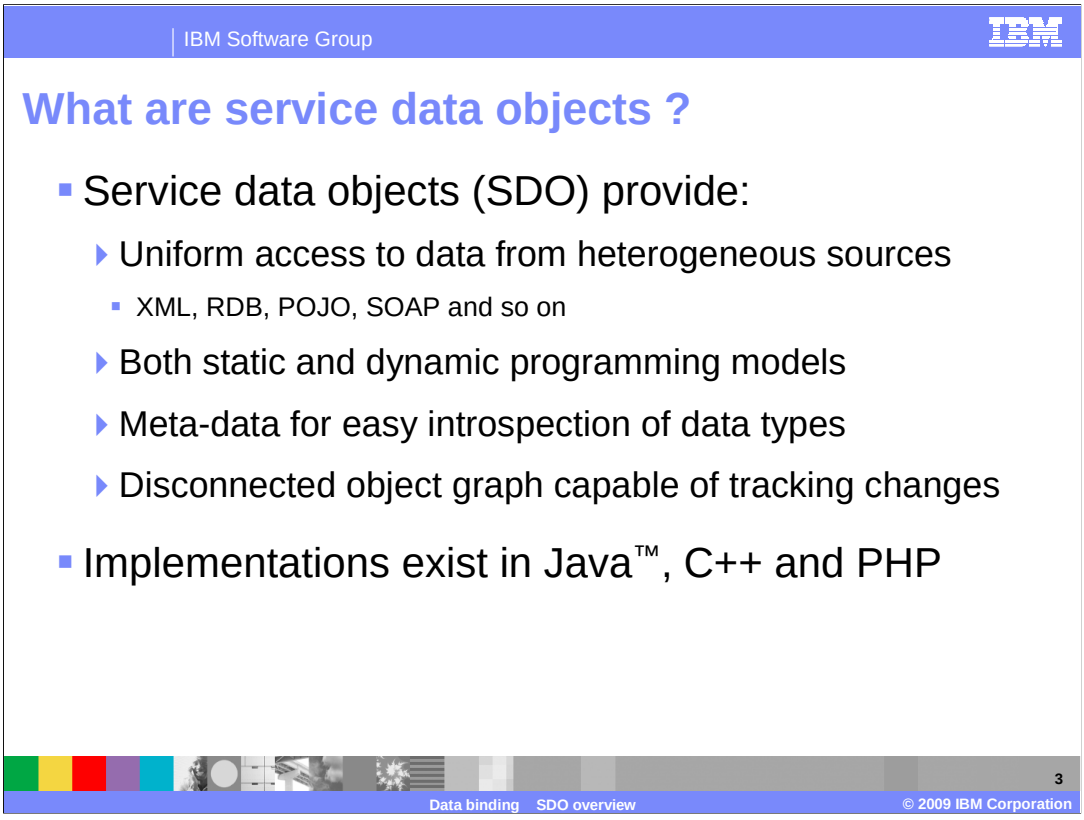
This presentation is an overview of the SDO data binding in the WebSphere® Application Server V7.0 Feature Pack for Service Component Architecture V 1.0.1.

SDO versus SDO in SCA

- Some of the capabilities of SDO are not used in any particular way when doing SCA service and client development
- Some of the features of SDO described in the first portion of the presentation (like RDB access) are
  - Described to give background context regarding SDO
  - Do not necessarily play a role in SCA feature pack development

Before covering the overview, here are some things to keep in mind. Some of the capabilities of SDO are not used in any particular way when doing SCA service and client development. Some of the features of SDO are described to give background context regarding SDO. They do not play a role in SCA feature pack development.



What are service data objects?

Service data objects, or SDOs, provide uniform access to data from heterogeneous sources such as XML, RDB, POJO, and SOAP. They also provide both static and dynamic programming models. In addition, they provide meta-data for easy introspection of data types. And they provide disconnected object graphs capable of tracking changes. SDO Implementations exist in Java, C++ and PHP.

SDO key features

- Dynamic data API (DataObject)
- XML and XML schema integration
  - Serialization can conform to pre-defined XSD
  - Or, can generate XML schema
- XPath navigation through graphs of data
- Change tracking (ChangeSummary)
- Metadata (type and property)
- Validation and constraints
- Relationship integrity

SDO key features include a dynamic data API, XML and XML Schema integration, XPath navigation through graphs of data, change tracking, Metadata, validation and constraints, and relationship integrity.

