SDO and SCA

Martin Nally

IBM Fellow, CTO IBM Rational

Service Data Objects (SDO)

Rational. software

SDO motivation

- SDO started in the IBM tools team writing tools to simplify web applications for enterprise data
- Too many options for enterprise data, all different
  - JDBC, XML (DOM, SAX, …), web services, JCA connectors, EJBs, messaging, POJOs, …
- Different in 3 dimensions
  - Data representation
    - Used to be part of the programming languages
  - Data access
    - This problem spawned generations of 4GLs
    - Optimistic concurrency? Deadlock?
  - Meta-data
    - Ill wind that nobody blows good
- Compete with MS ADO Dataset

Service Data Objects Features

- Service Data Objects (SDO) provide:
  - Uniform access to data from heterogeneous sources
    - XML, RDB, POJO, SOAP, JCA/COBOL, etc...
  - Both static and dynamic programming models
  - Meta-data for easy introspection of data types
  - Disconnected object graph capable of tracking changes
  - Xpath navigation over in-memory data

- Implementations exist in Java, C++ and PHP

What are Service Data Objects?

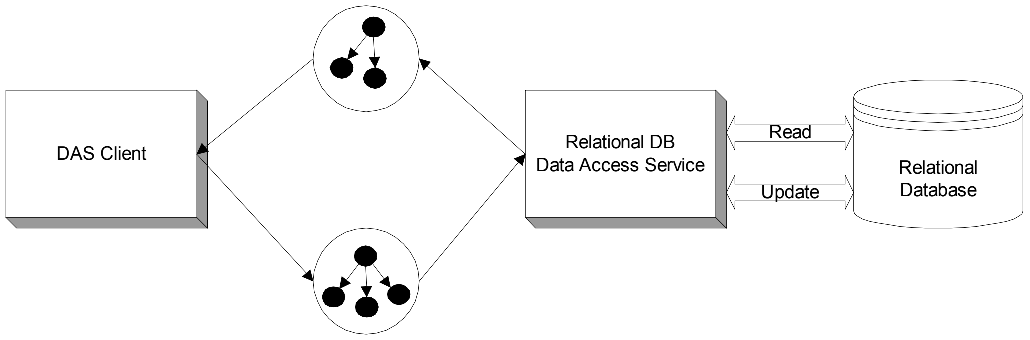
- Data Objects
  - Simple, in-memory objects that represent business data
  - Objects can form a tree or graph
  - Nothing technology-specific (a few specific helper classes kept on the side)
  - Objects may be of generated, statically-typed classes or a standard generic, dynamically-typed class
    - Generated classes have get(), set (value) methods
    - Dynamic classes get (name), set (name, value)
  - Richer meta-data includes type, property, cardinality, relationships, inverses,
    - Meta-data available at run-time
    - Generated and Dynamic objects have same meta-data
  - Relationship referential integrity (inverse and cardinality management)
  - Basic data types are those of the host language
- Data Access Service
  - A service that “gets” and “puts” graphs of data objects

Service Data Objects – Initial 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
- Support for XML data and XML data sources amongst many others
  - XML/Java bindings
  - JAX-RPC objects

SDO in Practice

- Two primary SDO use cases have emerged in practice
- SDO as simplified programming model for disconnected access to RDB



- SDO provides a dynamic object binding for XML
  - SCA/WAS/WPS/WESB programmer who wishes to read/write/modify XML using a dynamic object API
  - The XML is something that conforms to a predefined XML Schema, for example a Business Object or Message. The schema is often defined by a third party

SDO for RDB

- SDO focuses mostly on the data and meta-data APIs
- Overall programming model depends on the Data Access Service
  - Simple: define query(-ies) in DAS in some form, deduce SDO object shapes from query result, allow both static (via tool code-generation) and dynamic approaches
  - More complex O<->R mapping easily possible

Standard XML Schema types

- XSD has become the "standard" way to define many data structures shared by industry-specific applications
- Industrial schemas:
  - OTA (Travel) - http://www.opentravel.org/
    - OpenTravel's primary activity is to develop and maintain a library of Extensible Markup Language (XML) schemas for use by the travel industry
  - HL7 (Health Care) - http://www.hl7.org/
    - Develop coherent, extendible standards that permit structured, encoded health care information of the type required to support patient care, to be exchanged between computer applications while preserving meaning
  - OAGIS (B2B) - http://www.openapplications.org/
  - UBL (B2B) - http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html
  - ACORD (Insurance) - http://www.acord.org/
  - Parlay (Telco)
  - SWIFT (Financial)
  - IFX (Financial)
  - OFX (Financial)
  - PIDXML (Petroleum Trading)

Thousands of types
- performance issue during importing and building
- usability, only a small subset of types are used, and hard to locate type used or root type

SDO for XML

- To be successful in this scenario SDO must provide:
  - 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
- SDO 3 plans to further improve SDO in these areas
- IBM plans to focus on this XML scenario
  - with additional implementation features:
    - data virtualization support, i.e., the "XML document" may not have a natural physical serialization as XML (e.g., COBOL data structures)
    - lazy loading and large object support

Open Service Orientated Architecture

- Informal alliance of industry leaders with common goal:
  - Define a language-neutral programming model for developing software that exploits Service Oriented Architecture characteristics and benefits
- Service Data Object (SDO)
  - Simplify and unify the way applications handle data
  - Data Access Services (DAS) definition
- Service Component Architecture (SCA)
  - For building applications and systems using a SOA

SDO Specification Status

- SDO 2.1 http://www.osoa.org/display/Main/Service+Data+Objects+Home
  - This is the "current" published version
  - Maintained by OSOA collaboration (not a "standards body")
  - No compliance tests (TCK)
- SDO 2.1.1 http://jcp.org/en/jsr/detail?id=235
  - Next version currently being finalized (target 4Q/08)
  - Relatively small change over 2.1 (mostly errata and clarifications)
  - This is the Java (JCP) standard version (JSR 235)
    - Oracle (previously BEA) is providing the RI
    - IBM is providing the TCK
  - Since JCP is Java only, OASIS will standardize non-Java languages
- SDO 3 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=sdo
  - Multi-language standard (including Java) at OASIS
    - final Java review draft target 03/09
  - 12 functional changes are in scope for SDO 3 including
    - better support and fidelity for XML/XSD
    - unification of static SDO with other standards such as JAX-B

SDO Implementation Status

SDO 2.1

- Open source:
  1. Apache Tuscany (http://tuscany.apache.org)
     - Initial contribution from IBM
     - Implementations in Java and C++ (also support for PHP client)
  2. EclipseLink (http://www.eclipse.org/eclipselink)
     - Initial contribution from Oracle
- BEA (now Oracle), SAP and Rogue Wave, also have 2.1 implementations

SDO 2.1.1

- Oracle is in final phase of delivery of the JSR 235 RI (based on BEA implementation)
- IBM is in final phase of delivery of the JSR 235 TCK
- Some companies are upgrading their 2.1 implementations to 2.1.1 (IBM is not planning to implement SDO 2.1.1)

SDO 3

- Some companies are starting to implement 3.0 features, ahead of the spec
  - IBM is especially interested in XML fidelity improvements

DAS Status

Specification Status

- Informal (OSOA) collaboration has been meeting regularly
- Target date for R1.0 is still not decided
- R1.0 is probably going to be tied to SDO 3, which will provide enhancements in support of DAS (e.g., key support, change summary improvements)
- Data Direct (formally XCalia) is leading this TC. IBM, SAP, and Oracle(+BEA) are also participating

Implementation Status

- Apache Tuscany includes an RDB DAS implementation
- Tuscany is not compliant with any DAS specification
- IBM currently has no plans to implement a compliant SDO DAS

Alternatives to SDO

- Just use what's defined by the APIs of the various back-ends
  - Pros: We know it works – don't need more abstractions, frameworks
  - Cons: Lots of different APIs, hard to write tools, hard to write frameworks
- JAXB
  - Only works for XML
  - Hybrid between Object concepts and XML ones. Can SDO 3 avoid this?
  - No support for recording changes – may require programmatic diff
  - Meta-data is weaker
  - No dynamic capability

SDO alternatives

- JPA
  - Only does RDBMS (some failed attempts to generalize at IBM)
  - Implicit data access triggered by relationship traversal (single-level store)
    - In SDO, data access is explicit, (IBM’s EMF implementation can do both)
  - Limited, proprietary query language
  - Meta-data is weaker

SDO Alternatives

- LINQ
  - Add query capabilities to the programming language (SQL-like, or SQL-subset)
  - Support query over programming language objects, collections
  - Adds support for mapping to a real relational DB (LINQ to SQL)
  - Provides O<->R mapping, generates SQL query (LINQ to Entities)
    - MS's JPA competitor
  - Adds special API for XML (LINQ to XML)
    - API alternatives for W3C DOM ("my API is better than your API")
  - No support for change history?

