

Service Data Objects

服务数据对象

刘创 2005/3/12

什么是SDO

- SDO是Java平台的一种数据编程架构和API，它统一了不同数据源类型的数据编程，提供了对通用应用程序模式的健壮支持，并使应用程序、工具和框架更容易查询、读取、更新和检查数据。
- 说白了：基于规范的“高级DTO”
- Service Data Objects (SDO) is a data programming architecture and API for the Java that unifies data programming across data source types, provides robust support for common application patterns, and enable applications, tools, and frameworks to more easily query, view, bind, update, and introspect data.
(from JSR 235)

SDO的特性

- 统一异构数据源间的访问。
- 统一的静态和动态的数据访问API。
- 为工具和其他框架提供支持。
- 支持非连接的编程模型。
- 支持定制化的数据访问层。
- 解耦合应用代码和数据访问代码。

基本概念

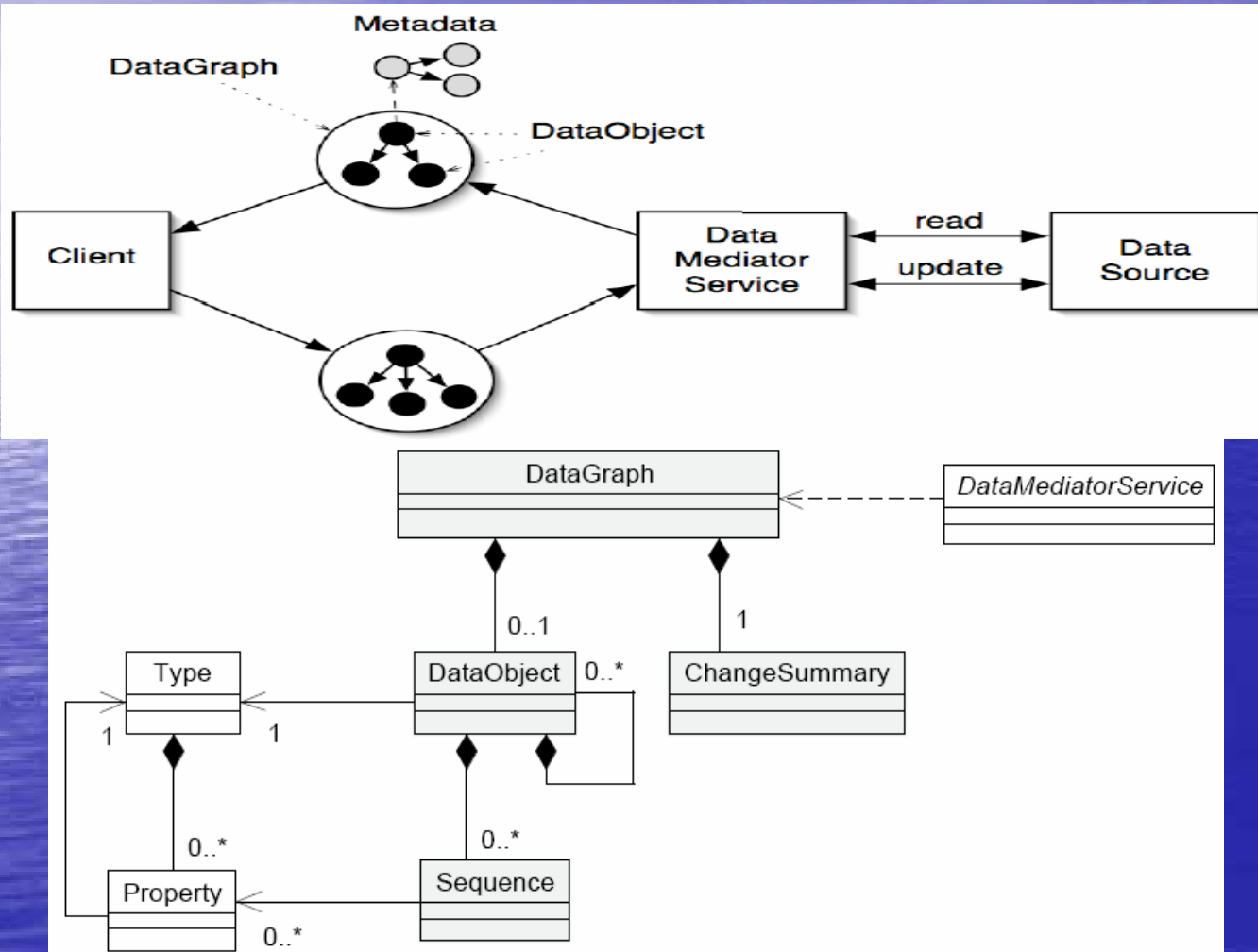
- SDO 客户机
- Data 中介服务(Data mediator service)
- 数据源(Data Source)
- 数据对象(Data Object)
- 数据图 (Data Graph)
- 变更摘要(Change Summary)
- 属性、类型和序列(Property Type Sequence)

