Getting Started with Tuscany

Ready, Set, Go - Getting started with Tuscany

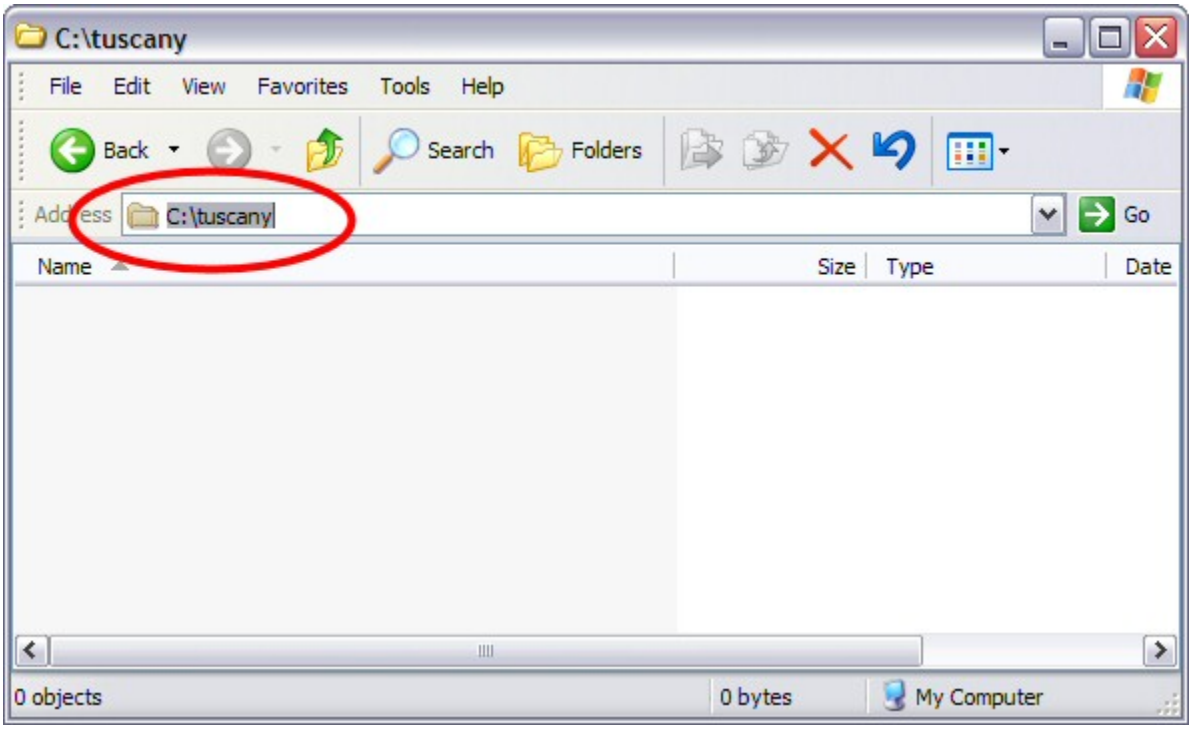
This article will show you how to download a Tuscany distribution, add the distribution as a user library in Eclipse, create the Store sample as a Java project, and then run the Store and use it from a web browser.

> Although this guide show some images based on the Java SCA 1.3.2 release, you should be able to use latest Java SCA release while going to the necessary steps for this guide.

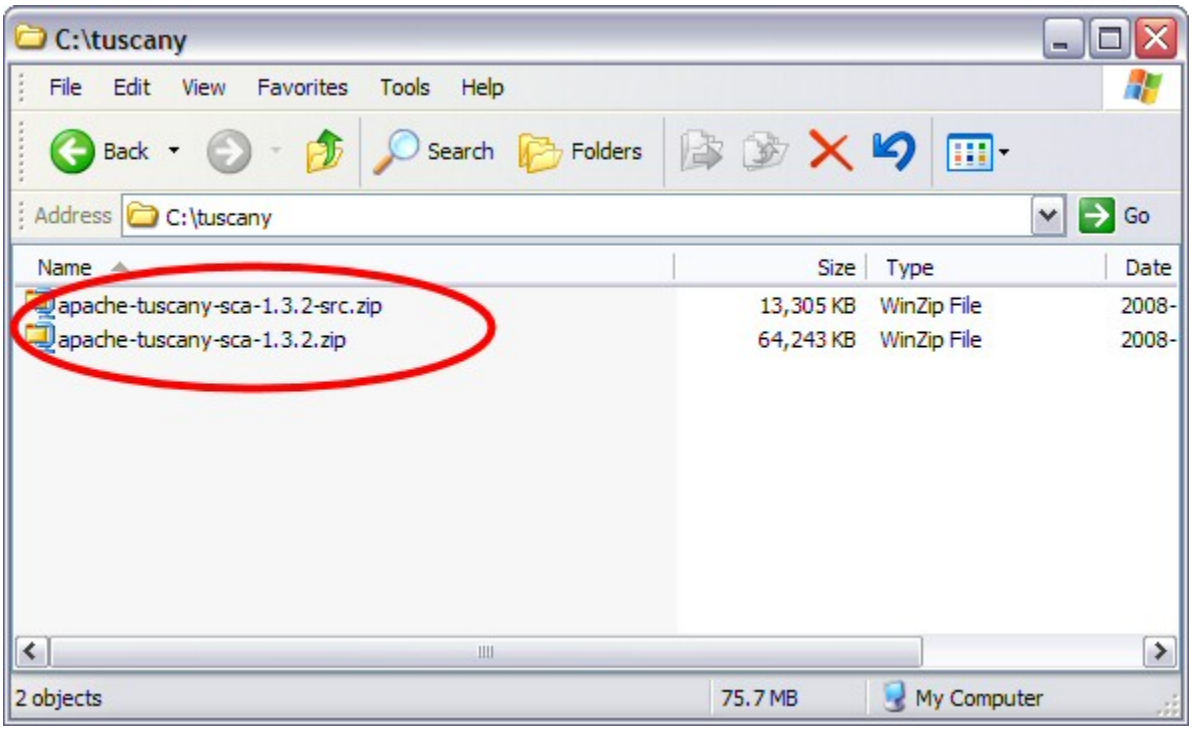
⭐ "Get Started with Store Demo in Eclipse video"

Install the Tuscany Distribution

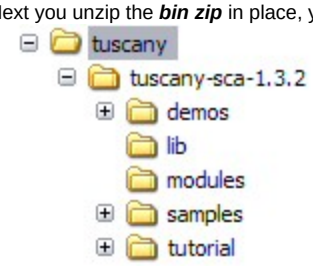
The first thing you do is to create a file system folder into which you will download the TUSCANY distribution.



Next you download the latest release distribution. Launch your browser and enter the following URL.
Latest Release - http://cwiki.apache.org/TUSCANY/sca-java-releases.html
Download both the ***bin zip*** as well as the ***src zip*** to the folder that you created on your disk. Once you completed the download you should see the following on your disk.



