SDO: Service Data Objects

Stephen A. Brodsky, Ph.D.
IBM
Collaboration partners: BEA, IBM, Oracle, SAP, Xcalia
Agenda

- SDO – The Big Picture
- SDO – Key Concepts and Core APIs
- SDO in Action
- Summary
Information as a Service

Virtualizing the IT Infrastructure

Business Processes

SDO

Monitoring Workflows

Application Adapters

Information Services

SDO

Analysis Quality Integration

Data & Content

Rich client SDO

Business Applications

SDO

Portals

Composite Applications

Workflows

Browser

Mobile Clients

SDO

SDO
SDO Is Used Everywhere

- **Service Oriented Architecture (SOA)**
  - SDOs are the input and output of services
- **Web Services**
  - SDOs represent the XML on the wire
- **XML**
  - When XML-enabling an application
  - When accessing XML files/documents/resources/messages
- **EJB**
  - SDOs are Data Transfer Objects (DTO), value objects
  - J2EE Design Pattern
- **Enterprise Service Bus (ESB)**
  - SDOs are the input and output of services
- **Model Driven Architecture (MDA)**
  - SDO model (Type and Property) defined by Unified Modeling Language (UML) Classes and Components
  - SDO applications follow UML Sequence, Flow, State, and Collaboration
- **Data access**
  - Access relational, XML, EJB, JDO, Hibernate data sources
  - SDOs are the DTOs
- **Messaging**
  - SDOs represent the messages
- **Connectors/Adapters (EIS, CICS)**
  - SDOs represent the data records
- **BPEL-J**
  - SDOs are the Java business objects
- **ADO.NET**
  - DataSet is a subset of SDO Data Graphs
- **Cross-language programming model**
  - Complete applications may span tiers, languages
- **Java**
  - SDOs are smart POJOs with POJO interfaces
Programming Data Services

- Object/Java API that spans types of data to provide
  - Object(value)-based **read-modify-write**
  - Complex data structures (not just rows)
  - Disconnected access pattern
  - Optimistic concurrency control model
  - Tools for data binding
- Industry Support
  - Joint specification: BEA, IBM, Oracle, SAP, Sybase and Xcalia
  - Open source implementations (Apache, Eclipse)
  - Products
SOA (SCA) is the component model
- Components may be wired together
- SDO DataObjects are the data flowing on wires between Components
Heterogeneous Data Access

Client

SDO

Data Access Service

RDB

XML DB

JDBC

XPath / XQuery

Local

XML/HTTP

CCI / Proprietary

EJB: Customer

Web service

JCA
Usage Patterns

- Web apps are semi-connected, optimistic-concurrency based apps
  - Open DB connection
  - Retrieve data from DB
  - Close DB connection
  - Generate HTML; ...
  - Receive response (e.g. HTML form post)
  - Open DB connection
  - Perform update to DB
  - Close DB connection

- Standard patterns for optimistic collision detection, pagination, sorting, etc.
SDO Data Transfer Objects (DTO’s)

Data Access Service

Client

Data Graph

Data Object

Change Summary

RDB

XML DB

EJB: Customer

Web service

JCA

JDBC

XPath / XQuery

Local

XML/HTTP

CCI / Proprietary
## Comparison of Data APIs

<table>
<thead>
<tr>
<th>Model</th>
<th>API</th>
<th>Data Source</th>
<th>MetaData API</th>
<th>Query Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO</td>
<td>Disconnected</td>
<td>Both</td>
<td>Any</td>
<td>SDO Metadata API, Java Introspection</td>
</tr>
<tr>
<td>JDBC Rowset</td>
<td>Connected</td>
<td>Dynamic</td>
<td>Relational</td>
<td>Relational</td>
</tr>
<tr>
<td>JDBC Cached Rowset</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>Relational</td>
<td>Relational</td>
</tr>
<tr>
<td>Entity EJB</td>
<td>Connected</td>
<td>Static</td>
<td>Relational</td>
<td>Java Introspection</td>
</tr>
<tr>
<td>JDO</td>
<td>Connected</td>
<td>Static</td>
<td>Relational, Object</td>
<td>Java Introspection</td>
</tr>
<tr>
<td>JCA</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>Record-based</td>
<td>Undefined</td>
</tr>
<tr>
<td>DOM and SAX</td>
<td>Disconnected</td>
<td>Dynamic</td>
<td>XML</td>
<td>XML InfoSet</td>
</tr>
<tr>
<td>JAXB</td>
<td>Disconnected</td>
<td>Static</td>
<td>XML</td>
<td>Java Introspection</td>
</tr>
<tr>
<td>JAX-RPC</td>
<td>Disconnected</td>
<td>Static</td>
<td>XML</td>
<td>Java Introspection</td>
</tr>
</tbody>
</table>
SDO – Summary of Goals

