerializable(element, "foo", foo);
    }

    public void initializeFrom(Element element) throws DocumentSerializationException {
        for (Enumeration e=element.getChildren(); e.hasMoreElements(); ) {
            Element childElement = (TextElement) e.nextElement();
            String tagName = (String) childElement.getKey();

            if (tagName.equals("name"))
                name = DocumentSerializableUtilities.getString(childElement);
            else if (tagName.equals("age"))
                age = DocumentSerializableUtilities.getInt(childElement);
            else if (tagName.equals("foo"))
                foo = (Foo) DocumentSerializableUtilities.getDocumentSerializable(childElement, Foo.class);
        }
    }
}
```

Class DocumentSerializableUtilities

```
static void addBoolean(Element element, String tagName, boolean value)
static boolean getBoolean(Element element, String tagName, boolean defaultValue)
static boolean getBoolean(Element element)

static void addDouble(Element element, String tagName, double value)
static double getDouble(Element element, String tagName, double defaultValue)
static double getDouble(Element element)

static void addLong(Element element, String tagName, long value)
static long getLong(Element element, String tagName, long defaultValue)
static long getLong(Element element)

static void addInt(Element element, String tagName, int value)
static int getInt(Element element, String tagName, int defaultValue)
static int getInt(Element element)

static void addString(Element element, String tagName, String value)
static String getString(Element element, String tagName, String defaultValue)
static String getString(Element element)

static void addDocumentSerializable(Element element, String tagName,
                                    DocumentSerializable documentSerializable)
static DocumentSerializable getDocumentSerializable(Element element,
                                    DocumentSerializable documentSerializable)
static DocumentSerializable getDocumentSerializable(Element element,
                                                    String tagName, Class clazz)
static DocumentSerializable getDocumentSerializable(Element element, Class clazz)

static void writeAsXmlString(OutputStream out, DocumentSerializable documentSerializable)
static void writeAsXmlString(OutputStream out,
                     DocumentSerializable documentSerializable, String rootTagName)

static DocumentSerializable getDocumentSerializableFromXml(InputStream in, Class clazz)
static DocumentSerializable getDocumentSerializableFromXml(byte buf[], Class clazz)
static DocumentSerializable getDocumentSerializableFromXml(String buf, Class clazz)

static void printAsXmlString(DocumentSerializable documentSerializable)
static String toXmlString(DocumentSerializable documentSerializable)
static String toXmlString(DocumentSerializable documentSerializable, String rootTagName)

static DocumentSerializable copyDocumentSerializable(
                                    DocumentSerializable documentSerializable)

static StructuredDocument createStructuredXmlDocument(String docType,
                                    DocumentSerializable documentSerializable)
static void copyChildren(Element toElement, Element fromElement)
static Element createChildElement(Element element, String tagName)
static Element getChildElement(Element element, String tagName)
```