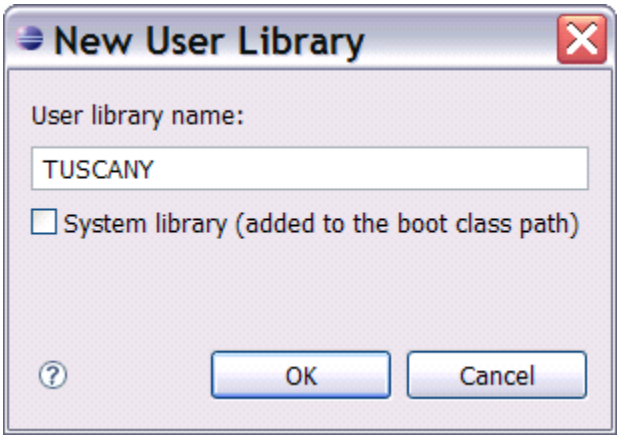
Next you unzip the ***bin zip*** in place, you should see the following folder file structure on your disk after unzip is complete.



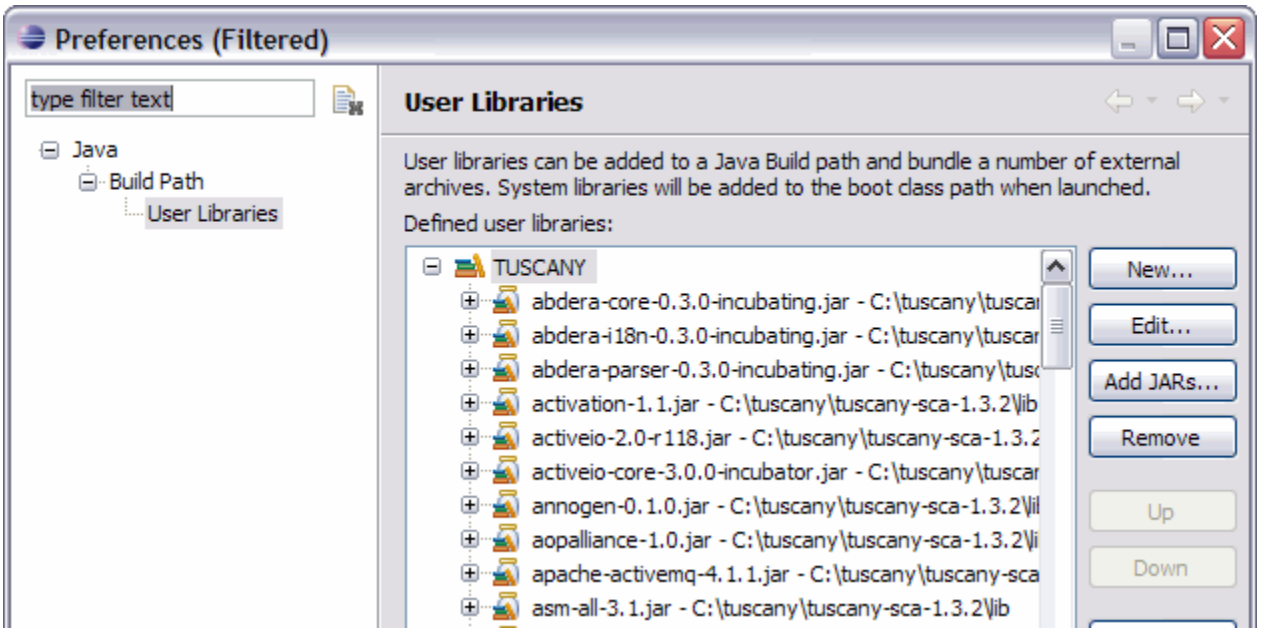
Setup Eclipse for Tuscany

Start Eclipse and create a User Library to contain the TUSCANY runtime jar's as well as their depending jar's.

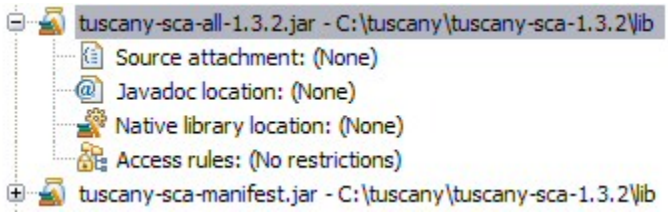
From the menu bar select ***Window*** and then ***Preferences...*** . The Preferences dialog will appear, in its left navigation tree select ***Java***, followed by ***Build Path***, and followed by ***User Libraries***. Select the ***New...*** pushbutton on the right of the New Libraries dialog to create a new user library.



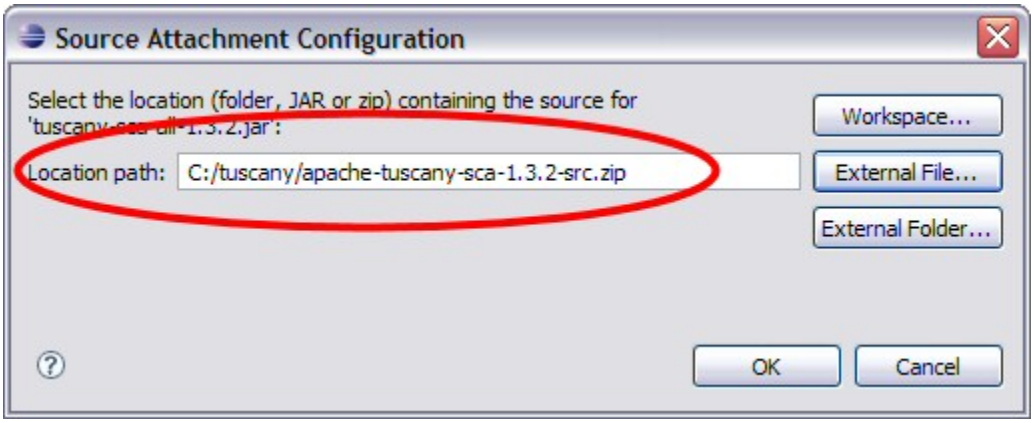
The user library created is empty, select the ***Add JARs...*** pushbutton on the right to add all the jar's from your Tuscany installation ***lib folder***. When completed all the jar's will appear under the TUSCANY user library.



Since some of you maybe interested in ***debugging*** also the Tuscany runtime code we will attach the Tuscany source to the Tuscany runtime jar in the following step. In the User Libraies dialog scroll down until you see the ***Tuscany runtime jar*** and select its ***Source attachment.***



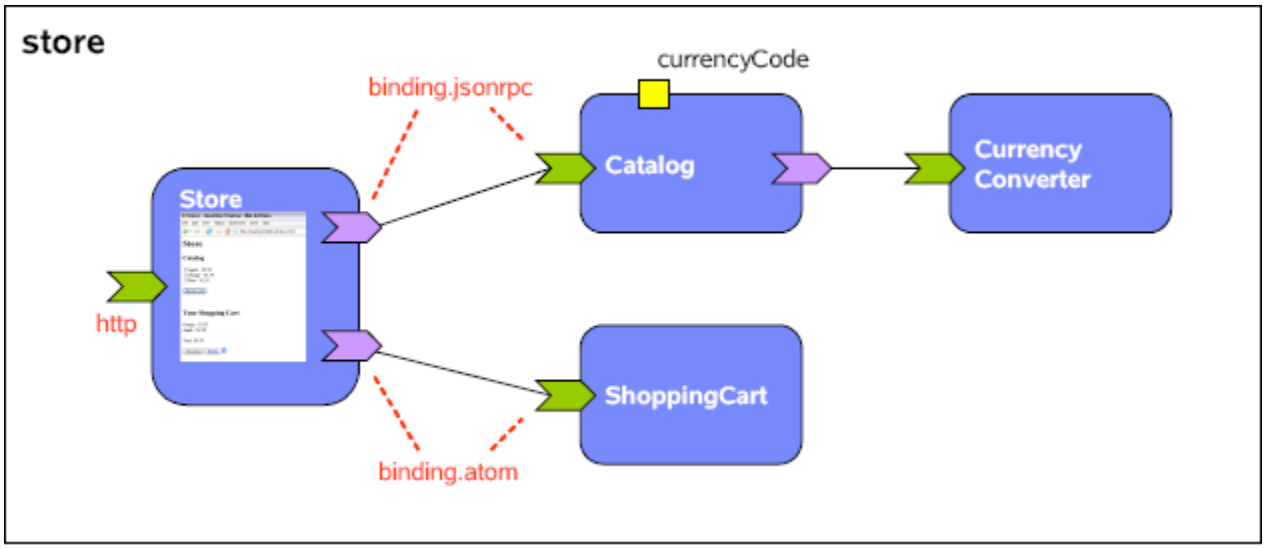
Select the ***Edit...*** pushbutton on the right and in the Edit dialog use the ***External File...*** pushbutton to the select the Tuscany ***src zip*** that we downloaded earlier.