- Unified & consistent data access to heterogeneous data sources
  - Simplified programming model for the application programmer
  - Enable tools and frameworks to work consistently across heterogeneous data sources
- Robust client programming model for several J2EE best practice application patterns
  - Disconnected client programming model
  - Custom data access layers based on common design patterns
- First class support for XML Schema, XML InfoSet, and XML data sources
  - XML/Java bindings
  - JAX-RPC objects
Agenda

- SDO – The Big Picture
- **SDO – Key Concepts and Core APIs**
- SDO in Action
- Summary
SDO Key Features

- Generated Data API
- Dynamic Data API
- Rich Data Objects
- XML and XML Schema integration
- XPath Navigation through graphs of data
- Change Summary
- Metadata
- Relationship integrity
SDO Components

- Generated data API: POJO beans
- Dynamic data API: DataObject
- Change summary API: ChangeSummary
- Introspection API: Type and Property
- XML-based serialization for transferring data sets on the wire
  - Can conform to pre-defined XML Schema
  - Can generate XML Schema
DataObject

- Composed of properties
- Single and many-valued properties
- Properties accessed & modified by name, offset, Property, XPath
- Can contain other DataObjects as properties
- Reverse link to containing DataGraph
public interface Person {
    String getName();
    void setName(String name);
    int getPostalCode();
    void setPostalCode(int code);
}

Person p = (Person) dataFactory.create(Person);
p.setName("John");
p.setPostalCode(94133);  
System.out.println(p.getName());
Dynamic Data API Example

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

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

- `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()`
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
Update SDOs

DataObject customer1 =
customers.getDataObject("customer[1]");
customer1.setString("firstName", "Kevin");

ChangeSummary updated to mark object changed with old value of "Adam"

<customers xmlns="http://customers.com">
  <customer SN="1" firstName="Kevin" />
  <customer SN="2" firstName="Baker" />
</customers>
Example 1 –
Accessing DataObjects

// Get an employee using an SDO xpath expression
// starting from the company
DataObject employee =
  company.getDataObject("departments[number=123]/employees[SN=0002]"巩

// Or, an SDO xpath expression can find the employee
// based on positions in lists:
DataObject employee =
  company.getDataObject("departments.0/employees.1");

// Or, use the API to go step by step to find the employee
// Get the list of departments starting from the company
List departments = company.getList("departments");
// Get the department at index 0 on the list
DataObject department = (DataObject) departments.get(0);
// Get the list of employees for the department
List employees = department.getList("employees");
// Get the employee at index 1 on the list
DataObject employeeFromList = (DataObject) employees.get(1);
Example 2 – Updating DataObjects

// Set the company name
company.setString("name", "ACME");

// create a new employee
DataObject newEmployee = department.createDataObject("employees");
newEmployee.set("name", "Al Smith");
newEmployee.set("SN", "0004");
newEmployee.setBoolean("manager", true);

// Set employeeOfTheMonth to be the new employee
company.set("employeeOfTheMonth", newEmployee);

<company name="ACME" employeeOfTheMonth="0004">
  <departments name="Advanced Technologies" location="NY" number="123">
    <employees name="John Jones" SN="0001"/>
    <employees name="Jane Doe" SN="0003"/>
    <employees name="Al Smith" SN="0004" manager="true"/>
  </departments>
</company>
Instance and Model