SDO alternatives

- PureQuery – use SQL, don't invent
  - PureQuery can be used stand-alone, or as an underpinning for SDO or JPA
  - Standard "embedded" SQL query on in-memory objects, collections
    - Does not change host language
  - Tools extensions give code-assist, static type checking
  - Thin layer on top of JDBC makes simple cases much simpler
    - Doesn't get in the way of problem determination, optimization
    - Access to full SQL of target DB
  - Framework for O<->R mapping
    - Integrates with JPA or SDO or roll-your-own
    - Many nice features for supporting standard patterns (disconnected, two-way-join, three-way-join, …)
      - SDO would view it as the dream toolkit for implementing DAS's

SDO alternatives

- XJ
  - Language extensions
    - Use XML infoset data model, not language object/type model
    - In-line XML construction
    - Integrated XPath
    - In-place XML updates
    - Dynamic or static typing – optional import of XML Schemas
  - Efficient implementation
    - Native XPath compilation support
  - Eclipse plug-in
  - No support for change history

IBM

Sample XJ Program

Refer to schema declarations

Inline XPath

XML Construction

```
import com.amazon.devheavy.*;

import com.amazon.devheavy.ProductInfo.Details;

import myOutput.*;

public Book checkReviews (ProductInfo pi,  String title, String author, float goodRating) {

      Sequence<Details>  bookSeq = pi[| //Details[ProductName = $title] |];

      for (XMLCursor<Details> iter = bookSeq.iterator(); iter.hasNext();) {

            Details book = iter.next();

             if (!book[| [$author = .//Author] |].isEmpty())  {

                  double discountMult = 0.5;

                  if (!book[| [.//AvgCustomerRating >  $goodRating] |].isEmpty() )
                        discountMult = 0.75;

                   Book result = new Book(<Book name='{title}' >
                                          <isbn> {book[| /Isbn |] } </isbn>
                                                             <price> {book[| //Price |] *
discountMult} </price>                                       </Book>);

                  return result;

}}}
```

Conclusion

- Many permutations have been explored
  - Object-centric (SDO, JPA, LINQ, pureQuery, JAXB), data-centric (XJ, pureQuery, LINQ to XML)
  - Standard query languages (XJ, PureQuery) or proprietary (LINQ, JPA)
  - Implicit access to secondary storage (JPA, LINQ) or explicit (SDO, pureQuery, LINQ)
  - Language extensions (LINQ, XJ) or frameworks/tools (SDO, pureQuery, JAXB)
- No clear winner(s) yet
  - Trade-off simplicity, control for abstraction
  - Some systems offer multiple approaches (LINQ, pureQuery)

Service Component Architecture (SCA)

Simplicity, **C**onsumability, **A**gility

SCA motivation

- SCA started in an IBM team with a goal to simplify deployment and configuration of implementation artifacts
- Points of Variability are a common feature of implementations in various technologies
  - E.g. The time-zone or locale in which the software is deployed
  - E.g. addresses of other deployed applications that are messaged or invoked
    - In SOA, this last example is especially important (the addresses are called services)
- Many ways to express these POVs, and configure their values on deployment
  - Even within J2EE, address POV are specified differently by technology
- Many ways in which the implementation artifacts themselves are deployed
- New languages, like BPEL, were about to create more
- Special models for “service mediations” were being proposed
- This made it very hard for users, and very difficult and expensive to tool

SCA in a Nutshell

- A development and deployment model for SOA
- Service-based models for the
  - Construction
  - Assembly
  - Deployment
- of composite service applications
- In a distributed and heterogeneous environment of
  - Multiple languages
  - Multiple container technologies
  - Multiple service access methods

History

- Fall 2002-2003: JService Design in IBM
- Dec 2003: Start collaborating with BEA on SCA
- Nov 2005: 0.9 specs published
  - BEA, IBM, Oracle, SAP, IONA, and Sybase
- July 2006: 0.95 specs and OSOA.org (Open SOA)
  - Added: Cape Clear, Interface21, Primeton Technologies, Progress Software, Red Hat, Rogue Wave, Siemens AG, Software AG, Sun, TIBCO
- Mar 2007: 1.0 specs published
  - Submitted to OASIS
- April 2007: OASIS Forms Open CSA Member Section
- Sept 2007: Formal standardization starts in OASIS Open CSA

Standards and Open Source

Open SOA - http://www.osoa.org

OASIS Open CSA - http://www.oasis-opencsa.org/

Apache Tuscany - http://cwiki.apache.org/TUSCANY/home.html

Quick Tour – Construction and Assembly

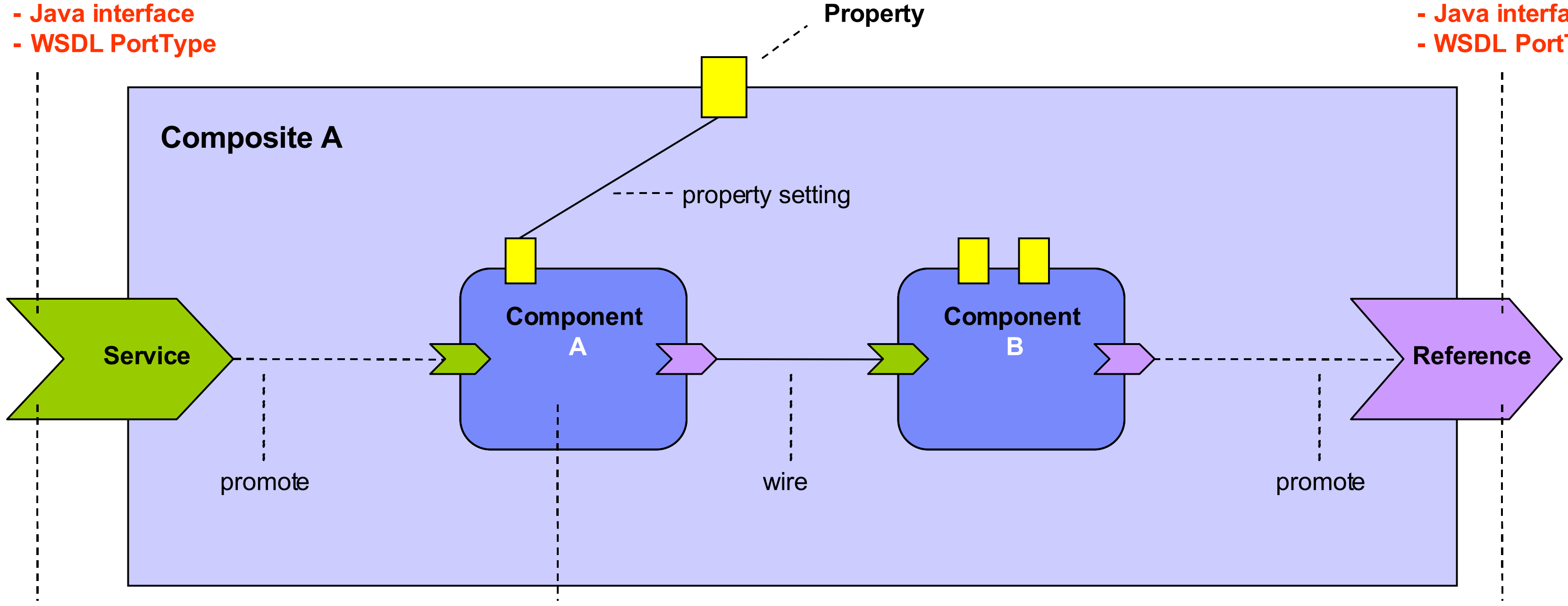
- **Construction** - component definition and implementation
  - **Terminology: A component is an instance in SCA, not a class**
  - **many implementation types**
    - programming languages, scripting languages, dsl's, …
    - focus on business logic, choose language to best fit business problem
    - no APIs
  - **define implementation features**
    - **services** (provided, referenced)
    - **properties**

