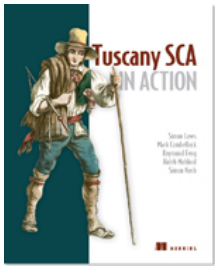

Tuscany SCA in Action

By Simon Laws, Mark Combellack, Raymond Feng, Haleh Mahbod, Simon Nash

In this article, based on Chapter 7 of Tuscany SCA in Action, the authors discuss bindings, which encapsulate the complexities of communication protocols and enable components to be implemented and wired together without a direct dependency on the communication protocols.

Introduction to SCA Bindings

Service Component Architecture (SCA) supports many communication protocols. In SCA terms, these are called bindings. Bindings encapsulate the complexities of communication protocols and enable components to be implemented and wired together without a direct dependency on the communication protocols used.

What does this mean in reality? It means that an SCA developer can implement a component without polluting the business code with details of how the communication between components will be handled. This separation of concerns has the added advantage of making the communication protocol a pluggable entity that can be changed at any time based on how and where the component is deployed. For example, suppose we want to make a Tuscany SCA application a web service. Simply, just add the SCA Web Services binding to the service defined in the composite XML file. Suppose we then want to make the Tuscany SCA application an RMI service as well. Simply, just add the SCA RMI binding to the service defined in the composite XML file. Both of these changes can be done without modifying the component implementation code for the application.

Using SCA bindings

SCA bindings can be added to a composite application either on an SCA service or on an SCA reference. The following subsections provide a quick overview of these two usage patterns.

Using SCA bindings on an SCA service

One or more SCA bindings can be used on a service to allow the service to be invoked over each binding's protocol. To illustrate this, listing 1 shows the payment composite with an SCA Web Services binding and an SCA RMI binding on the `Payment` service.

Listing 1 `Payment` composite XML file with web service and RMI binding

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
           targetNamespace="http://scatours"
           name="payment">
    <component name="Payment">
        <implementation.java class=
            "com.tuscanyscatours.payment.impl.PaymentImpl" />
        <service name="Payment">
            <interface.java interface="payment.Payment" />
            <binding.ws />                  #1
            <binding.rmi />                #2
        </service>
    </component>
</composite>
```

#1 SCA Web Services binding
#2 SCA RMI binding

By adding the Web Services binding (#1) and the RMI binding (#2), the `Payment` service can be invoked over both SOAP and RMI.

Using SCA bindings on an SCA reference

SCA references describe the dependencies that an SCA component has on other services. Multiple bindings can be assigned to a single reference, each enabling communication over a different type of protocol. Listing 2 shows how the payment component can invoke the `CustomerRegistry` service over the Web Services binding and `CreditCardCORBAService` over the CORBA binding.

Listing 2 `Payment` composite with various SCA bindings on the references

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
           targetNamespace="http://scatours"
           name="payment">
   <component name="Payment">
      <implementation.java class=
          "com.tuscanyscatous.payment.impl.PaymentImpl" />
      <reference name="customerRegistry">
         <binding.ws                                              #1
            uri="http://someserver/CustomerRegistry"/>            #1
      </reference>
      <reference name="creditCardPayment">
         <binding.corba                                           #2
           host="someserver" port="5080"                          #2
           name="CreditCardCORBAService"/>                        #2
      </reference>
      <reference name="emailGateway"
                 target="EmailGatewayComponent"/>
  </component>
</composite>
```

#1 Web Services binding
#2 CORBA binding

The `customerRegistry` reference on the `Payment` component has a Web Services binding (#1). This is indicated by the `<binding.ws>` element, which specifies the location of the customer registry web service using the `uri` attribute.

The `creditCardPayment` reference on the `Payment` component has a CORBA binding (#2). This is indicated by the `<binding.corba>` element, which specifies the host and port number as well as the CORBA service name of the credit card payment CORBA service.

Looking at the `emailGateway` reference, did you notice that no binding is listed? So, what SCA binding is used for this reference? The answer is that, if a service or reference doesn't specify a binding, then the SCA default binding is used.

Now, let's spend a few moments introducing the SCA components that can be used to demonstrate the SCA bindings.

Demonstrating SCA bindings

We'll demonstrate how to use bindings on services with a currency converter component and on references with a notification component.

Overview of the currency converter

The currency converter is a simple component that can be used to calculate the value of one currency in a different currency. Figure 1 shows an overview of the currency converter.