SDO overview

- Service Data Objects framework provides a unified framework for data application development
- SDO API works with data from multiple data sources:
  - Relational databases,
  - entity EJB components
  - XML pages
  - Web services

SDO provides a unified framework for data application development. With SDO, you do not need to be familiar with a technology-specific API in order to access and use data. You need to know only one API, the SDO API, which lets you work with data from multiple data sources. Data sources such as relational databases, entity EJB components, XML pages, Web services, the Java Connector Architecture, JavaServer Pages pages, and more. In a nutshell, SDO provides a dynamic programming model for writing SCA Java applications that work with schema-derived data. This dynamic programming model eliminates the need to generate Java classes from schema definitions (like in the case of JAXB), abstracts over certain XML details, and provides convenient and advanced methods for accessing data at run-time.

DataObject

- Composed of properties
- Single and many-valued properties
- Properties accessed and modified by name, offset, Property, XPath
- Can contain other DataObjects as properties
- Reverse link to containing DataObject
  - Example: DataObject parent = containedDataObject.getContainer();

The DataObject is composed of properties both single and many-valued properties. The properties are accessed and modified by name, offset, property, and XPath. DataObjects can contain other DataObjects as properties.

What reverse link to containing DataObject means is that the contained dataobject can traverse back to its containing dataobject. It is just like a DOM tree where you can find your parent node, as shown in this example.

Dynamic data API example

XSD file

```
<complexType name="Person">
  <attribute name="name" type="string"/>
  <attribute name="postalCode" type="int"/>
</complexType>
```

Java code

```
DataObject o = dataFactory.create(tns, "Person");
o.set("name", "John");
o.set("postalCode", 94133);
System.out.println(o.get("name"));
```


Here is an example of Dynamic Data API. In the top part, you see an XSD file, and on the bottom is the Java code corresponding to it. The XSD file shows the person's attributes and the corresponding Java code shows the specifics of the person.

DataObject core API

get(Property)
set(Property)

Properties by String, int, Property, XPath
- get("address")
- get(1)
- get(address)
- get("address/zip")

isSet(Property)
unset(Property)

create(Property)
delete()

The DataObject API is designed to make programming easier because it provides access to business data of all the common types and access patterns, such as name, index, and path.

The DataObject API includes several functions. Functions that:

- Get and set the properties of a DataObject.
- Query whether a Property is set.
- Create a new instance of a contained DataObject.
- Delete a DataObject from its container.
- Detach a DataObject from its container.
- Get the container of a DataObject and the containing property.
- Get the root DataObject and so forth.

For many applications that do not use generated code, the DataObject API is the only part of SDO that is used to write applications. For many applications that use generated code, the generated APIs themselves are what is used. The other parts of SDO are primarily use-as-you-go.

DataObject - typed accessors

- getXXX(property). XXX is
  - primitives: int, float, boolean, byte[], …
  - String
  - BigDecimal, BigInteger
  - Date
  - List for multi-valued properties
  - converts between primitives and Objects
  - converts between data types
- getInt("width") of 5.123 returns 5

DataObject accessor functions are separated into getters and setters for each basic type, so there is a getBoolean, getString, getInt and so on, rather than just a get() function. The notation getXXX() is used to indicate any one of these accessor functions. XXX can be primitives (like int, float, boolean, or byte[]), or they can be String, BigDecimal, BigInteger, Date, or List for multi-valued properties and so on. Also note that it converts between primitives and Objects and between data types.

Example: accessing DataObjects

Get an employee using an **SDO xpath expression** starting from the company

```
DataObject employee =
   company.getDataObject("departments[1]/employees[2]");
```

Or, an SDO xpath expression can find the employee based on simple query:

```
DataObject employee =
   company.getDataObject( "departments[number=123]/employees[SN=0002]" );
```

Or, use the API to go step by step to find the employee

```
List departments = company.getList("departments");
DataObject department = (DataObject) departments.get(0);
List employees = department.getList("employees");
DataObject employeeFromList = (DataObject) employees.get(1);
```


Here is an example of the different ways to access data using an SDO XPath expression. This is a DataObjects example on how to find an employee in a company in different ways:

By using the SDO XPath expression starting from the company, with employee as the DataObject type, you can use getDataObject method to locate them. You can also use a query or a step by step process using the getList () method to locate them as shown.