- **Assembly/Composition** - component configuration
  - **services**
    - – provided: set protocol binding
    - – referenced: wire to target service, set protocol binding
  - **properties**
    - – set property value

Quick Tour – Assembly

Service Interface
- Java interface
- WSDL PortType

Reference Interface
- Java interface
- WSDL PortType



Service Binding
- Web Service
- JMS
- JCA
- SLSB
- HTTP
- JSONRPC
- ATOM
- …

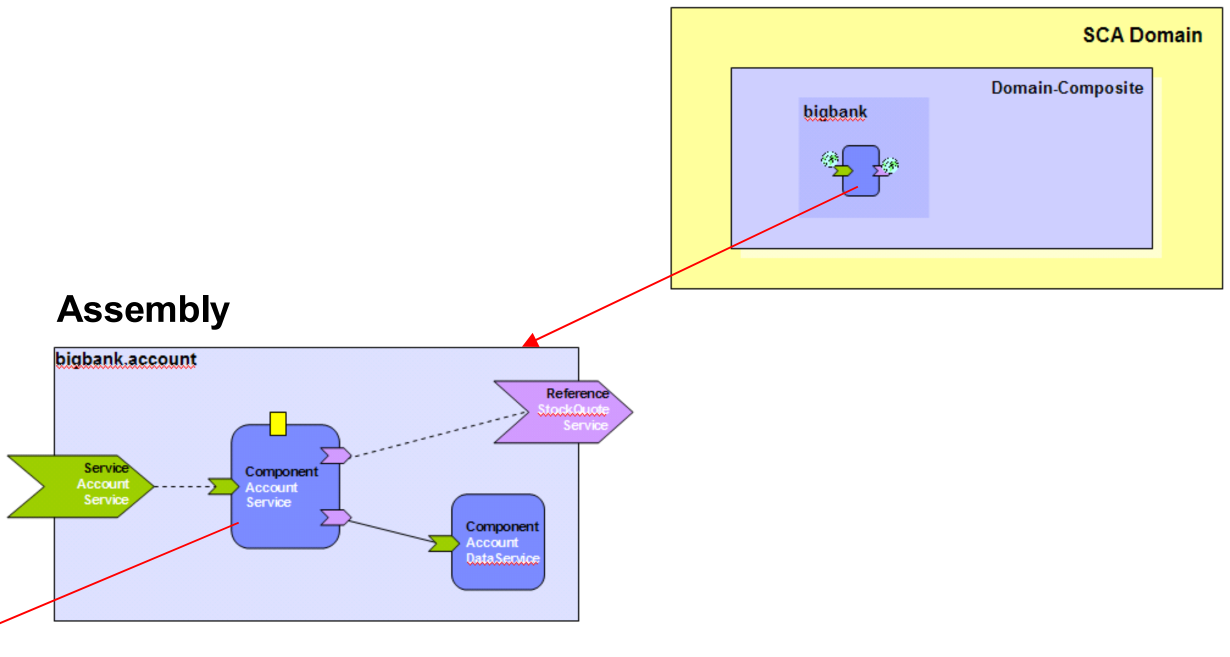
Implementation
- Java
- BPEL
- **SCA Composite**
- Spring
- JEE
- Scripting: Groovy, JScript, PHP, Python, Ruby, …
- XQuery
- …

Reference Binding
- Web Service
- JMS
- JCA
- SLSB
- HTTP
- JSONRPC
- ATOM
- …

Quick Tour - Deployment

Deployment

Assembly



Construction

```
@Remotable
public interface AccountService {

    AccountReport getAccountReport(String customerID);
}

public class AccountServiceImpl implements AccountService {
    ...
    @Reference
    public void setAccountDataService(AccountDataService value) {
        accountDataService = value;
    }
    @Reference
    public void setStockQuoteService(StockQuoteService value) {
        stockQuoteService = value;
    }

    @Property
    public void setCurrency(String value) {
        currency = value;
    }
    ...
}
```

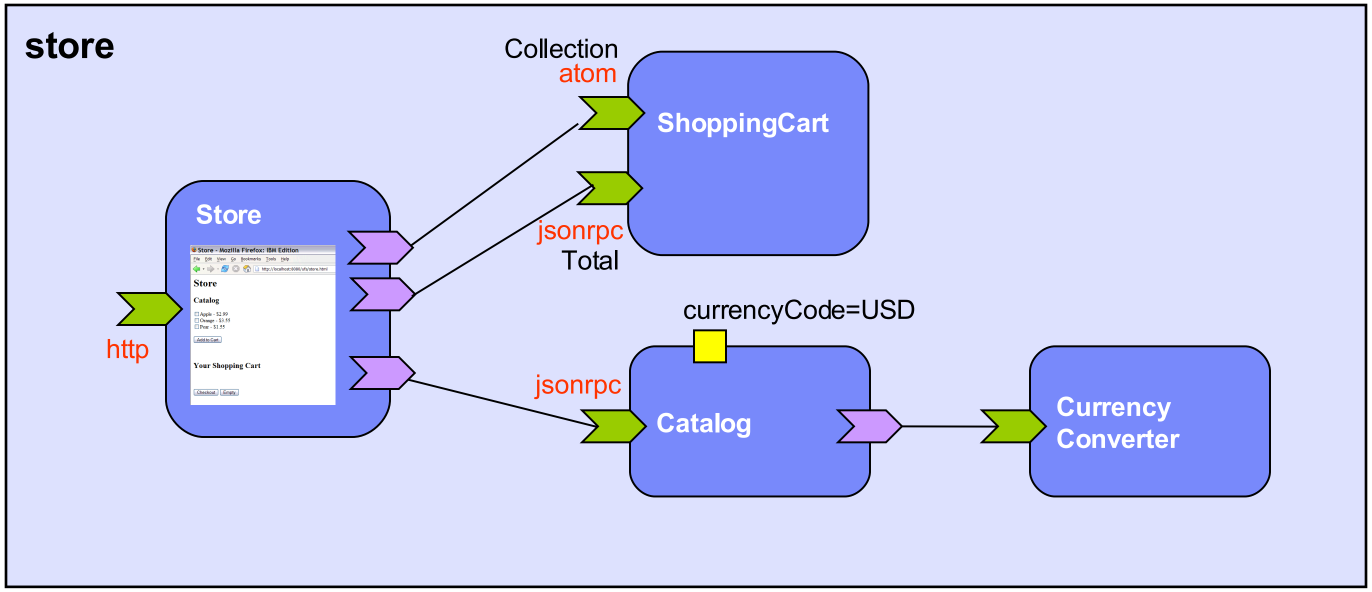
SCA in Action - Business Value Scenarios

