Generalitat de Catalunya
Centre de Telecomunicacions
i Tecnologies de la Informació

Framework Corporatiu J2EE

Servei de XML

Versió 1.0

Barcelona, 21 / febrer / 2006

Històric de modificacions

Data	Autor	Comentaris	Versió
13/01/2006	Atos Origin, sae openTrends	Versió inicial del document	1.0

Llegenda de Marcadors

Índex

1. Introducció

1.1. Propósit

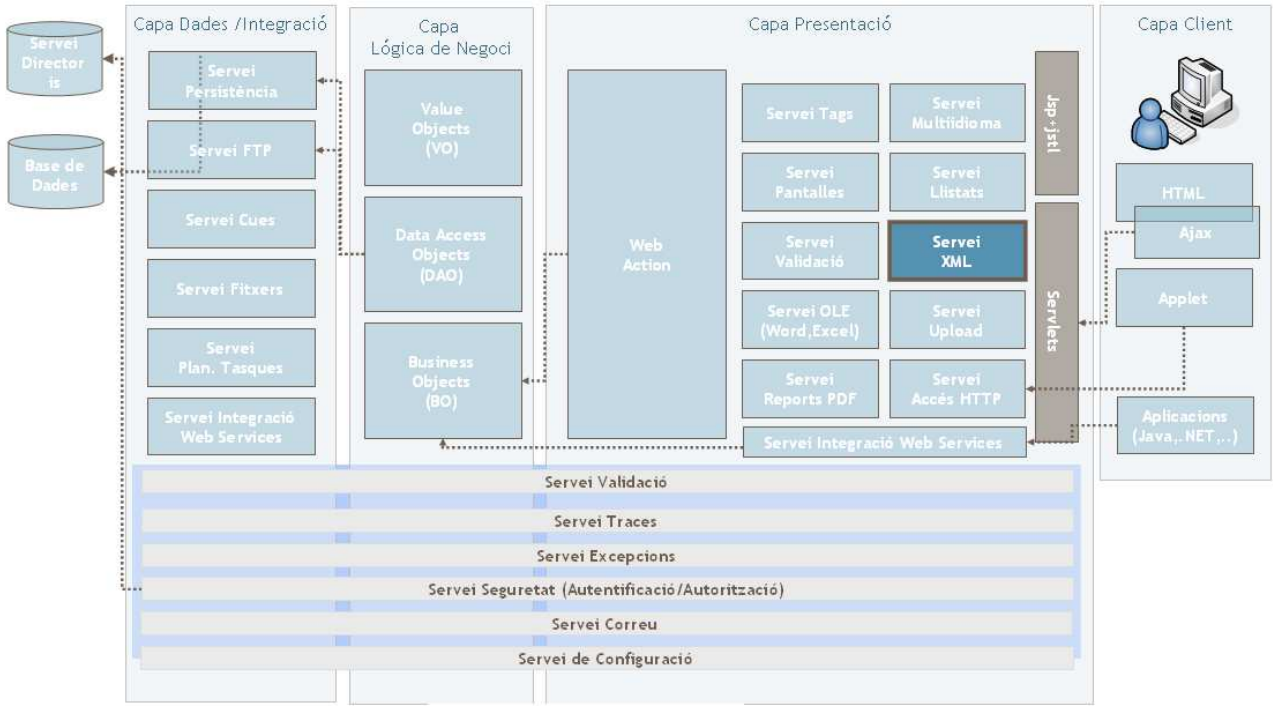
El Servei XML té 2 propósits principals:

- Serialització i deserialització d'objectes en documents XML
- Manipulació i lectura de fitxers XML

Es tracten de dos característiques diferenciades, però amb el propósit comú d'oferir la possibilitat de tractament de fitxers XML.

1.2. Context i Escenaris d'Ús

El Servei XML es troba ubicat dins els Serveis de Presentació de openFrame. L'explicació del per què s'ubica en aquesta capa és degut al seu objectiu d'integració amb clients. De la mateixa forma, la manipulació de fitxers XML i serialització/deserialització es considera un tractament necessari per transformar dades d'entrada en objectes de la lógica de negoci i viceversa.