详细参见 *SDO*评估文档

各种技术的比较

- SDO 和 WDO 前身
- SDO 和 JDO jdo仅持久层
- SDO 和 EMF 结合
- SDO 和 JAXB 仅序列和反序列
- SDO 和 ADO .NET 最像

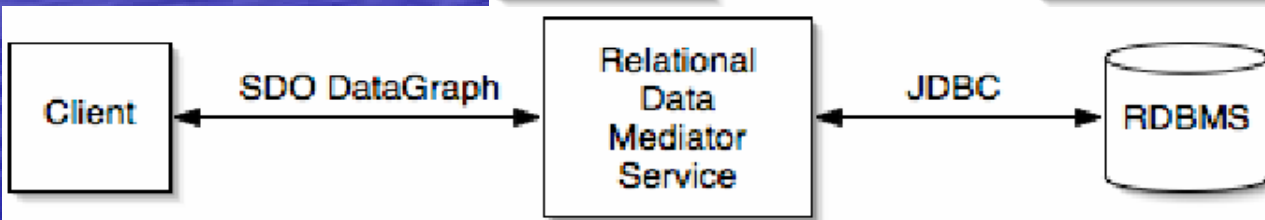
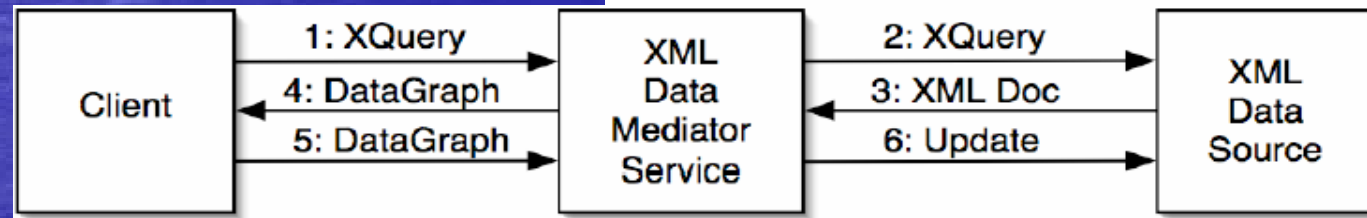
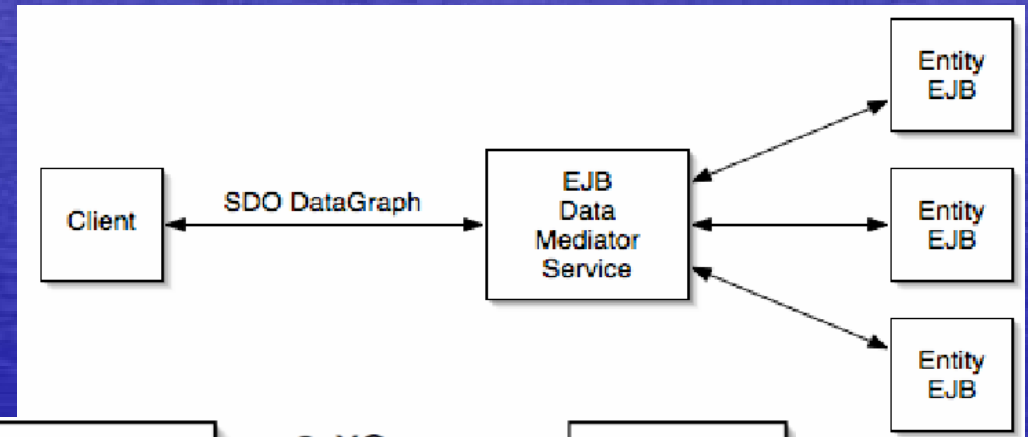
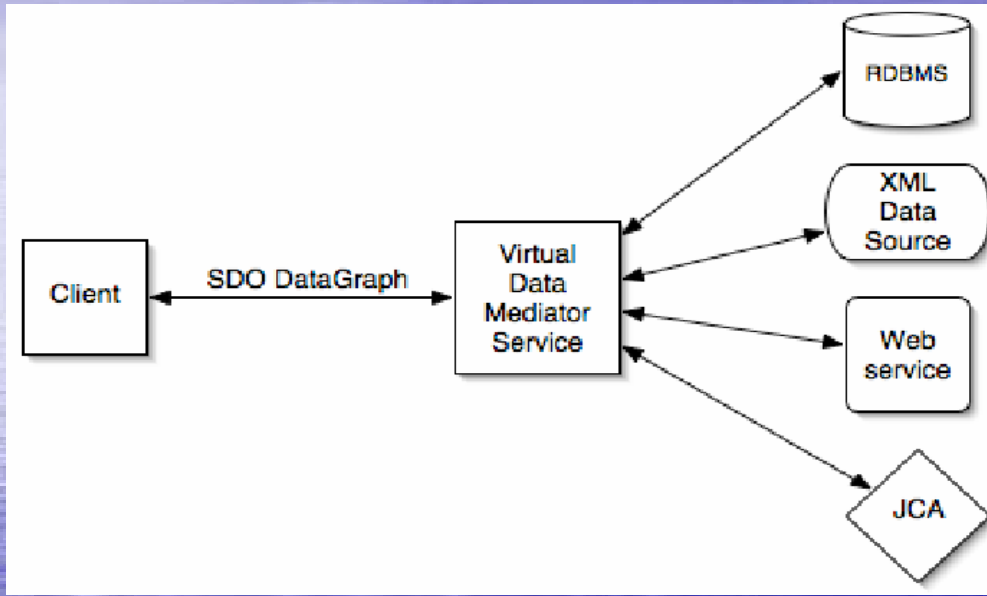
SDO的结构