The Rise of a Fruit Business

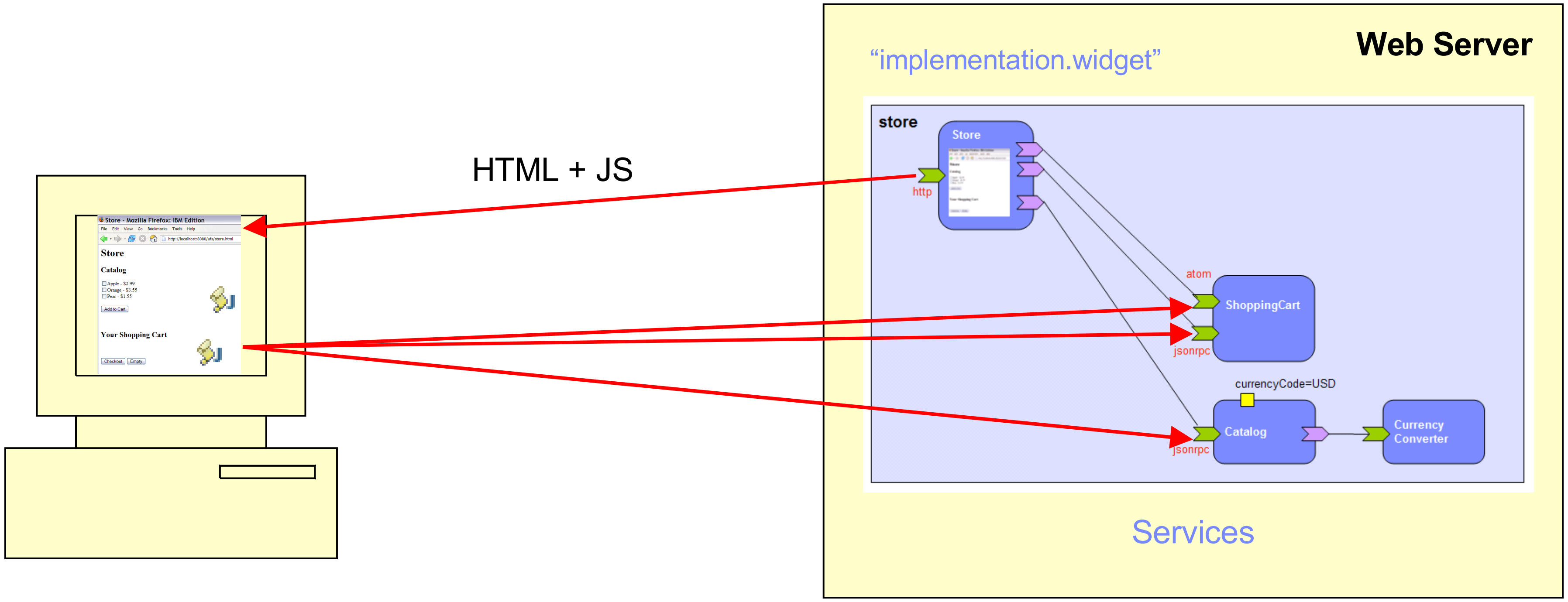
- The Fruit Store
- The Fruit&Vegetable Store
- The Fruit&Vegetable Store as Supplier
- The Fruit&Vegetable Store Solution Provider
- The Fruit Store Widget

The Fruit Store

- Creating an Online Business



Web 2.0 Composite Applications



"implementation.widget"

- HTML + Javascript with SCA reference wiring
- Access services from scripts – with async

The store composite

```
<composite name="store" … >

    <component name="Store">
      <t:implementation.widget location="uiservices/store.html"/>
       …
    </component>

   <component name="Catalog">
      <implementation.java class="services.FruitCatalogImpl"/>
      <property name="currencyCode">USD</property>
      <service name="Catalog">
             <t:binding.jsonrpc/>
      </service>
      <reference name="currencyConverter" target="CurrencyConverter"/>
   </component>

   <component name="ShoppingCart">
      <implementation.java class="services.ShoppingCartImpl"/>
      <service name="Cart">
             <t:binding.atom/>
      </service>
      <service name="Total">
             <t:binding.jsonrpc/>
      </service>
   </component>

   <component name="CurrencyConverter">
      <implementation.java class="services.CurrencyConverterImpl"/>
   </component>

</composite>
```

POJO Component Implementation

Catalog Implementation

```
@Remotable
public interface Catalog {
    Item[] get();
}

public class FruitCatalogImpl implements Catalog {

    @Property
    public String currencyCode = "USD";

    @Reference
    public CurrencyConverter currencyConverter;

    private List<Item> catalog = new ArrayList<Item>();

    @Init
    public void init() {
        String currencySymbol = currencyConverter.getCurrencySymbol(currencyCode);
        catalog.add(new Item("Apple",  currencySymbol +
                    currencyConverter.getConversion("USD", currencyCode, 2.99)));
            …
    }

    public Item[] get() {
        Item[] catalogArray = new Item[catalog.size()];
        catalog.toArray(catalogArray);
        return catalogArray;
    }
}
```

Marks interface usable over remote link

The service offered

Defines settable property

Defines a reference to service provided elsewhere

POJO Component Configuration

Catalog Configuration

```
<composite name="store" … >

    …

    <component name="Catalog">
      <implementation.java class="services.FruitCatalogImpl"/>
      <property name="currencyCode">USD</property>
      <service name="Catalog">
           <t:binding.jsonrpc/>
      </service>
      <reference name="currencyConverter" target="CurrencyConverter"/>
    </component>


    <component name="CurrencyConverter">
      <implementation.java class="services.CurrencyConverterImpl"/>
    </component>

</composite>
```

Set property value

Set JSONRPC binding

Wire reference to target service

HTML & JS SCA Implementation

Store Implementation

```
<html>
<head>
<title>Store</title>

<script type="text/javascript" src="store.js"></script>
<script language="JavaScript">

   //@Reference
   var catalog = new Reference("catalog");

   //@Reference
   var shoppingCart = new Reference("shoppingCart");

   //@Reference
   var shoppingTotal = new Reference("shoppingTotal");

   …

   function catalog_getResponse(items) {…}
   function shoppingCart_getResponse(feed) {…}

   function init() {
      catalog.get(catalog_getResponse);
      shoppingCart.get("", shoppingCart_getResponse);
   }

</script>

</head>

<body onload="init()">
<h1>Store</h1>
   …
</body>
</html>
```

Defines references to services

Call reference operations

HTML & JS Component Configuration

Store Configuration

```
<composite name="store" … >

   <component name="Store">
      <t:implementation.widget location="uiservices/store.html"/>
      <service name="Widget">
           <t:binding.http uri="/ui"/>
      </service>
      <reference name="catalog" target="Catalog">
           <t:binding.jsonrpc/>
      </reference>
      <reference name="shoppingCart" target="ShoppingCart/Cart">
           <t:binding.atom/>
      </reference>
      <reference name="shoppingTotal" target="ShoppingCart/Total">
           <t:binding.jsonrpc/>
      </reference>
   </component>

   <component name="ShoppingCart">
      <implementation.java class="services.ShoppingCartImpl"/>
      <service name="Cart">
           <t:binding.atom uri="/ShoppingCart/Cart"/>
      </service>
      <service name="Total">
           <t:binding.jsonrpc/>
      </service>
   </component>
   …

</composite>
```