Select ***OK*** to complete this and the Preferences dialog, and you are done with the Tuscany setup for Eclipse.

Create your 1st Composite Service Application

The following shows the composition diagram for the composite service application you are about to create.

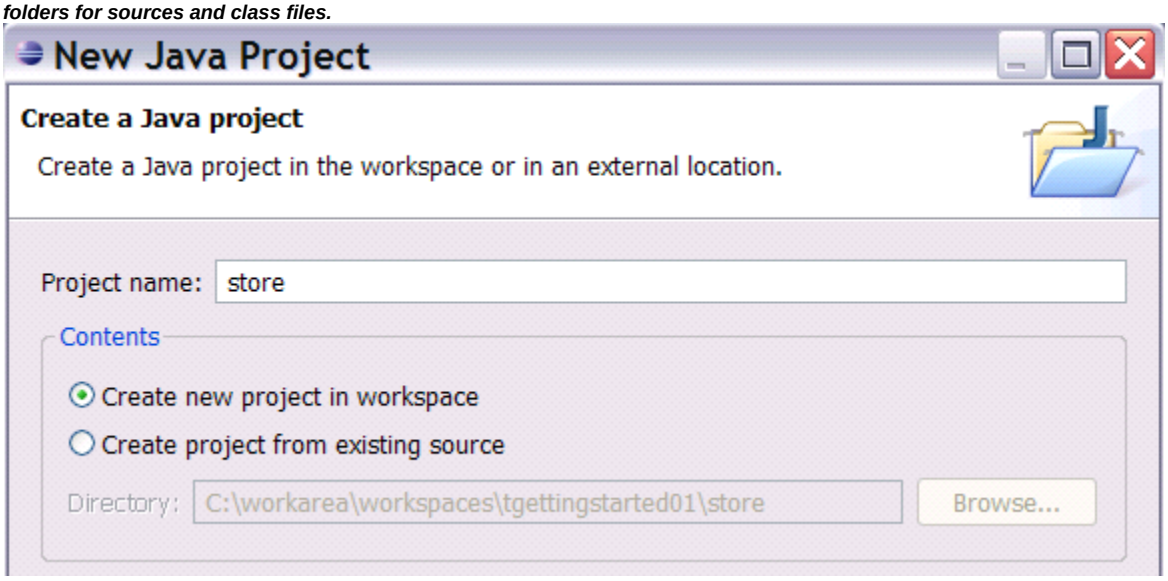


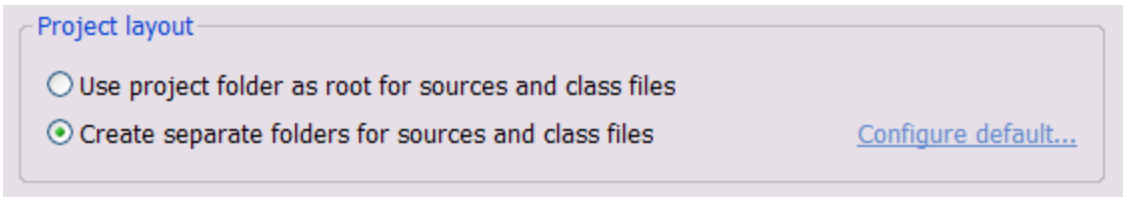
The composite service application you will create is a composition of four services. The composed service provided is that of an on-line store.
There is a Catalog service which you can ask for catalog items, and depending on its currency code property configuration it will provide the item prices in USD or EUR. The Catalog service is not doing the currency conversion itself it references a CurrencyConverter service to do that task. Then there is the ShoppingCart service into which items chosen from the catalog can be added, it is implemented as a REST service. The Catalog is bound using the JSONRPC binding, and the ShoppingCart service is bound using the ATOM binding. Finally there is the Store user facing service that provides the browser based user interface of the store. The Store service makes use of the Catalog and ShoppingCart service using the JSONRPC, and ATOM binding respectively.

Create a Java Project

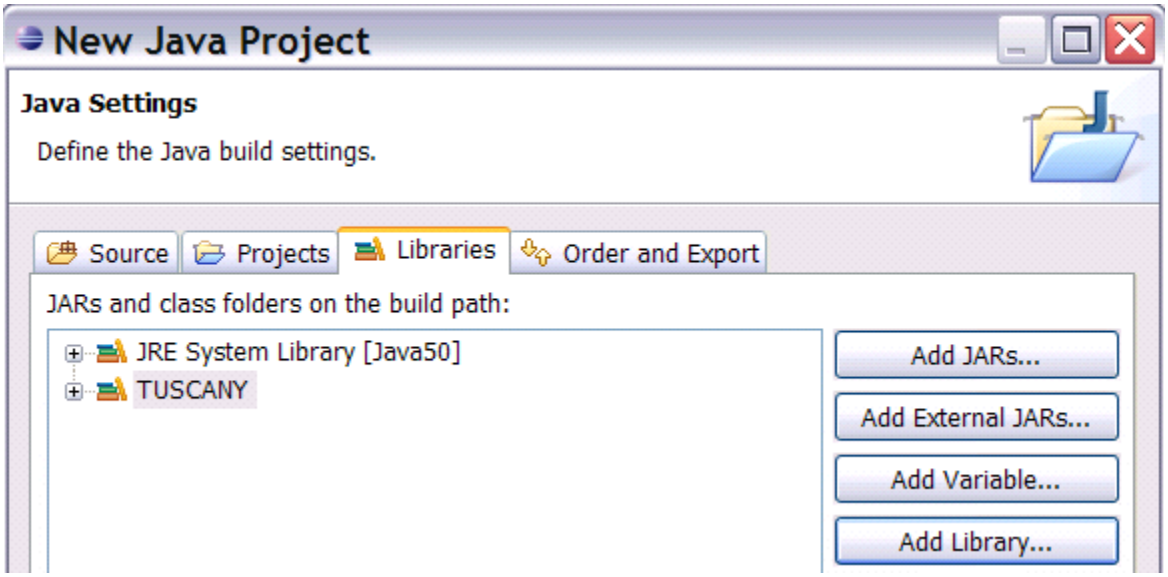
In this step you create a Java Project in Eclipse to hold the composite service application.