和其他技术的关系

| | Model | API | Data Source | Metadata API | Query Language |
|------------------------------|--------------|---------|-----------------------|---|------------------|
| JDBC RowSet | Connected | Dynamic | Relational | Relational | SQL |
| JDBC CachedRowSet | Disconnected | Dynamic | Relational | Relational | SQL |
| Entity EJB | Connected | Static | Relational | Java introspection | EJBQL |
| JDO | Connected | Static | Relational, Object | Java introspection | JDOQL |
| JCA | Disconnected | Dynamic | Record-based | Undefined | Undefined |
| DOM and SAX | N/A | Dynamic | XML | XML infoset | XPath, XQuery |
| JAXB | N/A | Static | XML | Java introspection | N/A |
| JAX-RPC | N/A | Static | XML | Java introspection | N/A |
| SDO | Disconnected | Both | Any | SDO metadata API, Java introspection | Any |

DMS的分类



注：SDO 1.0 没有规定标准 DMS API

两种实现方式

- 动态SDO

Data Objects offer, at minimum, a dynamic data API for reading and modifying objects, including the object's properties. This dynamic API uses simple XPath expressions to locate Data Objects within the Data Graph.

- 静态SDO

Optionally, static Java interfaces for Data Objects can be generated from models or schemas (e.g., SQL relational schemas, XML Schema definitions, EMOF models, etc.). This provides a user-friendly API at development time.

元数据的提供

Data Objects can be introspected with the SDO metadata API (described below). This allows programs to get information about types, relationships, and constraints. Note that introspection is not necessarily limited to the SDO metadata API: some implementations might provide access to native metadata, such as an XML Schema object model. However, the SDO metadata API provides the most commonly needed metadata and applications should **revert to other metamodels only when necessary**.

This enables a variety of use cases where **normalized introspection of data** is useful. A common use case is tools that perform data binding between Web UI components and data sources. SDO enables these data binding frameworks to work with XML data, relational data, JCA record data, etc. independent of their native metamodels.

```
DataObject: Type getType();
DataObjectImpl:
public Type getType() {
    return SDOUtil.getType(this); //return a EType
}
EType: EClassifier getEClassifier();
```

关于DataGraph的XML序列化

- In general, the DataGraph serialization consists of a description of the **schema** used for the DataGraph, followed by the **DataObjects** that are contained in the DataGraph, followed by a description of the **changes**. The serialization of DataObjects follows the XMI specification or the XSD for the DataObject model, producing the same XML stream independent of the enclosing DataGraph element. When XML Schema is used as the metadata, the XML serialization of the DataObjects follows the XSD and the resulting XML elements should validate with the XML Schema when all the constraints for the XSD are enforced.
- The description of the schema is **optional** and can be expressed either as an XSD or EMOF model. The description of the changes is also **optional**.
- The **optional** serialization of the ChangeSummary also follows XMI, where properties that have not changed value are omitted.