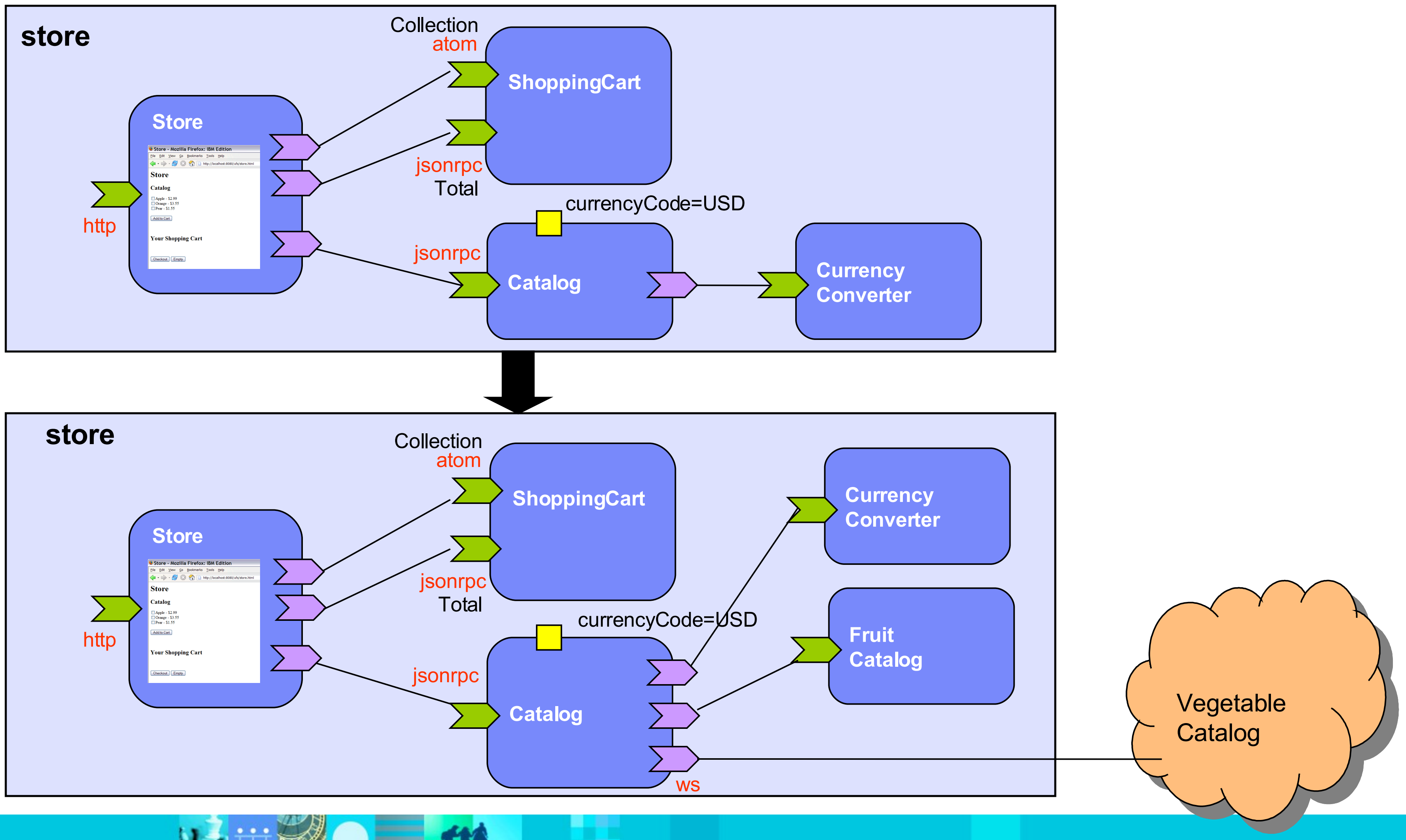
HTML & JS implementation

HTTP binding & address

Catalog service via JSONRPC

Wire

The Fruit&Vegetable Store

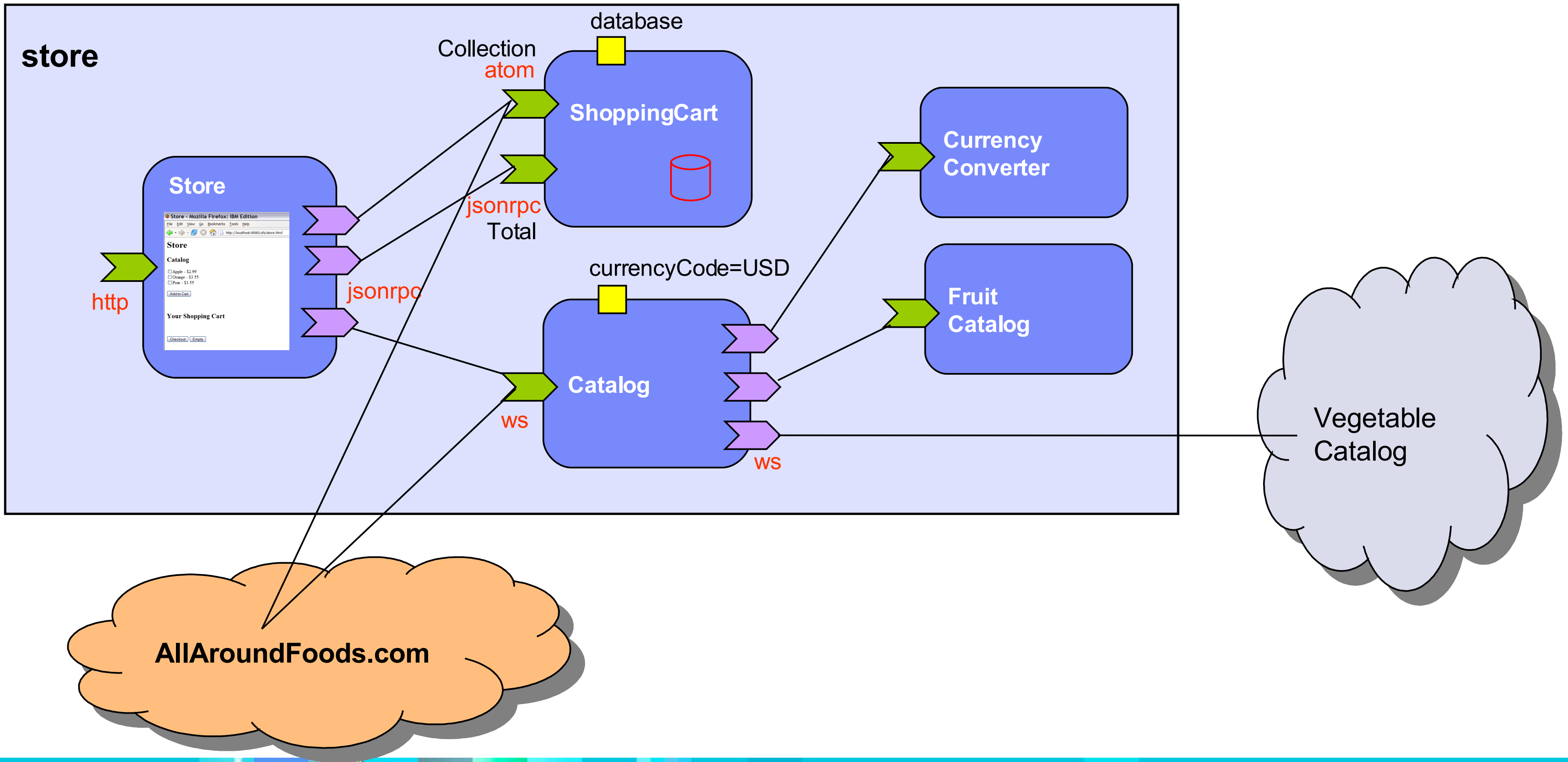


The Fruit&Vegetable Store Composite

```
<composite name="store-merger" … >
  …
  <component name="FruitCatalog">
     <implementation.java class="services.FruitCatalogImpl"/>
     <property name="currencyCode">USD</property>
     <reference name="currencyConverter" target="CurrencyConverter"/>
  </component>
  <component name="Catalog">
     <implementation.java class="services.merger.MergedCatalogImpl"/>
     <property name="currencyCode">USD</property>
     <service name="Catalog">
          <t:binding.jsonrpc/>
      </service>
     <reference name="fruitCatalog" target="FruitCatalog"/>
     <reference name="vegetableCatalog"
          <binding.ws uri="http://veggie.com/Catalog"/>
     </reference>
     <reference name="currencyConverter" target="CurrencyConverter"/>
  </component>
  …
</composite>
```

