

# Programming Loosely Coupled Data Oriented Systems in Java

*An Oracle White Paper*  
*October 2006*

# Programming Loosely Coupled Data Oriented Systems in Java

## INTRODUCTION

One of the major trends in enterprise software development is simplification of the development model. These simplifications target developer productivity and flexibility. This is most evident in the changes introduced in Java EE 5.0 and the continued growth of infrastructure solutions that embrace this philosophy, such as the Spring Framework.

In the Java world, this trend towards simplified development has led to less complex domain object models augmented with metadata represented through annotations or XML. One of the main goals has been to minimize coupling between application business logic and runtime containers that host that business logic. In practice, this means that developers are able to code normal and familiar Java objects, unencumbered by any unnecessary infrastructure.

While these new approaches to enterprise Java development have gone a long way to improve application development, they do not in themselves address all of the requirements faced by enterprises shifting from traditional application development to Service Oriented Architectures (SOA). SOA developers have benefited from simplified domain models and richer metadata but are still faced with challenges when attempting to consume the domain models dynamically to minimize coupling.

Oracle is helping to lead the development of a new standard called Service Data Objects (SDO) that will address these requirements.

## STANDARDS

SDO specifies a standard way to access and modify business data regardless of how it is physically stored. Developers and architects do not need to know the technical details of how to access a particular back-end data source in order to use SDO in their applications. They can use static or dynamic programming styles and obtain connected as well as disconnected access.

SDO addresses the ability to build loosely coupled services and several other key challenges in SOA:

- Standardized metadata and a common data object interface. This allows SDO consuming services to access and use the domain model's data without coupling to the physical data structure typically captured in a Java class model. This is an important benefit in data process and workflow consumers that are metadata driven.
- Support for marshalling the domain model into XML. This allows you to transport the domain model across service tiers and even implementation languages using a standardized XML binding representation based on the SDO's domain model structure defined in its metadata.
- Support for disconnected processing. An SDO can track changes made to the domain model after it has been transported between services. These changes, represented in a change-set, can be used to later apply the necessary modifications to a data store.

## ORACLE'S PRODUCT STRATEGY

Oracle Fusion Middleware will deliver a complete and compliant SDO implementation. Additional extensions and integration into the Fusion Middleware stack will provide greater flexibility, performance, and customization.

### Persistence

There are many technologies and strategies for solving the challenge of domain data object persistence. Oracle Fusion Middleware offers Java developers flexibility in their approach to Java persistence with Oracle TopLink implementing Java Persistence API (JPA), Oracle Application Development Framework's Business Components (ADF-BC), and direct JDBC access.

SDO will provide a solution that leverages these technologies. Developers will be able to continue accessing their domain model through their existing persistence solution and combine this with SDO to leverage its benefits.

### XML Binding

When developing with SDO, the domain model is commonly generated from an XML schema. The generated SDO model's default XML binding will be to this XML schema. While this provides the minimal XML binding support required, it lacks the flexibility that SOA developers may require. The Oracle Fusion Middleware SDO implementation leverages **Oracle TopLink JAXB** to provide additional flexibility and optimization to the SDO-XML binding.

**Oracle TopLink is an industry leading object-relational persistence solution.**

**There is an effort underway to define a Data Access Service that will standardize persistent access of SDOs.**

Additional mapping types, mapping configurations, and tuning options are supported by TopLink to enable much greater control of the domain model's structure versus that of the XML schema(s). For greater efficiency the TopLink JAXB solution supports partial conversion and lazy pulling of XML into the domain model. The particular parser used is configurable allowing any JAXP compliant SAX, DOM, or STaX parser to be used.