1.3. Versions i Dependències

En el present apartat es mostren quines són les versions i dependències necessàries per fer ús del Servei.

Dins la llista de dependències es mostren diferenciades:

- Dependències bàsiques. Llibreries necessàries per fer ús del servei
- Dependències adicionals. Aquestes dependències són necessàries per poder fer ús de característiques concretes del servei o per l'ús dels tests unitaris proporcionats amb el servei

1.3.1. Dependències Bàsiques

Nom	Tipus	Versió	Descripció
commons-io	jar	1.1	http://jakarta.apache.org/commons
commons-lang	jar	2.1	
commons-logging	jar	1.0.4	
dom4j	jar	1.6.1	
log4j	jar	1.2.12	
openFrame-core	jar	1.0	
spring	jar	1.2.5	
xercesImpl	jar	2.7.1	
xmlParserAPIs	jar	2.6.2	
xml-apis	jar	1.3.02	
xmlunit	jar	1.0	
xstream	jar	1.1.2	

1.4. A qui va dirigit

Aquest document va dirigit als següents perfils:

- Programador. Per conéixer l'ús del servei
- Arquitecte. Per conéixer quins són els components i la configuració del servei
- Administrador. Per conéixer cóm configurar el servei en cadascun dels entorns en cas de necessitat

1.5. Documents i Fonts de Referència

XStream	http://xstream.codehaus.org/
Dom4j	http://www.dom4j.org

Referències generals (benchmarking,...)	• http://www.rpbourret.com/xml/XMLDataBinding.htm • The Server side. Opinion: What tool for xml binding? http://www.theserverside.com/common/printthread.tss?thread_id=30658 • Xstream: http://weblogs.java.net/blog/scottschram/archive/2005/09/the_xstream_lib.html • Performances http://www.sosnoski.com/opensrc/xmlbench/results.html • Java document model usage: http://www-128.ibm.com/developerworks/xml/library/x-injava2/index.html

1.6. Glossari

2. Descripció Detallada

2.1. Arquitectura i Components

La comunitat opensource proposa una desena d'implementacions diferents per a la serialització de XML. Actualment, i a causa del seu grau de facilitat i de *performance* openFrame es basa en la llibreria XStream. A diferència de totes les altres , aquesta llibreria no necessita cap configuració específica.

En quant a manipulació dels documents XML, usarem la llibreria dom4j, amb un rendiment millor que jdom i amb més documentació.

El servei permet la manipulació de documents XML per mitjà de l'ús de tres interfícies:

- XMLSerializationService, per a la serialització / deserialització de documents XML
- XMLSerializationConfiguration, per a configurar el mapeig entre classes Java i XML
- XMLDomHelper, per a manipular el fitxer XML (org.w3c.Document) i usar fulls d'estil XSLT

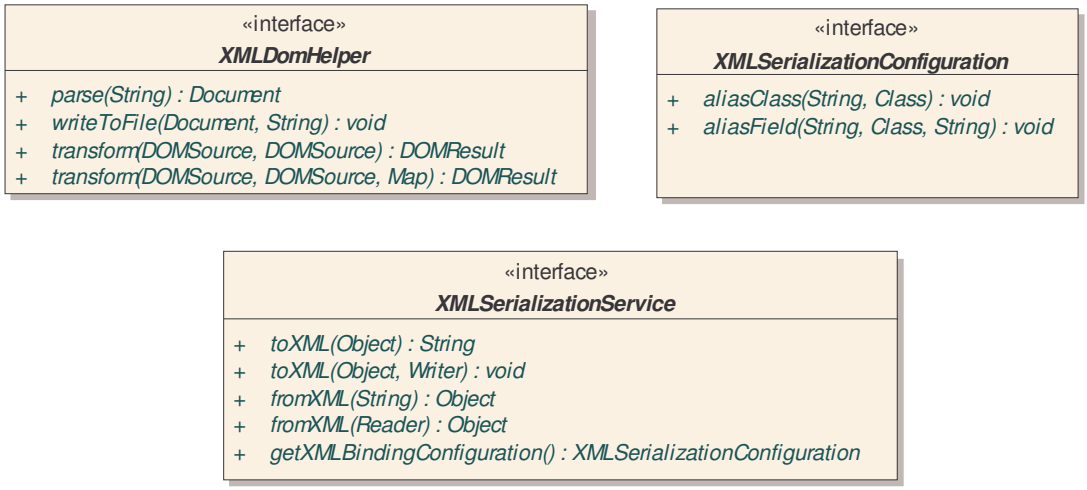
Per a més informació sobre les llibreries XML, una comparació de les APIs i un *benchmark* de les performances, veure la secció 'Documents i Fonts de Referència'