Click on the ***New Java Project*** button  in the toolbar to launch the project creation dialog. Next you enter "store" as the ***Project name***, and for ***Project Layout*** select ***Create separate folders for sources and class files.***

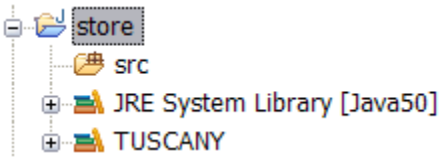




Hit the ***Next*** button, and on the following page go to the ***Libraries*** tab. Use the ***Add Library...*** button on the right to add the TUSCANY user library to the project.



Hit the ***Finish*** button to complete the ***New Java Project*** dialog to create the "store" java project.

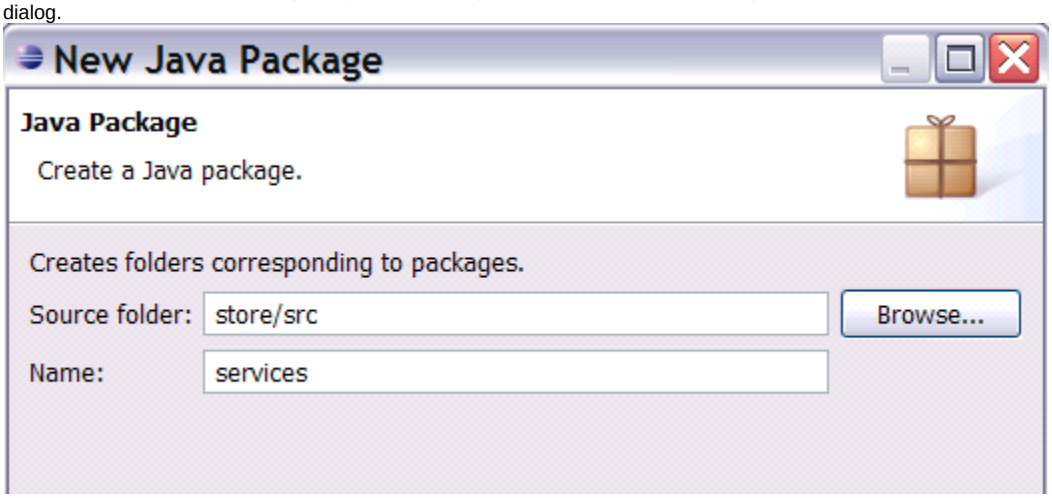


Construct Services

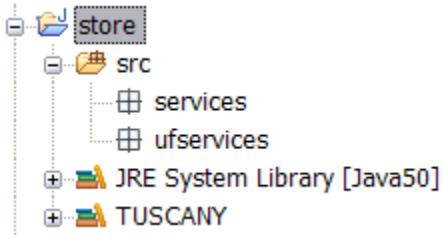
First you create two package folders into which later in this step you place service implementations.

Select the "store" project and click on the ***New Java Package*** button  in the toolbar to launch the package creation dialog.

Next you enter "services" as the package ***Name***, and press the ***Finish*** button to complete the dialog.



Repeat the previous step to create another package named "ufservices". The store project now should look as follows.



In the following you will place in the "services" package the regular services, and in the "ufservices" package the user facing services of the composite service application you create.

Catalog

In this step you create the Catalog service interface and implementation.
Select the "services" package. Next you click on the dropdown arrow next to the ***New Java Class*** button  and select the ***New Java Interface***  option from the dropdown list. In the dialog enter "Catalog" as the ***Name*** of the interface and select the Finish button to complete the dialog.
The Java editor will open on the new created Java interface. Replace the content of the editor by ***copy-paste*** of the following Java interface code snippet.

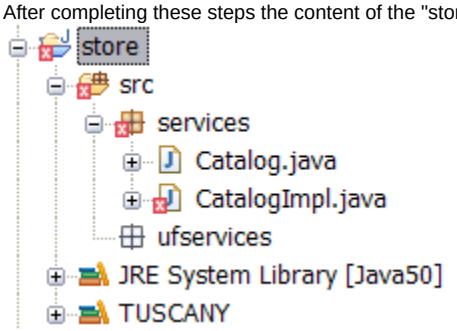
```
package services;

import org.osoa.sca.annotations.Remotable;

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

Select the "services" package again. Select the ***New Java Class*** button  . In the dialog enter "CatalogImpl" as the ***Name*** of the class, add "Catalog" as the interface this class implements, and then select ***Finish*** to complete the dialog.

The Java editor will open on the new created Java class. Replace the content of the editor by ***copy-paste*** of the following Java class code snippet.

```
package services;

import java.util.ArrayList;
import java.util.List;

import org.osoa.sca.annotations.Init;
import org.osoa.sca.annotations.Property;
import org.osoa.sca.annotations.Reference;

public class CatalogImpl 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)));
                catalog.add(new Item("Orange", currencySymbol
                                + currencyConverter.getConversion("USD", currencyCode, 3.55)));
                catalog.add(new Item("Pear", currencySymbol
                                + currencyConverter.getConversion("USD", currencyCode, 1.55)));
        }

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

After completing these steps the content of the "store" project will look as follows.



Note: CatalogImpl is red x'ed because it makes use of the CurrencyConverter interface that we have not implemented yet.

CurrencyConverter

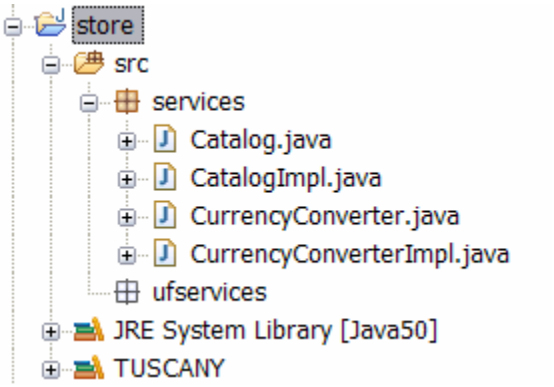
In this step you create the CurrencyConverter service interface and implementation.
You follow the same steps that you learned previously to create the interface and implementation.
First create a Java interface in the "services" package named "CurrencyConverter" and ***copy-paste*** the following Java interface code snippet into it.

```
package services;

import org.osoa.sca.annotations.Remotable;

@Remotable
public interface CurrencyConverter {
    public double getConversion(String fromCurrenycCode, String toCurrencyCode, double amount);

    public String getCurrencySymbol(String currencyCode);
}
```

Next create a Java class in the "services" package named "CurrencyConverterImpl" and ***copy-paste*** the following Java class code snippet into it.

```
package services;

public class CurrencyConverterImpl implements CurrencyConverter {
    public double getConversion(String fromCurrencyCode, String toCurrencyCode, double amount) {
        if (toCurrencyCode.equals("USD"))
            return amount;
        else if (toCurrencyCode.equals("EUR"))
            return ((double)Math.round(amount * 0.7256 * 100)) /100;
        return 0;
    }

    public String getCurrencySymbol(String currencyCode) {
        if (currencyCode.equals("USD"))
            return "$";
        else if (currencyCode.equals("EUR"))
            return "E"; //"â,¬";
        return "?";
    }
}
```

After completing these steps the content of the "store" project will look as follows.



ShoppingCart

In this step you create the Item model object, the Cart and Total service interfaces and the ShoppingCart service implementation. You follow the same steps that you learned previously to create the interface and implementation.