SDO meta-model

- SDO provides a simple, universal meta-model
  - Used across JavaBeans, XML, or any data source
- Meta-data classes
  - "Type"
    - Has name, URI, instance class, and properties
  - "Property"
    - Has name, type, default value, numeric index within Type

SDO provides a simple, universal meta-model which is used across JavaBeans, XML, or any data source. Metadata classes can be derived from Java, C++, UML or EMOF(Essential Meta Object Facility)

A class can be represented by an SDO Type – URI, instance class and properties. Each field of the class can be represented by an SDO property – name, type, default value, and so on.

Example - SDO metadata

```
DataObject obj = …;
Type type = obj.getType();
Collection c = type.getProperties();
Iterator i = c.iterator();
while (i.hasNext()) {
     Property prop = (Property) i.next();
     System.out.println(prop.getName());
}
```

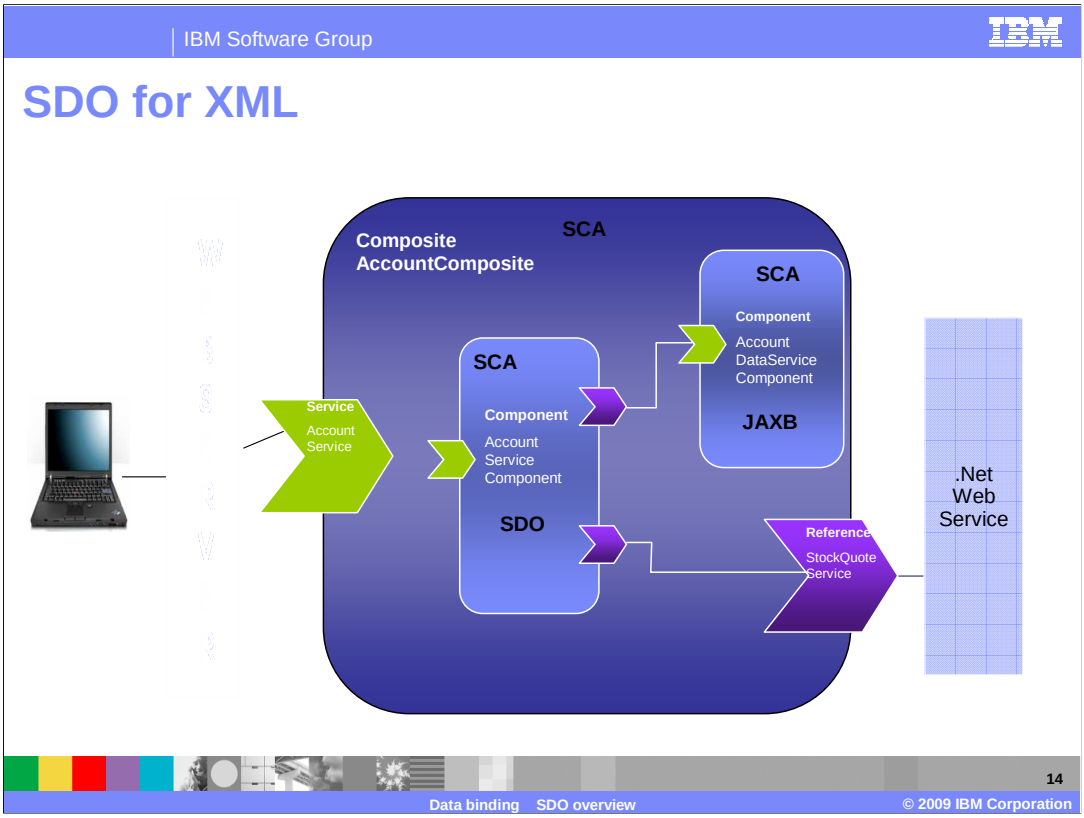

Here is an example of an SDO metadata. Many applications are coded with built-in knowledge of the shape of the data being returned. These applications know which functions to call or fields to access on the data objects they use. However, in order to enable development of generic or framework code that works with data objects, it is important to be able to introspect on data object metadata, which exposes the data model for the data objects. SDO provides APIs for metadata.

