Streamline business processes and increase user productivity with SAP NetWeaver

Build forms-based Web Dynpro applications using Interactive Forms based on Adobe software

by Markus Meisl and Marc Chan

Markus Meisl
SAP NetWeaver
Product Manager,
Forms Technology,
SAP AG

Marc Chan
Senior Consultant,
SAP NetWeaver Regional
Implementation Group (RIG),
Asia-Pacific Region,
SAP Hong Kong

(Full bios appear on page 50.)

This scenario is probably all too familiar — after a business trip, you log on to the SAP R/3 system, go to transaction PR05 (Travel Expense Manager), and then work through the tedious task of trying to piece together what you had for breakfast on the third day of the trip, if you ate at all, or on which day you took a cab from the convention center back to the hotel. Imagine the time (and headaches) you could save if you could record your expenses on the road and simply upload them to the system upon your return to the office.

This is but one of many scenarios for which users would benefit from an offline data-entry capability for SAP. Others include:

- Field staff who do not always have direct system access — for example, a sales representative who is at a customer site and needs to enter data for an order.
- Casual SAP users who appreciate simplified user interfaces — for example, a factory worker who has extremely limited authorization to the back-end system and needs to enter an order for supplies.
- External business partners who provide vital information to the system — for example, an external vendor-managed inventory supplier who needs to monitor certain aspects of a business, but for security reasons does not have direct system access.

Until now, users required direct system access to carry out such daily business tasks. Fortunately, SAP NetWeaver ’04 offers a new technology called Interactive Forms, included with SAP Web Application Server (SAP Web AS) 6.40, that changes all this. This new technology is based on Adobe’s widely accepted Portable Document Format (PDF) technology, and enables users to take parts of their business processes offline. With Interactive Forms functionality, end users can:

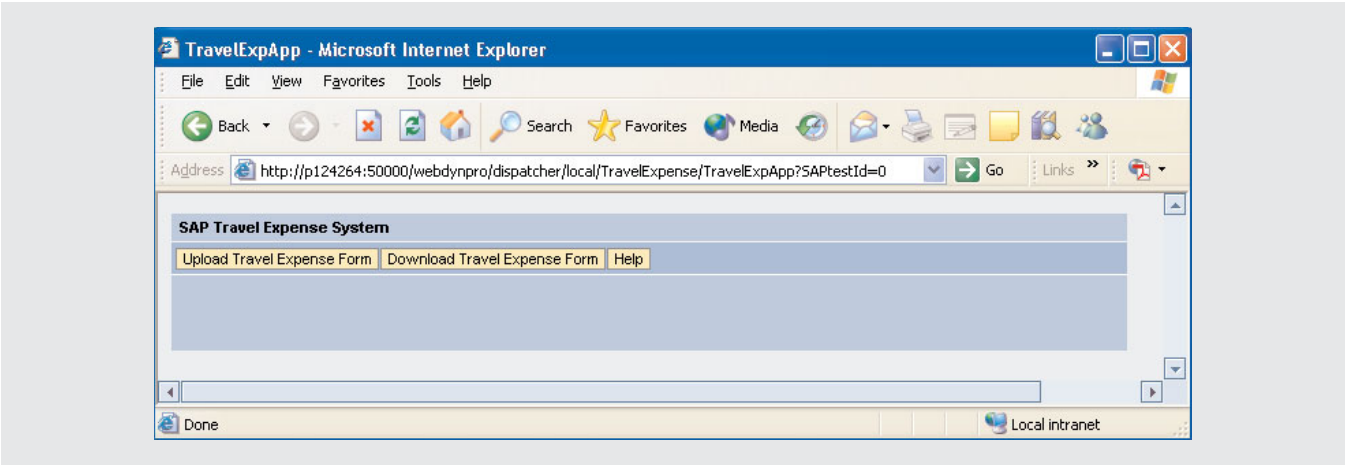


Figure 1 The sample Web Dynpro application

- View forms and fill in form fields online, and with a connection to the back-end system immediately synchronize the entered data with the back-end data.
- View forms and fill in form fields offline on a local computer, and then later synchronize the entered data with the back-end data when a connection to the back-end system is established.
- Fill in forms with specific data from back-end SAP applications, and transmit the data to intended recipients (e.g., HR personnel can fill in a form with company-sponsored benefits information from the back end and send it to employees).
- Transmit completed forms back to the SAP application, updating the business data automatically.

On the development side, Interactive Forms enables you to:

- Create form templates, which can include text fields, data fields, graphics, etc., enabling you to consistently address technical requirements, form appearance, functionality, etc., for data entry both online and offline.
- Provide simple checks on required fields (e.g., check whether a date has been entered in a required field) or format data without coding.
- Add programmatic validation when logical or complex validations are required (e.g., check that required calculations have been performed).
- Add functions, such as inserting comments or digital signatures, so that business needs for collaboration and security are met.