SDO

Metamodel

Type \(\rightarrow\) 0..* Property

Model

PurchaseOrder

Customer \(\rightarrow\) 1..* LineItem

Instance Data

XML

XSDType

XSDComplexType

XSDSimpleType

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema ...>
  <xs:element name="PurchaseOrder">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Customer" type="xs:string"/>
        <xs:element name="lineItem">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="count" type="xs:int"/>
              <xs:element name="sku" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PurchaseOrder>
  <Customer>Alice</Customer>
  <LineItem>
    <count>2</count>
    <sku>2834723-23234</sku>
  </LineItem>
  <LineItem>
    <count>4</count>
    <sku>2342342-34544</sku>
  </LineItem>
</PurchaseOrder>
```
SDO Meta-Model

- SDO provides a simple, universal meta-model
  - Used across JavaBeans, XML, or any data source
  - Useful for tools and IDE’s (Model in MVC)
- Meta-data Classes
  - “Type”
    - Has name, URI, instance class, and properties
  - “Property”
    - Has name, type, default value, index within Type
Example – SDO Metadata

```java
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());
}
```
XML / SDO Mapping

XML

```xml
<purchaseOrder orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>PA</state>
    <zip>90952</zip>
  </shipTo>
  ...
</purchaseOrder>
```

SDO

- `orderDate=1999-10-20`
- `country= US`
- `name= Alice Smith`
- `street= 123 Maple Street`
- `city= Mill Valley`
- `state= PA`
- `zip= 90952`

- `XML` to `SDO`
  - `PurchaseOrderType`
  - `USAddress Type`
  - Values
  - Properties
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
## XSD mapping <-> SDO

<table>
<thead>
<tr>
<th>XML Schema Concept</th>
<th>SDO Concept</th>
<th>Java Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>URI for Types</td>
<td>Package</td>
</tr>
<tr>
<td>Simple Type</td>
<td>Type, dataType=true</td>
<td>int, String, BigDecimal, etc.</td>
</tr>
<tr>
<td>Complex Type</td>
<td>Type, dataType=false</td>
<td>Interface</td>
</tr>
<tr>
<td>Attribute</td>
<td>Property</td>
<td>getX(), setX()</td>
</tr>
<tr>
<td>Element</td>
<td>Property</td>
<td>getX(), setX()</td>
</tr>
</tbody>
</table>
Agenda

- SDO – The Big Picture
- SDO – Key Concepts and Core APIs
- **SDO in Action**
- Summary
SDO in Action – Open Source
Open Source SDO – Apache Tuscany

- Tuscany Project (http://incubator.apache.org/tuscany)
- Provides runtime capabilities for applications built using a Service Oriented Architecture (SOA)
- Implementations of two specifications:
  1. Service Component Architecture (SCA)
  2. Service Data Objects (SDO)
  
  plus
  + Data Access Service (DAS)

- In total Tuscany currently includes 5 subprojects:
  1. SCA Runtime for Java
  2. SDO 2.01 Runtime for Java
  3. Data Access Service for Java
  4. SCA runtime for C++
  5. SDO 2.01 Runtime for C++
Apache Tuscany Status

- Initiated January 2006 and currently in incubation
- SDO for Java (sub)project goals:
  1. 100% implementation of 2.01 Specification by end of 2006
     - Currently approximately 60-70% complete
  2. Identify issues with SDO spec and provide feedback for future versions (e.g., 2.1 and 3.0)
  3. Provide value add features possibly including:
     - Static Java code generation
     - Dynamic java class (bytecode) generation (using ASM)
     - Import SDO metadata from (annotated) Java interfaces
     - High performance generated loaders/serializers
     - Lots of other possibilities
- Looking for volunteers to contribute!
  - Go to [http://incubator.apache.org/tuscany/](http://incubator.apache.org/tuscany/) for more info on how to get involved
Agenda

- SDO – The Big Picture
- SDO – Key Concepts and Core APIs
- SDO in Action
- Summary
Summary

- **SDO Programming Model**
  - One model for data across the enterprise
    - XML, Relational, Object
    - Generated and Dynamic
  - SOA patterns (disconnected clients, data services)