Els components podem classificar-los en:

- Interfícies i Components Genérics. Interfícies del servei i components d'ús general amb independència de la implementació escollida.
- Implementació de la serialització basada en XStream
- Implementació del parseig basada en DOM4J

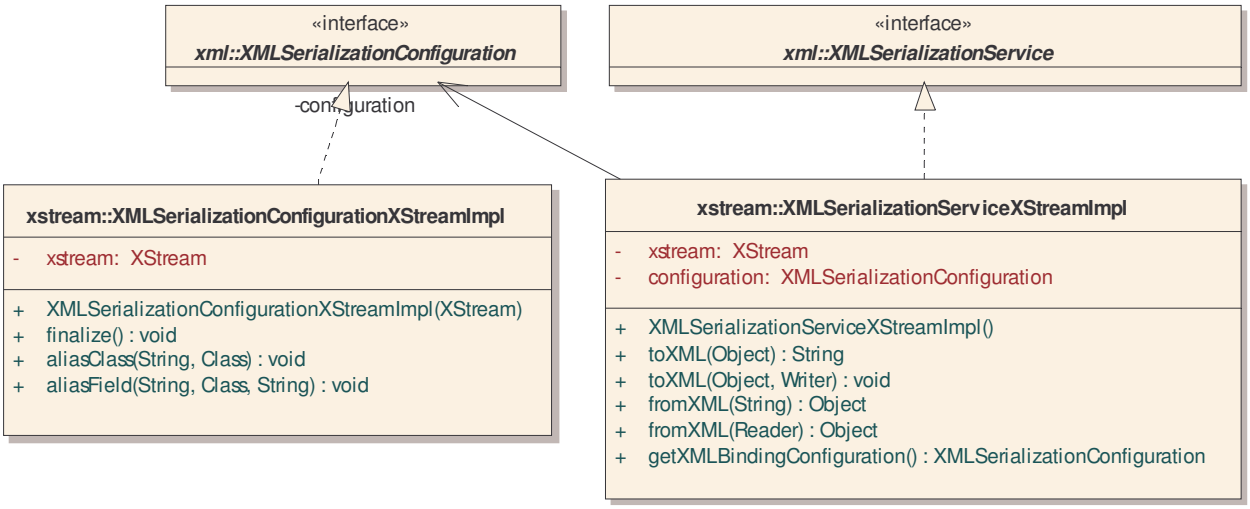
2.1.1. Interfícies i Components Genérics

Component	Package	Descripció
XMLDomHelper	net.opentrends.openframe.services.xml	Interfície de parseig de fitxers XML
XMLSerializationConfiguration	net.opentrends.openframe.services.xml	Interfície de definició de alias als objectes i camps en el tractament de la serialització/deserialització.
XMLSerializationService	net.opentrends.openframe.services.xml	Interfície de serialització d'objectes en XML
XMLSerializationServiceException	net.opentrends.openframe.services.xml	Excepció de tipus SystemException que llença el servei



2.1.2. Implementació de la Serialització basada en XStream

Component	Package	Descripció
XMLSerializationConfigurationXStreamImpl	net.opentrends.openframe.services.xml.xstream	Implementació de la configuració de la serialització basada en XStream
XMLSerializationServiceXStreamImpl	net.opentrends.openframe.services.xml.xstream	Implementació del servei de serialització basada en XStream



2.2. Instal·lació i Configuració

2.2.1. Instal·lació

La instal·lació del servei requereix de la utilització de la llibreria 'openFrame-services-xml' i les dependències indicades a l'apartat 'Introducció-Versions i Dependències'.