## Business Logic

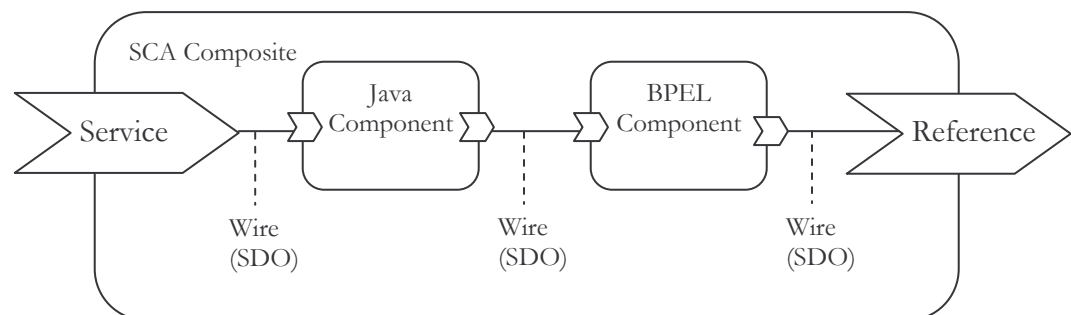
Within a service, developers code or configure specific business functionality. Examples of this include developing a service using an EJB Session Bean, which can be exposed as a Web Service, or defining a Business Process Execution Language (BPEL) process as a service.

BPEL will be seamlessly integrated with SDO. Typically, the services in a BPEL process communicate through the exchange of XML documents. The transformations and process decisions are based upon the data in these documents. Within Oracle Fusion Middleware the BPEL engine will support managing these documents as SDOs.

An SDO can be used within BPEL in conjunction with SCA. Within an SCA composite, an external service reference can be defined that leverages SDO as the data passing mechanism. This in turn enables an SCA component, which can be implemented by BPEL, to communicate with the external service, which can be a Data Mediator Service. During the lifecycle of the BPEL process instance the Data Mediator Service will be contacted automatically to fetch data and persist data changes using SDO.

BPEL's support of SDO as variables is especially valuable when it is not desirable to maintain a duplicated persistent copy of the data outside of application because the data may be concurrently modified by other interfaces, or infrequently used within the process. This feature enables BPEL accessing and pulling in data uniquely identified with a set of keys only when needed.

**Service Component Architecture (SCA):**  
Simplifies the representation of service-oriented business logic. SCA gives developers and architects the ability to represent business logic as reusable components that can be easily integrated into any SCA-compliant application or solution.



## **Presentation**

Service data objects can also be used within the presentation tier of an application or service. Leading Java presentation frameworks support the use of Java objects along with supplied and assumed metadata. Oracle Fusion Middleware will support the use of SDO's within its Application Developer Framework (ADF). Oracle ADF is an end-to-end Java EE framework that simplifies development by providing out of the box infrastructure services and a visual and declarative development experience. By building their SDO models using the data-binding (JSR 227) capabilities of ADF, SDO developers can benefit from the improved productivity of visually constructing the web and rich client interfaces of their SDO applications using Java Server Faces (JSF) and Ajax capabilities.

## **CONCLUSION**

Oracle's commitment to simplifying SOA development in Java is exemplified through its leadership with Java EE 5.0. Enterprise Java developers can now address complex persistence and business logic requirements more efficiently than ever. As these enterprises shift their focus from application to service development, the SDO support offered in Oracle Fusion Middleware will augment this functionality to better support the challenges of loosely coupled data-centric SOA solutions.

While the usage of SDO may not be ideal for all service development projects, it does fit well as a core SOA infrastructure and in services where consumers are more dynamic and metadata driven such as BPEL. Oracle Fusion Middleware offers SOA developer the ability to develop their domain models using either a Plain Old Java Object (POJO) approach (providing metadata for persistence and XML binding) or an SDO approach. Both styles have their benefits and can function well together.



Programming Loosely Coupled Data Oriented Systems in Java  
October 2006  
Author: Douglas Clarke  
Contributing Authors: Shaun Smith, Greg Pavlick, Alex Yiu

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[oracle.com](http://oracle.com)

Copyright © 2006, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.