Create a Java class in the "services" package named "Item" and ***copy-paste*** the following code snippet into it.

```
package services;

public class Item {
	private String name;
	private String price;

	public Item() {
	}

	public Item(String name, String price) {
		this.name = name;
		this.price = price;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getPrice() {
		return price;
	}

	public void setPrice(String price) {
		this.price = price;
	}
}
```

Create a Java interface in the "services" package named "Cart" and ***copy-paste*** the following code snippet into it.

```
package services;

import org.apache.tuscany.sca.data.collection.Collection;
import org.osoa.sca.annotations.Remotable;

@Remotable
public interface Cart extends Collection<String, Item> {

}
```

Create a Java interface in the "services" package named "Total" and ***copy-paste*** the following code snippet into it.

```
package services;

import org.osoa.sca.annotations.Remotable;

@Remotable
public interface Total {
	String getTotal();
}
```

Create a Java class in the "services" package named "ShoppingCartImpl" and ***copy-paste*** the following Java class code snippet into it.

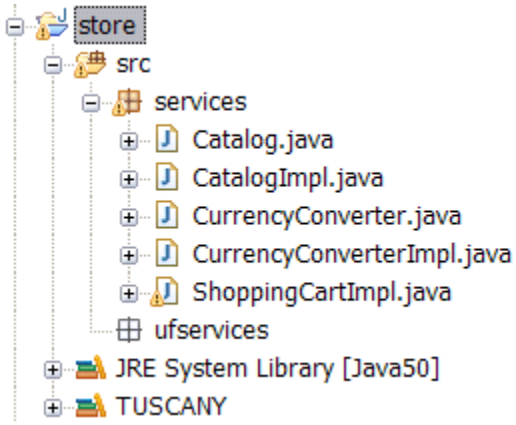
```
package services;

import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

import org.apache.tuscany.sca.data.collection.Entry;
import org.apache.tuscany.sca.data.collection.NotFoundException;
import org.osoa.sca.annotations.Init;
import org.osoa.sca.annotations.Scope;

@Scope("COMPOSITE")
public class ShoppingCartImpl implements Cart, Total {

        private Map<String, Item> cart;

        @Init
        public void init() {
                cart = new HashMap<String, Item>();
        }

        public Entry<String, Item>[] getAll() {
                Entry<String, Item>[] entries = new Entry[cart.size()];
                int i = 0;
                for (Map.Entry<String, Item> e : cart.entrySet()) {
                        entries[i++] = new Entry<String, Item>(e.getKey(), e.getValue());
                }
                return entries;
        }

        public Item get(String key) throws NotFoundException {
                Item item = cart.get(key);
                if (item == null) {
                        throw new NotFoundException(key);
                } else {
                        return item;
                }
        }

        public String post(String key, Item item) {
                if (key == null) {
                        key = "cart-" + UUID.randomUUID().toString();
                }
                cart.put(key, item);
                return key;
        }

        public void put(String key, Item item) throws NotFoundException {
                if (!cart.containsKey(key)) {
                        throw new NotFoundException(key);
                }
                cart.put(key, item);
        }

        public void delete(String key) throws NotFoundException {
                if (key == null || key.equals("")) {
                        cart.clear();
                } else {
                        Item item = cart.remove(key);
                        if (item == null)
                                throw new NotFoundException(key);
                }
        }

        public Entry<String, Item>[] query(String queryString) {
                List<Entry<String, Item>> entries = new ArrayList<Entry<String, Item>>();
                if (queryString.startsWith("name=")) {
                        String name = queryString.substring(5);
                        for (Map.Entry<String, Item> e : cart.entrySet()) {
                                Item item = e.getValue();
                                if (item.getName().equals(name)) {
                                        entries.add(new Entry<String, Item>(e.getKey(), e
```

```
                                                        .getValue()));
                                }
                        }
                }
                return entries.toArray(new Entry[entries.size()]);
        }

        public String getTotal() {
                double total = 0;
                String currencySymbol = "";
                if (!cart.isEmpty()) {
                        Item item = cart.values().iterator().next();
                        currencySymbol = item.getPrice().substring(0, 1);
                }
                for (Item item : cart.values()) {
                        total += Double.valueOf(item.getPrice().substring(1));
                }
                return currencySymbol + String.valueOf(total);
        }
}
```

Note: Since the Tuscany conversational support is not ready yet the cart is realized through a hack. The cart field is defined as static.

After completing these steps the content of the "store" project will look as follows.



Store

In this step you create the user facing Store service that will run in a Web browser and provide the user interface to the other services you created.

Select the "ufservices" package. ***Right click*** to get the context menu, select ***New***, and then ***File***. In the ***New File*** dialog enter "store.html" for the ***File name***, and then select ***Finish*** to complete the dialog.

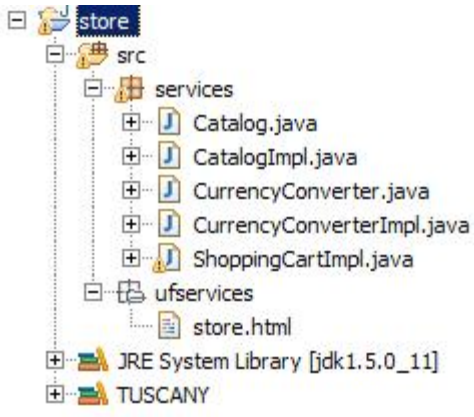
The Text editor will open on the new created html file. Replace the content of the editor by ***copy-paste*** of the following html snippet.

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

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

<script language="JavaScript">

        //@Reference
        var catalog = new tuscany.sca.Reference("catalog");

        //@Reference
        var shoppingCart = new tuscany.sca.Reference("shoppingCart");
```

```
        //@Reference
        var shoppingTotal = new tuscany.sca.Reference("shoppingTotal");

        var catalogItems;

        function catalog_getResponse(items) {
                var catalog = "";
                for (var i=0; i<items.length; i++) {
                        var item = items[i].name + ' - ' + items[i].price;
                        catalog += '<input name="items" type="checkbox" value="' +
                                                item + '">' + item + ' <br>';
                }
                document.getElementById('catalog').innerHTML=catalog;
                catalogItems = items;
        }

        function shoppingCart_getResponse(feed) {
                if (feed != null) {
                        var entries = feed.getElementsByTagName("entry");
                        var list = "";
                        for (var i=0; i<entries.length; i++) {
                                var content = entries[i].getElementsByTagName("content")[0];
                                var name = content.getElementsByTagName("name")[0].firstChild.nodeValue;
                                var price = content.getElementsByTagName("price")[0].firstChild.nodeValue;
                                list += name + ' - ' + price + ' <br>';
                        }
                        document.getElementById("shoppingCart").innerHTML = list;