The Fruit&Vegetable Store as Supplier

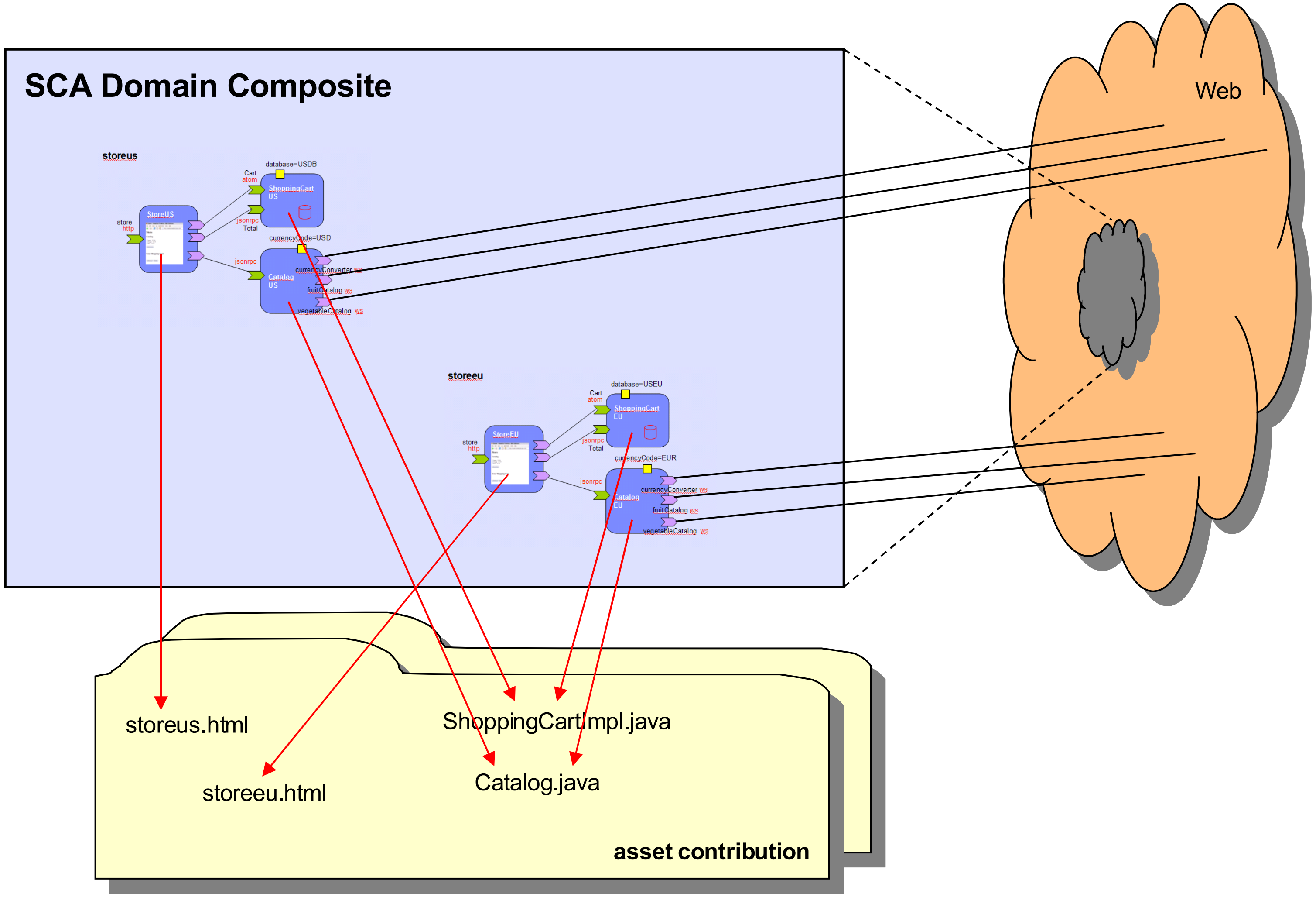
- Being a Supplier for other Online Stores



The supplier composite

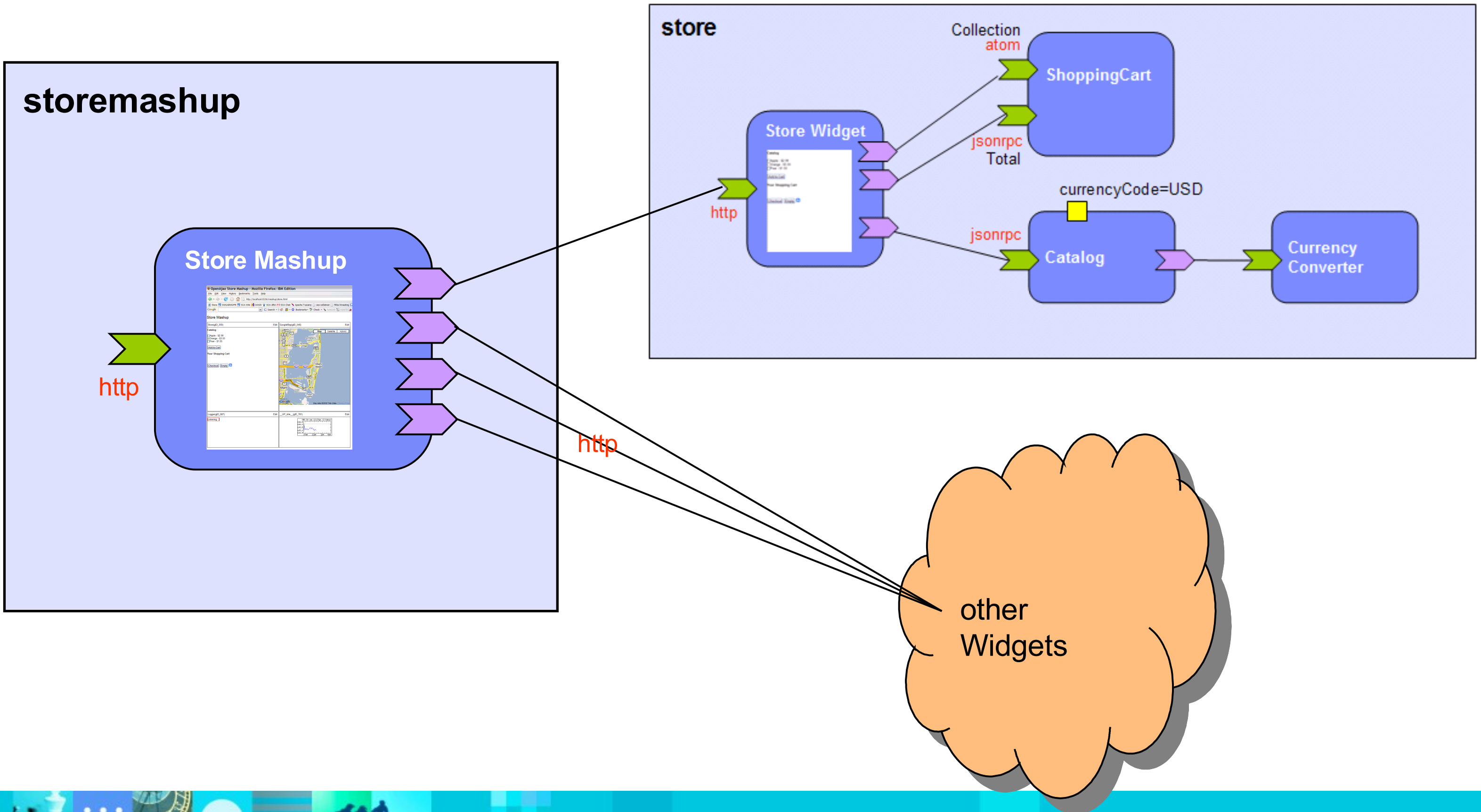
```
<composite name="store-merger" … >

   …

   <component name="Catalog">
      <implementation.java class="services.merger.MergedCatalogImpl"/>
      <property name="currencyCode">USD</property>
      <service name="Catalog">
           <t:binding.jsonrpc/>
           <binding.ws uri="/CatalogWebService"/>
       </service>
      <reference name="fruitsCatalog" target="FruitsCatalog"/>
      <reference name="vegetablesCatalog"
           <binding.ws uri="http://veggie.com/Catalog"/>
      </reference>
      <reference name="currencyConverter" target="CurrencyConverter"/>
   </component>

   …

</composite>
```