使用XPath访问数据图中的数据

- `DataObject department = company.getDataObject("departments.0");`
- `DataObject employee = department.getDataObject("employees[SN='0002']");`
- `DataObject employee = company.getDataObject("departments.0/employees[SN='0002']");`

一个例子:

- `<sdo:datagraph xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"`
- `xmlns:company="company.xsd"`
- `xmlns:sdo="commonj.sdo">`
- `<company:company name="ACME" employeeOfTheMonth="#id.0">`
- `<departments name="Advanced Technologies" location="NY"`
- `number="123">`
- `<employees name="John Jones" SN="0001"/>`
- `<employees xmi:id="id.0" name="Mary Smith" SN="0002"`
- `manager="true"/>`
- `<employees name="Jane Doe" SN="0003"/>`
- `</departments>`
- `</company:company>`
- `</sdo:datagraph>`

- `DataObject rootObject = dataGraph.getRootObject();`
- `DataObject company = rootObject.getDataObject("company");`
- `company.setString("name", " MegaCorp");`
- `List departments = company.getList("departments");`
- `DataObject department = (DataObject) departments.get(0);`
- `List employees = department.getList("employees");`
- `DataObject employeeFromList = (DataObject) employees.get(1);`
- `DataObject employeeFromXPath =`
- `company.getDataObject("departments.0/employees.1");`
- `DataObject employeeFromXPathByValue =`
`company.getDataObject("departments[number=123]/employees[SN='0002']");`
- `employeeFromList.delete();`
- `DataObject newEmployee =`
- `department.createDataObject("employees");`
- `newEmployee.set("name", "Al Smith");`
- `newEmployee.set("SN", "0004");`
- `newEmployee.setBoolean("manager", true);`
- `company.set("employeeOfTheMonth", newEmployee);`

- <sdo:datagraph xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
- xmlns:company="company.xsd"
- xmlns:sdo="commonj.sdo">
- <changeSummary create="#id.4" delete="#log.0">
- <company sdo:ref="#id.0" name="ACME" employeeOfTheMonth="#log.0"/>
- <departments sdo:ref="#id.1">
- <employees sdo:ref="#id.2"/>
- <employees xmi:id="log.0" name="Mary Smith" SN="0002"
- manager="true"/>
- <employees sdo:ref="#id.3"/>
- </departments>
- </changeSummary>
- <company:company xmi:id="id.0" name="MegaCorp"
- employeeOfTheMonth="#id.4">
- <departments xmi:id="id.1" name="Advanced Technologies"
- location="NY" number="123">
- <employees xmi:id="id.2" name="John Jones" SN="0001"/>
- <employees xmi:id="id.3" name="Jane Doe" SN="0003"/>
- <employees xmi:id="id.4" name="Al Smith" SN="0004"
- manager="true"/>
- </departments>
- </company:company>
- </sdo:datagraph>

例子

- 一个完整的例子，参看common sdo spec...
- EMF-SDO的例子

参考文献

- IBM和BEA关于SDO的官方网站:
- <http://dev2dev.beasys.com/technologies/commonj/sdo/index.jsp>
- <http://www.ibm.com/developerworks/library/j-commonj-sdowmt/>
- 一个SDO持久化的开源项目，不过，文档奇差，例子我也没有找到，不过可以参考。
- <http://www.sympedia.org/cdo/>
- TTS上对SDO的评价，这篇帖子后面的评价，好像大家都不太看好sdo:
- http://www.theserverside.com/news/thread.tss?thread_id=29160
- IBM上的一片使用SDO进行数据集成的文档:
- <http://www-106.ibm.com/developerworks/db2/library/techarticle/dm-0407saracco/index.html>
- SDO的JSR:
- <http://www.jcp.org/en/jsr/detail?id=235&showPrint>
- IBM中文网站上的教程:
- <http://www-900.ibm.com/developerworks/cn/java/j-sdo/index.shtml?ca=dwcn-newsletter-java>
- EMF/SDO的官方网站:
- <http://www.eclipse.org/emf/>