Forms goes SOA
Jüssen Stefan . Senior Consultant . 25.10.2010

Any change in business processes must immediately be reflected in the supporting software. Professional system development draws on proven process models and state-of-the-art engineering methods, frameworks and architecture blueprints.

With its Fusion Middleware Products Oracle delivers such a framework. But can Oracle Forms achieve these criteria’s? Can applications be changed rapidly and simple? Simple in this context means, to achieve the companies’ investments they have made in their existing applications [Liebhart, Daniel].

The following article describes how to integrate new functionality with Oracle Forms using solutions based on Service Oriented Architecture (SOA) to build a more universal system at any time.

1 How can Forms and SOA be brought into contact?

Why only a loose coupling of services and not directly calling? Numerous "how to’s" describe how Web services can be called directly from different versions of Oracle Forms. For example services can be called from Oracle Forms 11g, which is specified in [Oracle, Webservice].

Common to all is the following approach:

- Create a web Service
- Create a Web Service Proxy
- Deploy the Web Service Proxy
- Prepare the Oracle Forms Module to invoke the Web Service
  - Set the classpath and import the Java Code
- Invoke the imported Java Method

Much trouble for a simple service and the approach runs counter to the principle of accommodating as possible no logic to the presentation layer.

Other possibilities to integrate Web Services are:

- Use of „external events“ over „advanced queuing“
- Use of database adapter (inbound / outbound)

Which of these methods can be useful, could be decided on the basis of the interaction between the user and the application. Three possible interaction patterns, “One way call using database adapters”, “Synchronous call using utl_http” and “Asynchronous call using advanced queuing”, are shown below. They show how to use services by minimizing interference with the existing application. See also [Schmutz, Guido]: Oracle Service Bus vs. Oracle Enterprise Service Bus vs. BPEL.

2 One way call

In a one-way message, or fire and forget, the client sends a message to the service, and the service does not need to reply. A typical example for such a scenario is an order process. An order will not necessarily be processed in seconds. Thus, for the integration, a database adapter could be
used, that identifies incoming orders every minute and causes the order to be executed, triggered by an existing Forms application that stores the order data in the database.

2.1 A sample order process

2.1.1 Old process flow:

The delivery process is performed manually. After shipping, an email is sent. The user has no information about the detailed process progresses (Picture 1).

![Diagram of old process flow]

Picture 1: Old process flow
2.1.2 Modified process flow:

This type of service has no impact to Oracle Forms because it is controlled by a database adapter, which polls for incoming orders.

In the modified process flow (Picture 2) more emails are sent to the user to be informed about the progress. Thus, it lends itself to this process to implement an information service, which is controlled by parameters. One advantage of such a central service is the future expansion and general usability. SMS could be an alternative or supplement to send emails. The SOA Suite offers numerous possibilities.

Picture 2: Modified process flow
2.2 Information service

The information service is controlled by two parameters. An order number and the email type are sufficient that the service itself can raise the remaining data. The database function "GetEmailDetails" does this and then sends the email (Picture 3).

![Information service diagram](image3.png)

Picture 3: Information service

2.3 Order process flow

A BPEL process controls the process flow and makes an order service with human task available (Picture 4).

![Order service diagram](image4.png)

Picture 4: Order service
2.4 Human task

The human task allows orders to confirm or to reject (Picture 5). If an order cannot be executed, the customer receives a message. Delivered orders can also be identified by sensors on the dispatch department and the clerk will be able to focus entirely unfulfilled orders.

![Human task diagram](image)

Picture 5: Human task

2.5 Main ordering process flow

The main process polls for new orders (Picture 6), which are written to the database and invokes the "Order Service". If this service is published throughout the company, he can of course also be used from other applications.

![Main ordering process flow diagram](image)

Picture 6: Main ordering process
3 Synchronous call

In a synchronous process, the client sends a request to a service and waits for a response. A typical example of this scenario is a reservation (flight reservation, hotel booking).

3.1 A sample booking process

From Oracle Forms a booking service is most easily accessed through a database procedure, which handles the service and returns the result. The service expects the parameters `<HotelName>`, `<CreditCardType>` and `<CreditCardNumber>`.

3.2 Booking service

A BPEL process invokes a database routine that does the actual processing (Picture 7).

![Booking service](image)

**Picture 7:** Booking service

3.3 BPEL detail

A synchronous service is called using the `<invoke>` activity and transfers the result to the output message (Picture 8).

![Detailed BPEL flow](image)

**Picture 8:** Detailed BPEL flow
3.4 Invoking the service from a database function

A simple way to call a Web service from the database is to use the package utl_http. The result is a XMLTYPE and must be evaluated more appropriately.

```
FUNCTION book_hotel (p_hotel VARCHAR2, p_card VARCHAR2, p_card_no VARCHAR2)
IS
  resp XMLTYPE;
BEGIN
  soap_req :=
    '<soapenv:Envelope .... ' || '<soapenv:Body>'
    || ' <bpel:process>'
    || ' <bpel:hotel>' || p_hotel || '</bpel:hotel>'
    || ' <bpel:creditcard>' || p_card || '</bpel:creditcard>'
    || ' <bpel:card_no>' || p_card_no || '</bpel:card_no> '  
    || '</bpel:process>'
    || '</soapenv:Body>'
  http_req := UTL_HTTP.begin_request ('http: ... ', 'POST', 'HTTP/1.1');
  UTL_HTTP.set_header (http_req, 'Content-Type', 'text/xml');
  UTL_HTTP.set_header (http_req, 'Content-Length', LENGTH (soap_req));
  UTL_HTTP.set_header (http_req, 'SOAPAction', 'process');
  UTL_HTTP.write_text (http_req, soap_req);
  http_resp := UTL_HTTP.get_response (http_req);
  UTL_HTTP.read_text (http_resp, soap_resp);
  resp := XMLType.createXML (soap_resp);
  resp := resp.EXTRACT ('/env:Envelope/env:Body/node()','
     xmlns:env="http ... "');
END;
```

3.5 Call from Oracle Forms

Finally, the above-mentioned function is used in a Forms module.

```
DECLARE
BEGIN
  message (  
    booking.book_hotel ('Hotel California', 'Visa', '1234-1234-1234-1234'));
END;
/

Result: OK
```
4 Asynchronous call

For asynchronous requests, the client calls a service and expects a response. Till it comes, the application can continue working (Picture 9). Representatives of this scenario are long lasting measures, such as a Fourier analysis. The shutter button initiates the calculation, is working on and would like to be informed of the outcome at the end of the calculation.

Picture 9: Asynchronous call

Oracle Forms 11 can subscribe and react to external events that are published to the database’s Advanced Queuing (AQ) feature.

Oracle Forms initiated the process by writing in a queue. If the calculation has been completed, the "WHEN-EVENT RAISED" trigger that is associated with the event fires. The application can view the result of the calculation to the user.

Since many other technologies can publish events to AQ (examples include JMS and BPEL), Forms can now interact with those technologies in an asynchronous manner. The Oracle Forms applications can react to events from outside Forms, as well as interact with other Forms applications.
5 Conclusion

Particularly in heterogeneous environments services offer a great way to provide commonly available functionality.

Have fun serving your services and success in the use of Trivadis know-how.

Stefan Jussen
Trivadis GmbH
Industriestrasse 4
D-70565 Stuttgart
Tel: +49-711-903 63 230
Fax: +49-711-903 63 259
Mail: info@trivadis.com

6 Literature and Links...

Liebhart, Daniel

Schmutz, Guido

Oracle, Webservice