  - Efficient change communication across services

- **SDO Components**
  - Generated data API: POJO beans
  - Dynamic data API: DataObject
  - Change summary API: ChangeSummary
  - Introspection API: Type and Property
  - XML serialization on the wire
For More Information

- **SCA / SDO collaboration**

- **SDO specification**

- **SOA roadmap**

- **SDO overview**

- **Open Source**
  - Tuscany Project ([http://incubator.apache.org/tuscany](http://incubator.apache.org/tuscany))
Information as a Service – IBM
SOA Foundation

Enterprise Service Bus

Heterogeneous Applications & Information

DB2
IDS
IMS
IBM Information Server
IBM Content Manager
WAS Portal Server
WAS Process Server

Information Services - SDO

Insightful Relationships

On Demand Business

Insight

In-line, Real-time

Dashboards

Data & Content

Business Context
IBM Business Objects Are SDOs

WebSphere and Rational tools (WID, RAD, …)
WebSphere servers (WAS, WPS, …)

If Approved then
Send letter offering gold
If NOT Approved
Send letter offering Credit counseling service
The following slides were prepared jointly by BEA, IBM, Oracle, SAP and Xcalia, for submission to JavaOne 2006. These slides are reproduced here with permission.
SDO in Action - BEA
Data Services in AquaLogic

DSP

- Logical models capture data access and integration complexity once
- Same data model, programming model, and API for all enterprise data
AquaLogic DSP Update Automation

Update Framework
- XA and non-XA sources
- Automated change decomposition
- Automatic SQL generation for RDBMS
- Hooks for business validations, replacement logic, or compensating transactions
- ADO.NET interoperability
SDO in Action – Oracle
Oracle Fusion Architecture

- **Model Driven**
- **Service Oriented**
- **Grid architecture**
- **Information Centric**
- **Standards based**
- **Hot-pluggable**
SDO in Fusion Applications

- Oracle Fusion Application Services support the SDO Standard
- Service Data Objects facilitate
  - Separate Data Object implementation from Service
  - Easy vertical extensions
  - Standard means for integration (a2a, .net, …)
- **Oracle Business Objects are SDOs**
SDO in Action - SAP
SAP’s Business Process Platform

- **Composite Applications**
  - SAP NetWeaver Business Process Platform
  - Enterprise Services Repository
  - Legacy/3rd Party
  - Partner
  - SAP
  - Platform Process Components

- **PEOPLE PRODUCTIVITY**
- **EMBEDDED ANALYTICS**
- **APPLICATION COMPOSITION**
- **SERVICE ENABLEMENT**
- **BUSINESS PROCESS PLATFORM**
  - Business level models
  - Enterprise services/Business objects
  - Ready-to-run business processes
Service Data Objects in Netweaver

SAP NetWeaver enables loosely coupled and distributed business processes.

*Efficient* and *useful* provision of distributed data is crucial!

Service Data Objects provide the language bindings for
- data representation,
- meta-data access, and
- state transfer

of business data in composite applications.

SAP considers SDO a key technology in the next major SAP NetWeaver release.
SDO in Action - Xcalia
Service Data Objects and Xcalia

- Xcalia is a core contributor and has implemented SDO since 2005
- We have several SDO deployments in production to:
  - Deliver data to rich clients, manifest an extended DTO pattern
  - Manage dynamic, reflexive data models
- The Xcalia Intermediation Platform provides access to both data and service resources.
  - We offer a complete solution that implements JDO, EJB 3.0’s JPA and SDO
  - Our SDO implementation is complemented by our mature data access service
Xcalia’s SDO implementation deals with dynamic mapping of data when no business logic is required (web services, simple web applications, reporting, etc.).

- SDO compliments and does not replace POJO oriented-persistence (EJB3, JDO2)!

### Diagram:

- **Access Type**
  - Connected
  - Disconnected

- **Model Type**
  - Business model
  - Data Model

- **Xcalia Intermediation**
- **SDO**
- **JDO 2 EJB 3**