                        if (entries.length != 0) {
                                shoppingTotal.getTotal(shoppingTotal_getTotalResponse);
                        }
                }
        }

        function shoppingTotal_getTotalResponse(total) {
                document.getElementById('total').innerHTML = total;
        }

        function shoppingCart_postResponse(entry) {
                shoppingCart.get("", shoppingCart_getResponse);
        }

        function addToCart() {
                var items  = document.catalogForm.items;
                var j = 0;
                for (var i=0; i<items.length; i++)
                        if (items[i].checked) {
                                var entry = '<entry xmlns="http://www.w3.org/2005/Atom"><title>item<
/title><content type="text/xml">' +
                         '<Item xmlns="http://services/">' +
                         '<name xmlns="">' + catalogItems[i].name + '</name>' + '<price xmlns="">' + catalogItems
[i].price + '</price>' +
                          '</Item>' + '</content></entry>';
                                shoppingCart.post(entry, shoppingCart_postResponse);
                                items[i].checked = false;
                        }
        }
        function checkoutCart() {
                document.getElementById('store').innerHTML='<h2>' +
                                'Thanks for Shopping With Us!</h2>'+
                                '<h2>Your Order</h2>'+
                                '<form name="orderForm">'+
                                        document.getElementById('shoppingCart').innerHTML+
                                        '<br>'+
                                        document.getElementById('total').innerHTML+
                                        '<br>'+

                                        '<br>'+
                                        '<input type="submit" value="Continue Shopping">'+
                                '</form>';
```

```
                shoppingCart.del("", null);
        }
        function deleteCart() {
                shoppingCart.del("", null);
                document.getElementById('shoppingCart').innerHTML = "";
                document.getElementById('total').innerHTML = "";
        }

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

</script>

</head>

<body onload="init()">
<h1>Store</h1>
  <div id="store">
           <h2>Catalog</h2>
           <form name="catalogForm">
                <div id="catalog" ></div>
                <br>
                <input type="button" onClick="addToCart()"  value="Add to Cart">
           </form>

         <br>

           <h2>Your Shopping Cart</h2>
           <form name="shoppingCartForm">
                <div id="shoppingCart"></div>
                <br>
                <div id="total"></div>
                <br>
                <input type="button" onClick="checkoutCart()" value="Checkout">
                <input type="button" onClick="deleteCart()" value="Empty">
                   <a href="../ShoppingCart/Cart/">(feed)</a>
        </form>
  </div>
</body>
</html>
```

After completing these steps the content of the "store" project will look as follows.



Compose Services

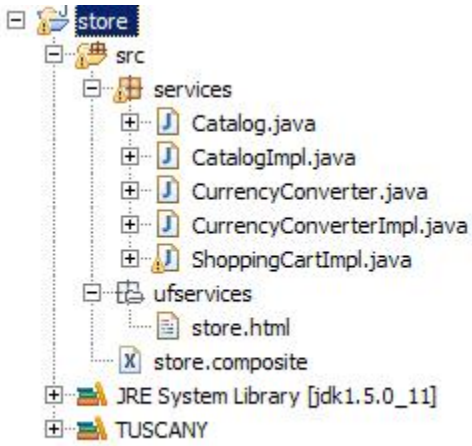
Now that you have all the required service implementations you compose them together to provide the store composite service. The composition is stored in a .composite file.

Select the "src" folder of the "store" project. ***Right click*** to get the context menu, select ***New***, and then ***File***. In the ***New File*** dialog enter "store.composite" for the ***File name***, and then select ***Finish*** to complete the dialog.

The Text editor will open on the new created composite file. Replace the content of the editor by ***copy-paste*** of the following composite snippet.

```
<?xml version="1.0" encoding="UTF-8"?>
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
        xmlns:t="http://tuscany.apache.org/xmlns/sca/1.0"
        xmlns:s="http://store"
        targetNamespace="http://store"
        name="store">

        <component name="store">
                <t:implementation.widget location="ufservices/store.html" />
                <service name="Widget">
                        <t:binding.http uri="http://localhost:8080/store" />
                </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="Catalog">
                <implementation.java class="services.CatalogImpl" />
                <property name="currencyCode">USD</property>
                <service name="Catalog">
                        <t:binding.jsonrpc uri="http://localhost:8080/Catalog"/>
                </service>
                <reference name="currencyConverter" target="CurrencyConverter" />
        </component>

        <component name="ShoppingCart">
                <implementation.java class="services.ShoppingCartImpl" />
                <service name="Cart">
                        <t:binding.atom uri="http://localhost:8080/ShoppingCart/Cart" />
                </service>
                <service name="Total">
                        <t:binding.jsonrpc uri="http://localhost:8080/Total"/>
                </service>
        </component>

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

After completing these steps the content of the "store" project will look as follows.



Launch Services

In this step you create the code to launch the Tuscany runtime with the new store composite service you created.

Select the "store" project and click on the ***New Java Package*** button  in the toolbar to start the package creation dialog. Use the dialog to create a new package named "launch".

Select the "launch" package. Select the ***New Java Class*** button  . In the dialog enter "Launch" as the Name of the class, check the checkbox for creating a main method stub, and then select Finish to complete the dialog.

The Java editor will open on the new created Java class. Replace the content of the editor by copy-paste of the following Java class code snippet.

```
package launch;

import org.apache.tuscany.sca.host.embedded.SCADomain;

public class Launch {
    public static void main(String[] args) throws Exception {
        System.out.println("Starting ...");
        SCADomain scaDomain = SCADomain.newInstance("store.composite");
        System.out.println("store.composite ready for big business !!!");
        System.in.read();
        System.out.println("Stopping ...");
        scaDomain.close();
        System.out.println();
    }
}
```

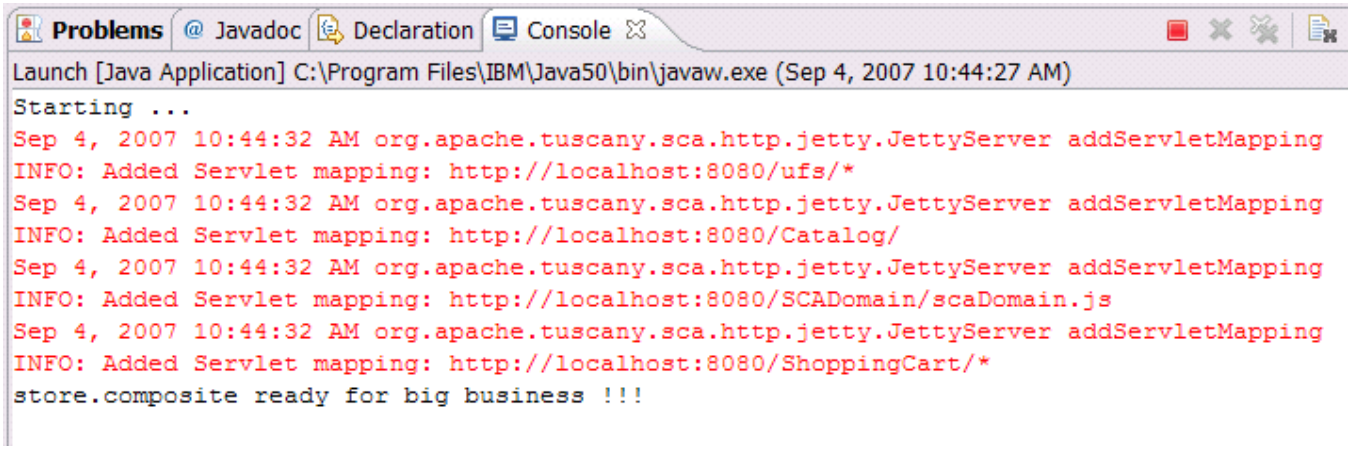
Congratulations you completed your 1st composite service applications, now its time to take it into action.

Use Services

In this step you launch and use the store composite service application you created.

