

Handling Heterogeneous Data Sources in a SOA Environment with Service Data Objects (SDO)

Luciano Resende
IBM
4400 N 1st St Suite 100, San Jose, CA 95134
+1.408.956.3311
luciano_resende@us.ibm.com

Raymond Feng
IBM
577 Airport Blvd., Suite 800, Burlingame, CA 94010
+1.650.558.3388
rfeng@us.ibm.com

ABSTRACT

Service Oriented Architecture (SOA) is gaining momentum in the development community based on its model for creating richer, reusable solutions at lower cost that can adjust to the agile business environments. The solutions built on this principle are composed of services that offer business functionality as a unit or as a whole. These services will access data from different data source types and potentially need to aggregate data from different data source types with different data formats. In this type of environment, handling of different forms of data poses a big challenge for application developers, and the challenge increases in size when you consider the broad diversity of programming models in use. Service Data Objects (SDO) is a specification for a programming model that unifies data programming across data source types and provides robust support for common application patterns in a disconnected way. Although Service Data Objects offer many other capabilities that are very useful for supporting integration with tools and other frameworks, this paper will focus on describing SDOs and how it can be used to handle heterogeneous data sources in SOA environments.

Categories and Subject Descriptors

H.2.3 [Information Systems / Database Management / Languages] : *Database (persistent) programming languages.*

General Terms

Design, Standardization, Languages.

Keywords

Service Oriented architectures, service data objects, data services, xml.

1. Introduction

Service Oriented Architecture (SOA) is starting to proliferate in the Industry as a solution for building business applications based on services. These services are often implemented in different languages, using different data sources and exposed in a distributed environment. In these environments, there is a great need of a programming model that allows handling heterogeneous data with a consistent and uniform set of APIs. This paper will focus on Service Data Object (SDO) characteristics and how it can help handle heterogeneous data sources in distributed SOA environments.

Copyright is held by the author/owner(s).
SIGMOD '07, June 12–14, 2007, Beijing, China.
ACM 978-1-59593-686-8/07/0006.

2. Service Data Objects (SDO)

SDO defines a unified, language independent programming model for handling data across multiple data source types, which could be relational, XML, webservices, or more. SDO is complemented with a data access service, which handles data mediation by handling CRUD operations as well as data introspection and data bindings.

Service Data Objects offer many capabilities that are very useful for supporting integration with tools and other frameworks, such as: Uniform data access across heterogeneous sources, dynamic data API and simple data introspection. But due to the scope of this paper, we are going to focus on the characteristics that give SDO the ability to handle heterogeneous data in distributed SOA environments.

Unified data access to heterogeneous data sources. Current data programming technologies are more or less specific to data source types. In SOA environments and real-world applications, however, data frequently comes from a variety of sources, including relational databases, custom data access layers (implemented with well-known design patterns), Web services, XML data stores, enterprise information systems, etc. The diversity of programming models needed to handle this data heterogeneity poses a big challenge to the development community. Service Data Object provides a common, and simple API that allows handling data from heterogeneous data sources in a uniform way.

Service Data Object represents data using the concept of data graphs, which gets created by Data Access Services that interact with data sources. As the data changes in the SOA application, the change is recorded on the data graph. Data Access Services are also responsible for applying changes from these data graphs back to the data sources. This enables SDO to provide a simple API for handling data from heterogeneous types and from multiple sources in a consistent and unified way.

Unified support for both static and dynamic data APIs. Service Data Objects provides an easy and convenient programming model based on static, strongly typed interfaces (e.g. `account.getFirstName()` or `account.getBalance()`), as well as a dynamic, untyped data APIs (e.g., `rs.getString("FIRST_NAME")` or `rs.getFloat("BALANCE")`).

Neither static nor dynamic data APIs alone are sufficient, however: both are necessary. Static data APIs provide the ease-of-use application programmers needs. But in some cases, static Java interfaces for data is neither feasible nor desirable. For example, dynamic queries where the shape of the resulting data is not known a priori, static Java interfaces for the data is not feasible. Thus, a unified data programming technology needs to support

both static and dynamic data APIs seamlessly. Service Data Objects support both static and dynamic APIs. For the support of static types, tooling can be used to allow the generation of these types based on various schema representations or even by introspection of the data source metadata.

Table 1 – Service Data Object dynamic API

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

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

Table 2 – Service Data Object static API

```
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());
```

Support for disconnected programming models. Many applications naturally have a disconnected usage pattern of data access: an application reads a set of data, retains it locally for a short period of time, manipulates the data, and then applies the changes back to the data source. For example, this is a very common pattern in SOA environment, where a service produce some data in a local read transaction, and then interact with a remote service that would process this data, and then uses a new transaction to update the data. In these scenarios, best practice typically dictates that optimistic concurrency semantics be used for this scenario, which provides for high levels of concurrency with the appropriate business-level semantic. In the case of SDOs, the Data Access Service component, that will be explained latter in this article, is responsible to apply these optimistic concurrency semantics while applying the updates to the data source.

Decouple application code from data access code. In order to be reusable and maintainable, business logic should be separable from data access code. Service Data Objects and its Data Access Services component provide a configuration side file where data access specific information can be externalized, such as data source connectivity, data query using specific query language (e.g. SQL, XQuery), etc. Decoupling data access code from application code

also allows for easier replacement of data sources by only updating these configuration side files.

3. Service Data Objects concepts

As defined in the SDO specification, there are three basic concepts that should be well understood in SDO:

Data Object: Data Objects represents business data, it holds the actual data, including primitive values and references to other Data Objects. Data Objects also have references to their metadata, which allows the Data Object to be introspected for information about data types, data relations, and data constraints.