2.2.2. Configuració

La configuració del Servei XML implica 3 pasos:

1) Definir els servei de serialització i el mapeig dels objectes
2) Definir el servei de parseig

Definició del servei de serialització

```
<bean
id="xmlSerializationService"
       ...>
```

Fitxer de configuració: openFrame-services-xml.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

En aquesta configuració es defineix quina implementació específica s'usarà per la serialització. En l'actualitat es permet l'ús de:

- net.opentrends.openframe.services.xml.xstream.XMLSerializationServiceXStreamImpl

  Per aquesta implementació podem definir de forma adicional quin serà el mapeig entre les classes i atributs i els tags XML del fitxer.

  Per a més referència consultar la web de 'XStream'. En l'apartat d'exemples es mostra un cas pràctic de configuració d'aquest fitxer de mapeig.

Definició del servei de parseig

```
<bean id="domHelper"
       ...>
```

Fitxer de configuració: openFrame-services-xml.xml

Ubicació proposada: <PROJECT_ROOT>/src/main/resources/spring

En aquesta configuració es defineix quina implementació específica s'usarà per la manipulació de fitxers XML. En l'actualitat es permet l'ús de:

- net.opentrends.openframe.services.xml.dom4j.XMLDomHelperDom4jImpl

2.3. Utilització del Servei

La utilització del Servei es basa en l´ús de les interfícies definides a l'apartat 'Arquitectura i Components'. Consultar l'apartat 'Exemples' per veure alguns exemples del tractamen de fitxers XML amb l'API proporcionada.

A mode de resum podem resumir que:

a) La interfície XMLSerializationService proporciona principalment els mètodes següents:

   - toXML, serialització d'un objecte (JavaBean o no) cap a un document XML
   - fromXML, deserialització d'un document XML cap a un objecte Java
   - getXMLBindingConfiguration, getter de la configuració del mapeig XML

b) La interfície XMLSerializationConfiguration conté els mètodes següents:

   - aliasClass, per a fer el mapeig del nom d'una classe
   - aliasField, per a fer el mapeig d'un camp

c) La interfície XMLDomHelper conté els mètodes següents per a manipular documents DOM de la interfície org.w3c.Document:
   - parse, per a construir un document XML a partir d'un String
   - writeToFile, per a escriure el contingut d'un document XML en un fitxer
   - transform, per a realitzar transformacions XSLT amb jaxp interfícies

2.4. Eines de Suport

2.5. Integració amb Altres Serveis

2.6. Preguntes Freqüènts

3. Exemples

3.1. Exemple de Serialització basada en XStream

Per a serialitzar un objecte seguirem el següent procediment:

1) Crear les classes que volem serialitzar en XML

```
public class Person {
  private String firstname;
  private String lastname;
  private PhoneNumber phone;
  private PhoneNumber fax;
  // ... constructors and methods
}

public class PhoneNumber {
  private int code;
  private String number;
  // ... constructors and methods
}

..

Person objectToSerialize = new Person("Joe", "Walnes");
objectToSerialize.setPhone(new PhoneNumber(123, "1234-456"));
objectToSerialize.setFax(new PhoneNumber(123, "9999-999"));
```

2) Configuració del mapeig

Per defecte, Xstream usa el nom complet de la classe de l'objecte que serialitza. Per exemple, la classe com.mycompany.business.model.Person genera el XML següent:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<com.mycompany.business.model.person>
...
```

Si volem un nom més fàcil d'ús, podem afegir el mapeig següent:

```
// Mapeig de la classe Person
xmlSerializationService.getXMLBindingConfiguration().aliasClass("person",
Person.class);
```

3) Serialització d'un objecte a XML

```
String xml = xmlSerializationService.toXML(this.objectToSerialize)
```

El resultat de la serialització és el document XML següent:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<person>
  <firstname>Joe</firstname>
  <lastname>Walnes</lastname>
  <phone>
    <id>123</id>
    <number>1234-456</number>
  </phone>
  <fax>
    <id>123</id>
    <number>9999-999</number>
  </fax>
</person>
```