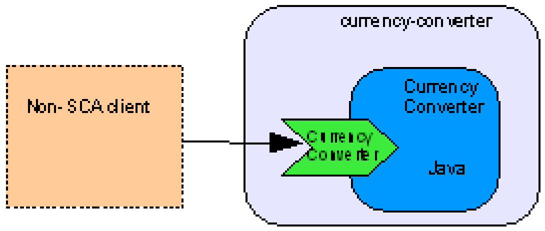

Figure 1 The currency converter with its single `CurrencyConverter` service

The composite definition for the currency converter SCA component can be found in the `contributions/currency` directory of the SCA Tours application and is shown in listing 3.

Listing 3 The currency converter composite XML file

```
<composite name="currency-converter" ...>
    <component name="CurrencyConverter">
        <implementation.java class= "com.tuscanyscatours.
[CA]currencyconverter.impl.CurrencyConverterImpl" />
        <service name="CurrencyConverter">
            <interface.java interface= "com.tuscanyscatours.
[CA]currencyconverter.CurrencyConverter" />
        </service>
    </component>
</component>
```

As can be seen from the above XML, the currency converter is implemented using the Java implementation type. It exposes a single service that's defined by the `CurrencyConverter` Java interface.

Listing 4 The currency converter Java interface

```
@Remotable
public interface CurrencyConverter {
    double getExchangeRate(String fromCurrencyCode,
                           String toCurrencyCode);
    double convert(String fromCurrencyCode, String toCurrencyCode,
                   double amount);
}
```

Listing 4 shows the Java interface that defines the methods on the currency converter service. It has two methods, `getExchangeRate` and `convert`.

The currency converter launcher in the `launchers/currency-converter` directory of the SCA Tours application can be used to test the currency converter. When it's run it will load the currency converter contribution into Apache Tuscany and perform a quick test producing the following output:

```
Quick currency converter test
USD -> GBP = 0.5
100 USD = 50.0GBP
```

As we haven't specified any bindings on the services or references, Tuscany will use the SCA default binding to wire the components together.

Overview of the notification service

The notification service is a simple component that can be used as part of an SCA application to send notifications to users. For the SCA Tours application, it could be used to send notifications such as "credit card payment accepted". The notification service acts as a façade that hides how the actual notification is sent to the user. In this basic implementation, the notification service sends notifications via SMS to the user's mobile phone. The notification service doesn't actually contain the code to send the SMS. To send the SMS, it invokes an external SMS gateway service. Figure 2 shows an overview of the notification service.

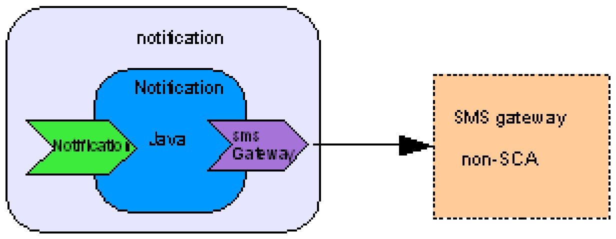

Figure 2 Invoking the notification service will cause it to invoke the SMS gateway web service

The composite definition for the notification component can be found in the contributions/notification directory of the SCA Tours application and is shown in listing 5.

Listing 5 The notification composite XML file

```
<composite name="notification" ...>
    <component name="Notification">
        <implementation.java class="com.tuscanyscatours.notification.
[CA]impl.NotificationImpl" />
        <service name="Notification">
            <interface.java  interface="com.tuscanyscatours.
[CA]notification.Notification"/>
        </service>

        <reference name="smsGateway">
            <interface.java
               interface="com.tuscanyscatours.smsgateway.SMSGateway"/>
            ...
        </reference>
    </component>
</composite>
```

The notification service is implemented using the Java implementation type. It exposes a single service that's defined by the `Notification` Java interface. It also has a reference to the SMS gateway.

The `Notification` Java interface (shown below) is used to define the methods on the notification service.

```
public interface Notification {
    boolean notify(String accountID, String subject, String message);
}
```

It has a single method called `notify` that will:

1. Look up the mobile phone number associated with the specified account ID.
2. Send an SMS to the user's mobile phone that contains the specified subject and message.

The notification service assumes that there is a mobile phone number associated with the specified account ID.

The SMS gateway is a non-SCA application that exposes a service that can be used to send SMS messages to mobile phones. It represents an existing business application not written using SCA, which we want to integrate into our SCA application.

The interface of the service that's exposed by the SMS gateway is shown below.

```
public interface SMSGateway {
    boolean sendSMS(String fromNumber, String toNumber, String text);
}
```

The `SMSGateway` Java interface for the SMS gateway has a single method called `sendSMS` that will send an SMS message containing the given text to the specified number. We don't really send SMS messages from our sample code but simply write the message that would be sent to the console.

Summary

We've seen how, by using SCA bindings, we can expose an SCA service and access external services using communication protocols. The binding approach provides a clean separation between the application logic contained with the component implementation and the communication protocols used to wire them together.

Last updated: February 16, 2011