Data Graph: A Data Graph provides an in-memory, nonpersistent copy of the data where an application can perform data operations even when there is no connection to the original data source. The data inside a Data Graph is organized as a multi-rooted set of Data Objects. It also has the concept of Change Summary where all changes to data, including new Data Objects, changed Data Objects, and removed Data Objects.

Data Access Service: Data Access Service provides a layer of abstraction between SDOs and the data sources. This abstraction layer gives SDOs the flexibility to handle heterogeneous data uniformly.

The relationship between the SDO components is expressed on the diagram below.

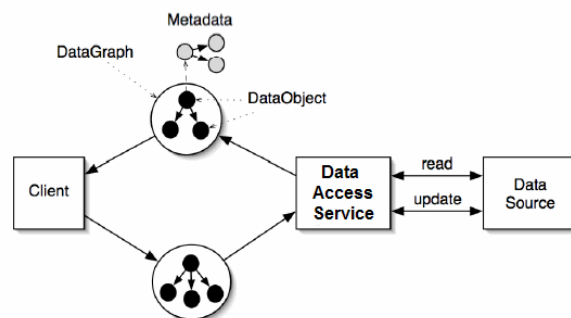


Figure 1 – Components of the SDO solution

The relationships between the objects in SDO can be expressed in a UML model:

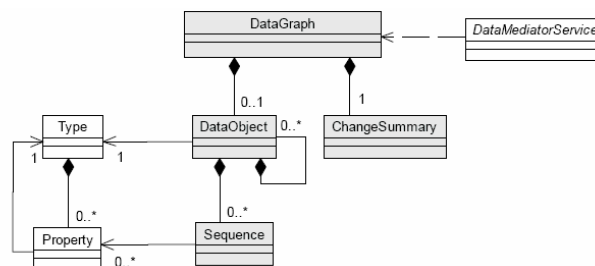


Figure 2 – UML model of core SDO components

There is also a good white paper, “Next-Generation Data Programming: Service Data Objects” [1] that gives more details on SDO characteristics and concepts., and for more information, complete description and Interface definitions on the SDO components, refer to Java SDO Specification v2.1.0 [2].

4. Data Access Services (DAS)

The SDO architecture is based upon the concept of disconnected data graphs. Under the disconnected data graphs pattern, a client retrieves a data graph (i.e., a collection of tree-structured or graph-structured data objects) from a data source, mutates the data graph, and can then apply the data graph changes back to the data source. Most commonly, the update is performed with optimistic concurrency semantics, which means that if any of the underlying data was changed before the client applies the changes, the update is rejected and the application must take corrective action. Optimistic concurrency semantics is a natural semantic that serves most business applications well.

The Data Access Services component is responsible to provide access to data sources. This component is responsible for querying data sources, creating graphs of data containing data objects, and applying changes to data graphs back to data sources. The client is typically disconnected from the data access service: it connects only to retrieve or update a data graph. By adding this data access services layer, applications utilizing SDO can handle heterogeneous data from a variety of data sources uniformly.

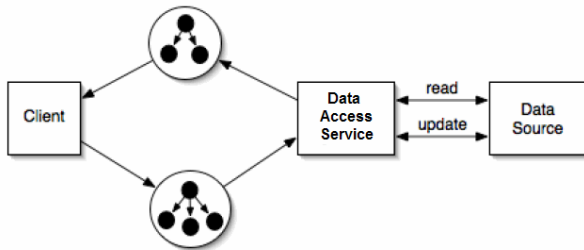


Figure 3 – Disconnected data graph architecture

5. Current SDO Status

Various Industry Partners have been developing the Service Data Objects specifications documents under the Open Service Oriented Architecture (OSOA) collaboration group [3].

The SDO 2.0.1 for Java and C++ final version of the specification is available from the OSOA collaboration group website. You can also find past specification releases, as well as draft specification on the OSOA website [4].

- SDO for Java Specification V2.1
- SDO for C++ Specification V2.1
- SDO for C Specification V2.1 (draft)
- SDO for COBOL Specification V2.1 (draft)

In the near future, these specifications are going to be moved to OASIS

As for concrete implementations, other than the ones available from the various Industry Partners, SDO implementations are also available from open source communities such as Apache Tuscany (Java and C++), Eclipse (Java) and SDO for PHP.

6. Conclusion

This paper provided introduction to Service Data Objects (SDO) to encourage the reader to further pursue the details of this powerful technology for handling data. We outlined how the simple and yet powerful SDO programming model can handle data across heterogeneous data sources in a disconnected fashion and make it a perfect fit for SOA environment.

7. ACKNOWLEDGMENTS

Many thanks to Laura Haas, the ACM SIGMODE Industrial programming chair, for giving us the opportunity to share SDO technology with ACM SIGMOD attendees as well as to those who contributed to this paper.

8. REFERENCES

- [1] Beatty, J., Brodsky, S., Nally, M., Patel, R. Next-Generation Data Programming: Service Data Objects (Nov. 2003)
<http://osoa.org/download/attachments/287/SDO+V2.1+White+Paper.pdf>
- [2] OSOA – Java SDO 2.1.0 Specification
<http://www.osoa.org/download/attachments/36/Java-SDO-Spec-v2.1.0-FINAL.pdf>
- [3] Open Service Oriented Architecture (OSOA)
<http://www.osoa.org>
- [4] Service Data Object Specifications
<http://www.osoa.org/display/Main/Service+Data+Objects+Specifications>