First select the "Launch" class in the "launch" package of your "store" project. ***Right click*** to get the context menu, select ***Run As***, and then ***Java application***. The Tuscany runtime will start up adding the store composition to its domain.

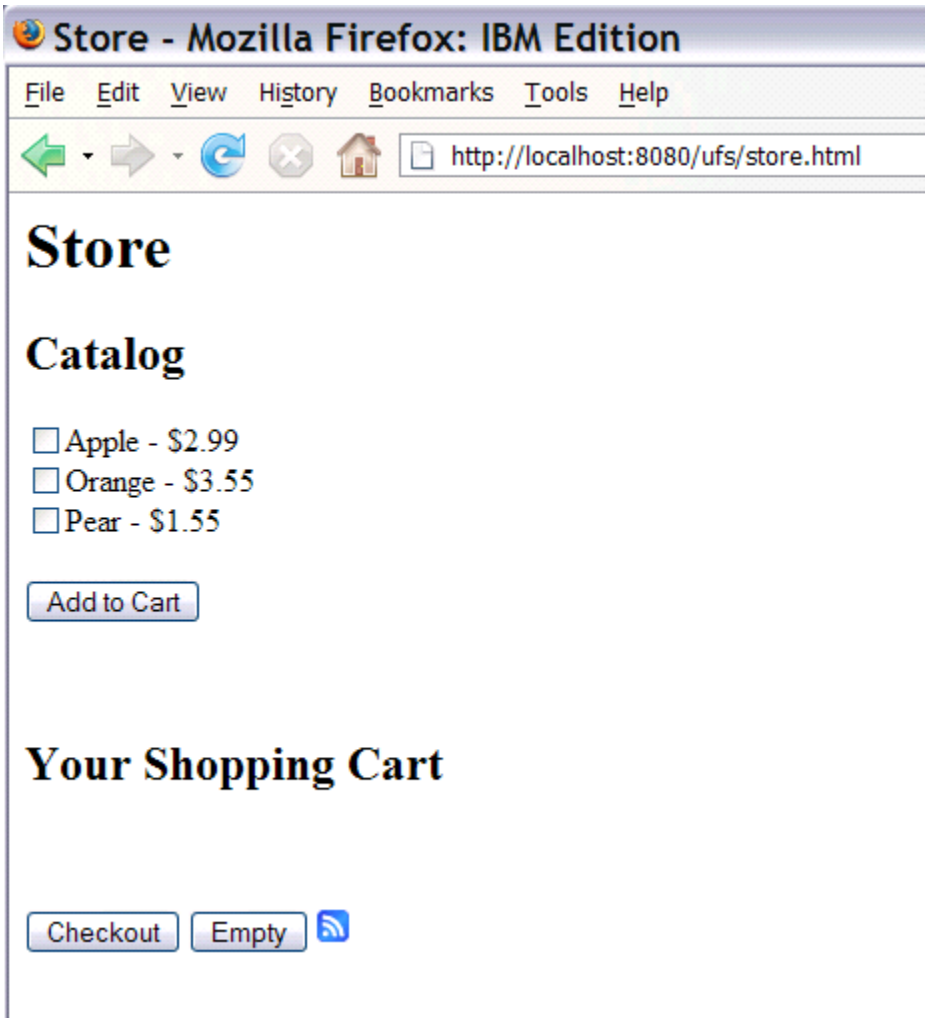
The Eclipse console will show the following messages.



Next Launch your Web browser and enter the following address:

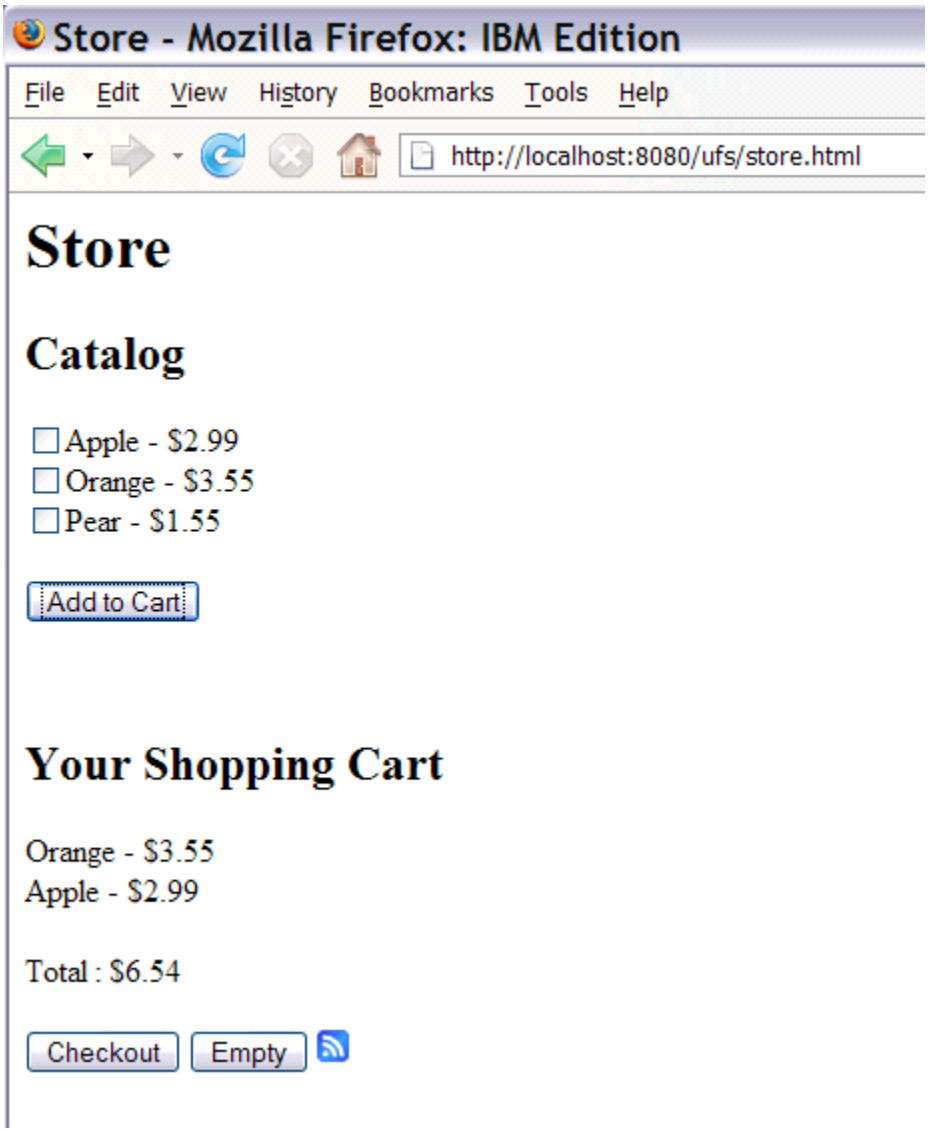
http://localhost:8080/store/store.html

You get to the Store user facing service of the composite service application.