4) Deserialització de l'objecte des de XML

```
Person person2 = xmlSerializationService.fromXML(xml)
```

3.2. Exemple de Manipulació de Fitxers XML

Presentarem a través d'exemples concrets les APIs bàsiques de dom4j per realitzar amb poc codi les operacions següents:

1) Parsing XML
2) Ús d'Iterators
3) Navegació amb XPath
4) Manipulació de gran documents
5) Creació de document XML
6) Escriure en un fitxer
7) Conversió en un String
8) Realitzar transformacions XSLT

Per a més informació sobre dom4j, veure la pàgina http://www.dom4j.org/cookbook.html

En els exemples següents, "domHelper" referencia el bean del servei XML.

3.2.1. Parseig XML

Per a construir un document XML, amb la interfície org.**w3c**.Document:

```
String xml = "<?xml version="1.0" encoding="ISO-8859-1"?><person> ...</person>";
org.w3c.Document = domHelper.parse(xml);
```

amb la interfície org.**dom4j**.Document:

```
String xml = "<?xml version="1.0" encoding="ISO-8859-1"?><person> ...</person>";
SAXReader reader = new SAXReader(new StringReader(xm));
org.dom4j.Document document = reader.read();
```

3.2.2. Ús d'Iteradors

```
public void bar(org.dom4j.Document document) throws DocumentException {

    Element root = document.getRootElement();

    // Iteración de los elementos  de la rais del Documento
    for ( Iterator i = root.elementIterator(); i.hasNext(); ) {
        Element element = (Element) i.next();
        ...
    }

    // Iteración de los elementos  de la raizdel Documento cuyo nombre es "foo"
    for ( Iterator i = root.elementIterator( "foo" ); i.hasNext(); ) {
        Element foo = (Element) i.next();
        // do something
    }

 }
```

3.2.3. Navegació amb XPath

Podem avaluar amb XPath expressions per qualsevol element de l'arbre representat al document XML (Attribute, Element):

```
public void bar(org.dom4j.Document document) {
    List list = document.selectNodes( "//foo/bar" );

    Node node = document.selectSingleNode( "//foo/bar/author" );

    String name = node.valueOf( "@name" );
}
```

Per a més informació sobre XPath, veure el tutorial
'http://www.zvon.org/xxl/XPathTutorial/General/examples.html'

3.2.4. Manipulació de grans documents

Els grans documents no es poden manipular amb iterators perquè aquells necessiten massa memòria. En en el seu lloc usarem mètodes recursius:

```
org.dom4j.Element root = document.getRootElement();
treeWalk(root);

public void treeWalk(org.dom4j.Element element) {

    for ( int i = 0, size = element.nodeCount(); i < size; i++ ) {
        Node node = element.node(i);
        if ( node instanceof Element ) {
            treeWalk( (Element) node );
        }
        else {
            // do something....
        }
    }
}
```

3.2.5. Creació de document XML

```
org.dom4j.Document document = DocumentHelper.createDocument();
Element root = document.addElement( "root" );

Element author1 = root.addElement( "escritor" )
    .addAttribute( "name", "Jaume" )
    .addAttribute( "location", "Barcelona" )
    .addText( "Jaume Primero" );
```

3.2.6. Escriure en un fitxer

Amb la interfície org.**w3c**.Document:

```
org.w3c.Document document = ...
domHelper.writeToFile(document, “foo.xml”);
```

Amb la interfície org.**dom4j**.Document:

```
org.dom4j.Document document = ...
FileWriter out = new FileWriter( "foo.xml" );
document.write( out );
```

3.2.7. Conversió del fitxer XML en un String

```
org.dom4j.Document document = ...;
```

```
String text = document.asXML();
```

3.2.8. Realitzar transformacions XSLT

```
// Get the XML source and XSLT
DOMSource xml = new DOMSource(new File("afile.xml"));
DOMSource xslt = new DOMSource(new File("astyle.xslt"));

DOMResult result = domHelper.transform(xml, xslt);
```

4. Annexos