Building Solutions from Assets

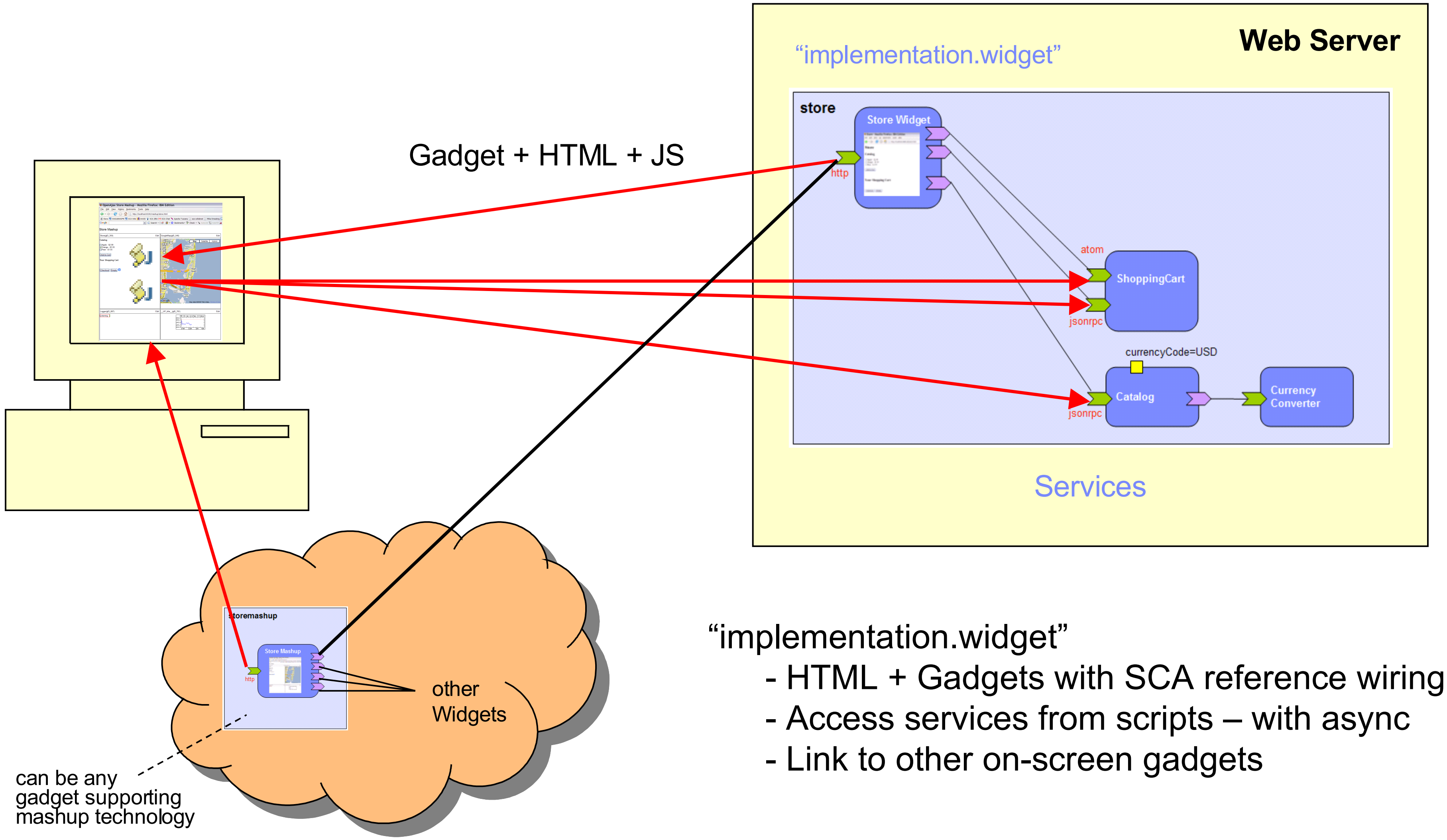


The Fruit Store Mashup

- Store Mashup - Offering the Store as an OpenAjax Widget



Web 2.0 Gadgets meet SCA

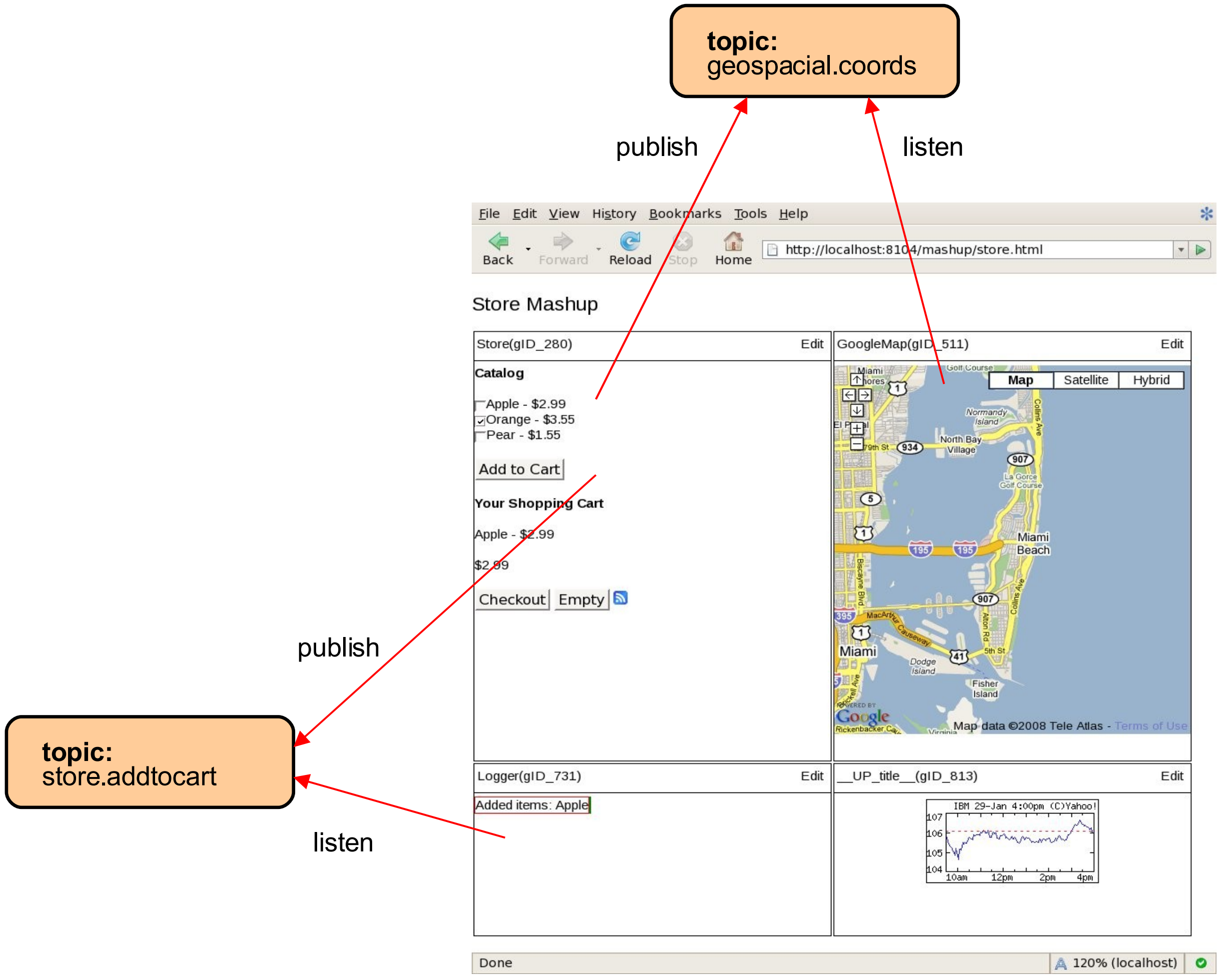


"implementation.widget"

- HTML + Gadgets with SCA reference wiring
- Access services from scripts – with async
- Link to other on-screen gadgets

Web 2.0 Gadgets meet SCA

Store Mashup - Offering the Store as an OpenAjax Widget



http://localhost:8104/mashup/store.html

Store Mashup

Store(gID_280)

Catalog

- Apple - $2.99
- ☑ Orange - $3.55
- Pear - $1.55

Add to Cart

Your Shopping Cart

Apple - $2.99

$2.99

Checkout Empty

GoogleMap(gID_511)

Logger(gID_731)

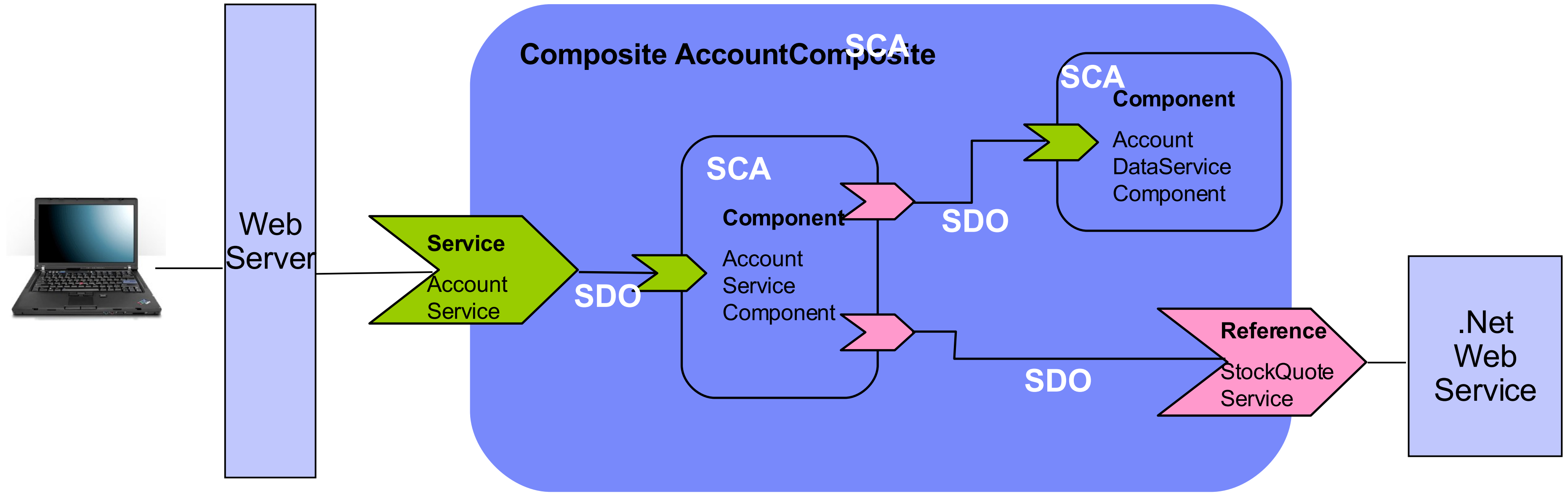
Added items: Apple

__UP_title__(gID_813)

Useful Links

Apache Tuscany	http://incubator.apache.org/tuscany/
- Tuscany SCA Java	http://cwiki.apache.org/TUSCANY/sca-java.html
- getting started	http://cwiki.apache.org/TUSCANY/getting-started-with-tuscany-release-10.html
- getting started using the eclipse dowloadable feature	http://jsdelfino.blogspot.com/2007/10/developing-sca-application-with-apache.html
- how to use with WAS 6.1	http://jsdelfino.blogspot.com/2007/10/how-to-use-apache-tuscany-with.html
SCA Specification Work	
- Intro to SCA by external consultant	http://www.davidchappell.com/articles/Introducing_SCA.pdf
- OASIS Open CSA	http://www.oasis-opencsa.org/
- V1 level specs	http://www.oasis-opencsa.org/sca
- Open CSA Technical Committees	http://www.oasis-opencsa.org/committees