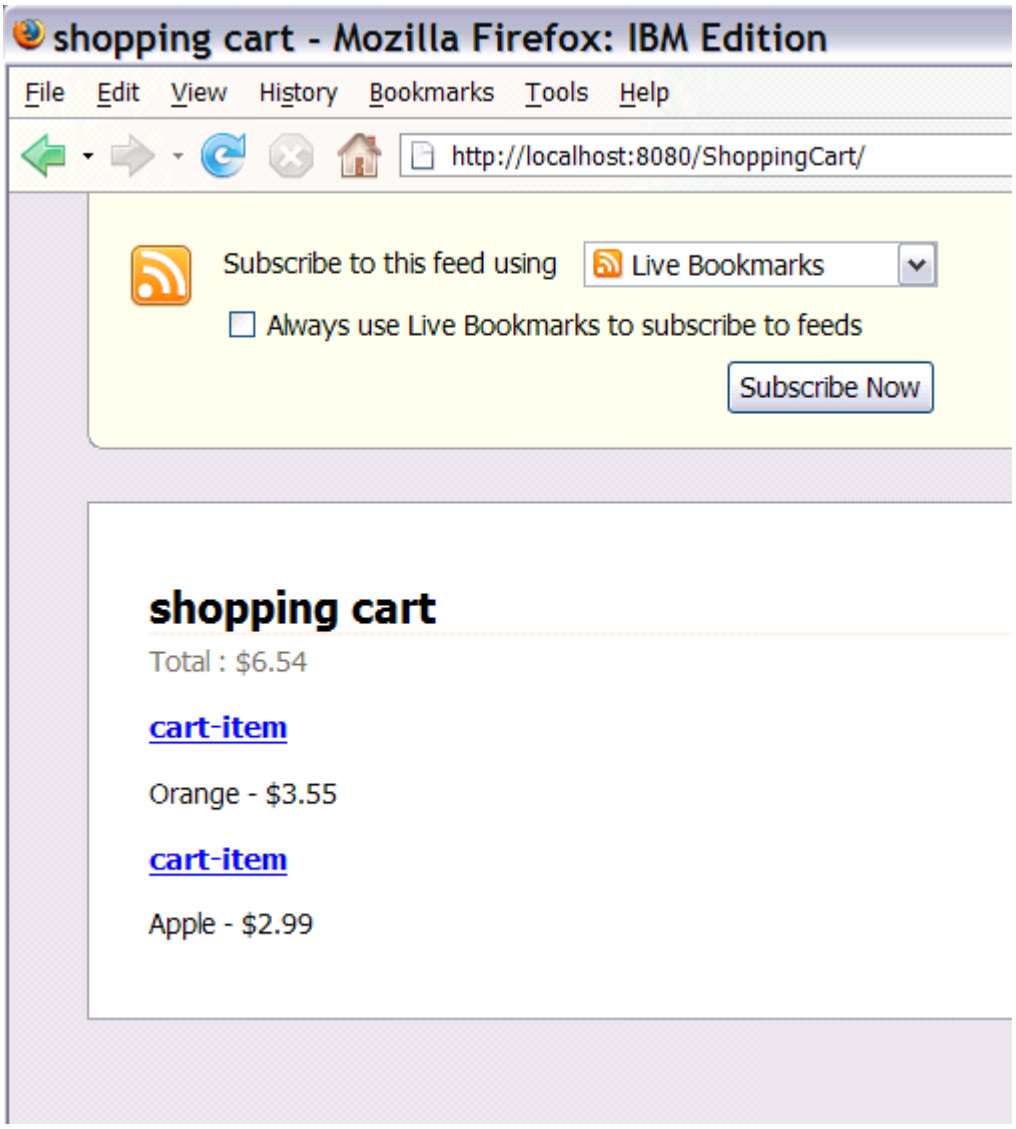


You can select items from the Catalog and add them to your Shopping Cart.

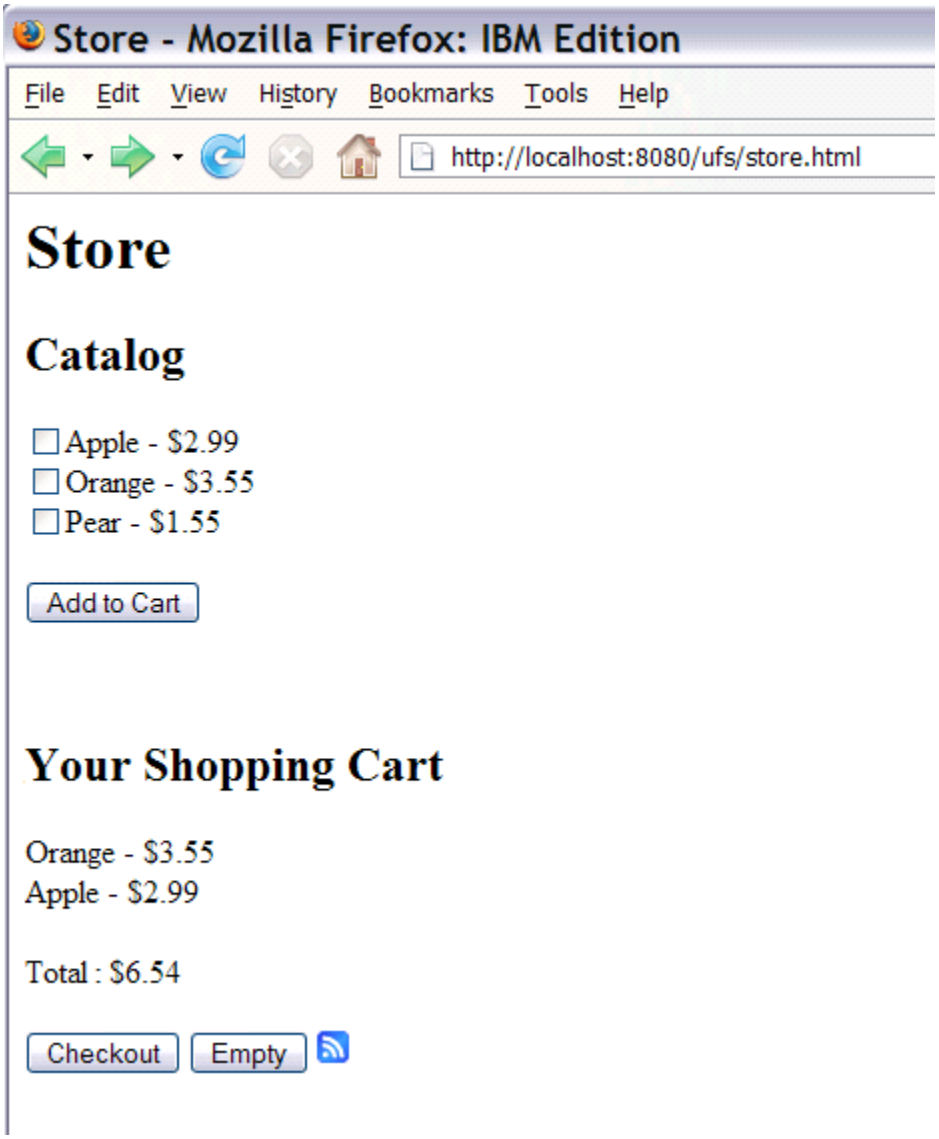
Note: When adding items for the first time you will be asked for ***userid and password*** by the browser. Enter "admin" for both.



Since the ShoppingCart service is bound using the ATOM binding, you can also look at the shopping card content in ATOM feed form by clicking on the feed icon . You get the browsers default rendering for ATOM feeds.



Use the browser back button to get back to the Store page.



And then you can Checkout to complete your order.

Congratulations, you have created an accessible, flexible, reusable Store application using Tuscany SCA technology.