SDO today sample usecase

- SDO provides a dynamic object binding for XML

In this SDO sample use case you have the Java client and the xml artifact.

SDO provides a dynamic object binding for XML SCA WebSphere programmer, who wants to read, write, or modify XML using a dynamic object API. The XML conforms to a predefined and often vendor-specific XML schema.



Here are the SDO for XML "must haves"
XML Fidelity - API and model must support all valid XML schemas

Naturalness - API, model, and behavior must seem natural to an XML-savvy programmer

Performance - API must not inject features that prevent high-performance implementations

Tolerance - must be able to tolerate some degree of erroneous XML
Additional IBM requirements

Data virtualization support, that is, the "XML document" can not have a natural physical serialization as XML (for example., COBOL data structures)

Lazy loading and large object support

XML/XSD integration

- Direct correspondence between XML and DataObjects
- XMLHelper
  - Load and save DataObjects to XML streams
- XSD mapping to and from SDO
- XSDHelper
  - Get XML specific information - isElement, isMixed, local name, appinfo
  - Define Types and Properties from XSDs
    Annotations or XSLT for mapping control
  - Generate XSDs from Types and Properties

There is a direct correspondence between XML and DataObjects. An XMLHelper converts XML streams to and from graphs of DataObjects. It loads and saves DataObjects to XML streams. XSD mapping to and from SDO is also possible. Note that Because an XSD contains more information than type and property, there are many XSD capabilities unused by the default generation, like the preference between serializing with XML elements or attributes. The recommended procedure is to generate the XSD from types and properties, customize the XSD using tools or with XSLT, and use the customized XSD as the original from which to define the SDO types and properties.

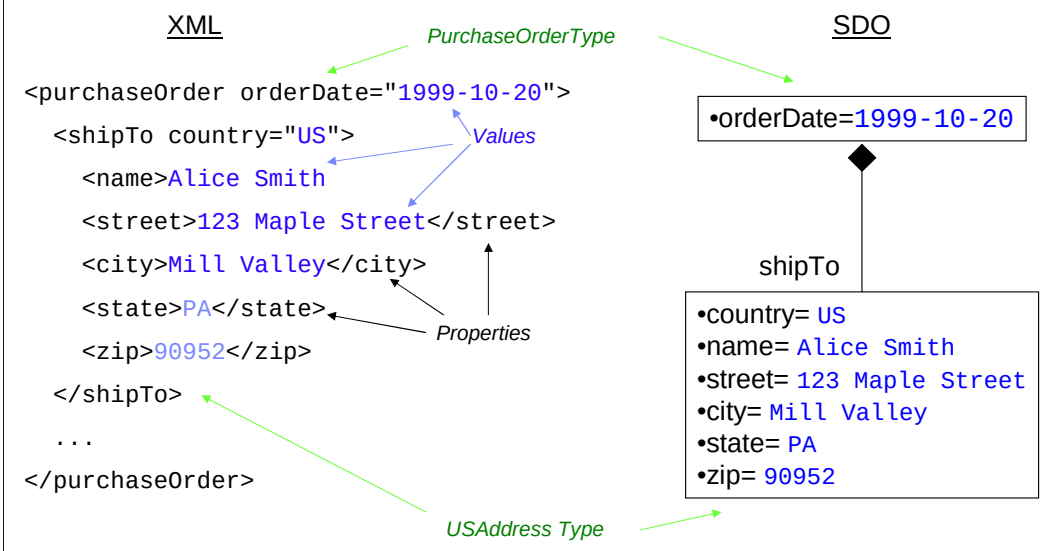
An XSDHelper provides additional information when a type or property is defined by an XML schema. XSDHelper is useful in getting XML specific information, defining types and properties from XSDs, and generating XSDs from types and properties.