- Webinars	http://www.oasis-opencsa.org/resources
- OSOA	http://osoa.org/display/Main/Home
- V1 level of specs	http://osoa.org/display/Main/Service+Component+Architecture+Specifications
- lots of other information e.g.	http://osoa.org/display/Main/SCA+Resources

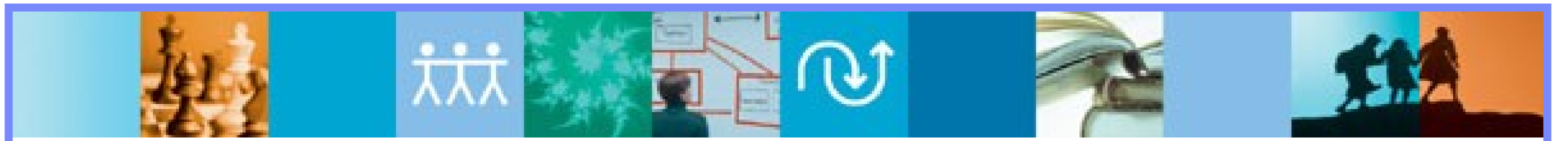
SDO helps deliver SCA implementation independence



- SDO representation transparency supports independence of client from services implementation technologies (XML web services, RDBMS, …)
  - SDO Data Access Services are SCA components
  - SDO DataObjects are the data on the wires

Observations and alternatives

- SCA base concepts are very “classic”
  - Components, Ports (in and out), Interfaces, Composites
  - You will find very similar concepts in CORBA component model or UML
- SCA adds more concrete detail for direct execution on a run-time
  - XML file formats
  - Bindings for various concrete technologies (e.g. web services)
  - Deployment model
- There aren’t a lot of obvious alternatives, just other versions of the same
  - Widget composition models are just SCA wannabes, not new
  - Other component models require concrete mappings (UML) or are technology-specific (CORBA)
  - OSGi declarative services model is very similar – by cooperation
- WCF
  - Not really a component model
  - Model for resolving end-points, but not recursive assembly
    - http://osoa.org/display/Main/SCA%20relationship%20with%20Windows%20Communication%20Foundation



Thank You

Martin Nally

nally@us.ibm.com