XSD mapping <-> SDO

XML schema concept	SDO concept	Java concept
Schema	URI for types	Package
Simple type	Type, dataType=true	int, String, BigDecimal
Complex type	Type, dataType=false	Interface
Attribute	Property	getX( ), setX( )
Element	Property	getX( ), setX( )

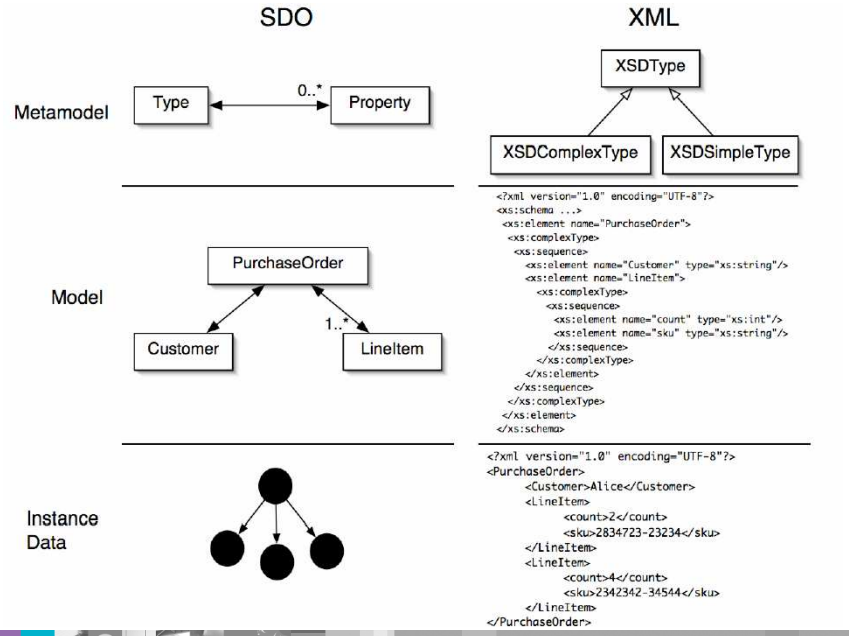
This slide shows you the XSD/SDO mapping. For example, the equivalent of a Schema in XML is a URI for types on SDO, and an XML attribute is equivalent to a property in SDO.

XML / SDO mapping



Here is an example of XML / SDO mapping of a purchase order. Take note of the orderdate, values and properties and how they map from XML to SDO.

Instance and model



This slide shows you the instance and model and how SDO/XML mapping takes place. To highlight a few things, in the SDO metamodel you have type/property. On the XML side of things you have XSDtype which can be complex or simple. Similarly for model, on the SDO side you have a purchase order with the “Customer” and the lineItem, and on the XML side you have the XSD containing the elements and the attributes.

SDO 2.1.1 and JSR235 compliancy support

- Compliant to JSR235 specification
  - http://www.ibm.com/developerworks/java/library/specification/j-jsr235/
- Key features:
  - XML and XML schema integration
  - Dynamic data API (access data without generating Java codes)
  - SDO metadata (allow users to introspect the data model of the data)
  - Generate XML schema
  - XPath navigation through graphs of data

The Feature Pack for Service Component Architecture (SCA) implementation complies with Service Data Objects (SDO) specification version 2.1.1, also known as JSR 235.

The JSR-235 defines the Service Data Object API, which is designed to simplify and unify the way in which applications handle data in a heterogeneous environment. The SDO API frees developers from handling the complexity of data programming and enables them to remain focused on business application development. Key features shown here are also listed on Slide 4 of this presentation.

Section

Summary

In summary…

SDO summary

- SDO is a framework for data application development, which includes an architecture and API:
  - Simplifies the Java EE data programming model
  - Unifies data application development
  - Supports and integrates XML
  - Incorporates Java EE patterns and best practices

SDO is a framework for data application development, which includes an architecture and API. SDO simplifies the Java EE data programming model and unifies data application development. It also supports and integrates XML and incorporates Java EE patterns and best practices.

References

- IBM Education Assistant:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/wasfpsca/1.0/JAXB.html

- Introduction to service data objects
  http://www.ibm.com/developerworks/java/library/j-sdo/
- SDO 2.1 specification:
  http://www.osoa.org/display/Main/Service+Data+Objects+Home
- SDO JSR235 Data objects
  http://www.ibm.com/developerworks/java/library/specification/j-jsr235/
- SCA feature pack information center

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.soafep.multiplatform.doc/info/welcome_nd.html

Here are some reference links for more information.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA101_SDO_Overview.ppt

This module is also available in PDF format at: ../WASv7SCA101_SDO_Overview.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

EJB, Java, JavaServer, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.