In SAP NetWeaver ’04, Interactive Forms is integrated into the Web Dynpro for Java environment (see the sidebar on page 30 for more details on the Interaction Forms architecture). But keep reading even if you are an ABAP developer: SAP NetWeaver 2004s will bring the integration of Interactive Forms into Web Dynpro for ABAP, and this article teaches you all the principles you’ll eventually be able to apply.

This article demonstrates how to build and deploy a custom data-entry form using Interactive Forms within a Web Dynpro for Java application (see **Figure 1**). We do not cover all the capabilities of Interactive Forms listed previously; instead, we focus on the basics so that you have a solid foundational knowledge to build upon as you develop your own integrated forms applications. We will show you how to use Interactive Forms technology to enable remote users to record and store travel expenses locally (i.e., on a laptop) as a PDF document, and then submit the data to the back-end SAP R/3 system at the click of a button when they return to the office. Specifically, we will create a Web Dynpro application that accesses a

Web page to download an interactive PDF form from a back-end system to the local hard drive on a remote user's laptop. The user then records and stores travel expenses in the PDF form, and upon returning to the office, uploads the PDF form to the SAP R/3 system through that same Web Dynpro application. The application can then validate and automatically transfer the data to the corresponding Travel Expense Manager process in the SAP R/3 system via a standard SAP-delivered BAPI. The system automatically creates the corresponding trip in the system and passes the trip number back to the user through a Web page.

Building and deploying the sample Web Dynpro travel expense form application involves four technical components:

- **A custom Web Dynpro for Java application**, which will retrieve data from a form being uploaded (using the integrated Adobe document services) and transfer the data to the SAP R/3 back-end system. The Web Dynpro application will also provide the Web pages used to download and upload the PDF form.
- **A custom, standalone, interactive PDF form**, which we will build using the Adobe LiveCycle Designer forms layout tool. The form will be integrated into the Web Dynpro for Java application.
- **A standard BAPI function module on the back-end SAP R/3 system**, which will receive and post the data sent by the Web Dynpro application.
- **A System Landscape Directory (SLD)**, configured on SAP Web AS, which will provide the connection information needed for the custom Web Dynpro application to connect to the back-end SAP R/3 system.[1]

[1] The SLD, which is included with SAP Web AS Java 6.40 and higher, is a central data repository that is used by various SAP applications, such as SAP Solution Manager and SAP Exchange Infrastructure, to access information on systems and solutions deployed in the system landscape. To find out more about how to set up and use the SLD, go to http://service.sap.com/sld, or read the corresponding SAP NetWeaver documentation at http://help.sap.com (navigate to the SAP Library via Documentation → SAP NetWeaver → SAP NetWeaver 2004 → English; then, in the SAP Library, navigate to SAP NetWeaver → Application Platform → Java Technology in SAP Web AS → Administration Manual → Server Administration → SAP System Landscape Directory).

Note!

Ideally, you already have general background knowledge about Java development within an SAP context.[2] While the example application is built using Web Dynpro for Java, the underlying principles also govern the development process for Web Dynpro for ABAP, which is shipped with SAP NetWeaver 2004s and also integrates Interactive Forms. Therefore, all references in this article to the term "Web Dynpro application" technically refer to a Web Dynpro for Java application, but also applies to Web Dynpro for ABAP applications.

Note!

SAP NetWeaver Developer Studio — the Java development environment — and the Web Dynpro framework execute many of the more tedious programming tasks in the background. So, even if you are not too familiar with the development tool, you should be able to design the most important individual parts, such as the views and the context, of the example application without much trouble.[3]

While the example is a travel expense form application, remember that the processes and techniques described can apply to any number of scenarios (e.g., supplier feedback surveys that a company uses to gather input for its supplier rating system, an order form that is included in a sales executive's customer visit report, etc.) and

[2] For information on Java development within an SAP context, go to the SAP Developer Network at http://sdn.sap.com.

[3] For more information on SAP NetWeaver Developer Studio, read the article "Get Started Developing, Debugging, and Deploying Custom J2EE Applications Quickly and Easily with SAP NetWeaver Developer Studio" (*SAP Professional Journal*, May/June 2004).

The SAP Interactive Forms architecture

The Interactive Forms solution consists of technology provided by both SAP and Adobe.

SAP contributes:

- **SAP Web Application Server:** SAP Web AS includes the development environments for Java (SAP NetWeaver Developer Studio) and ABAP (ABAP Workbench), as well as the necessary runtime environments.
- **Active Components Framework:** To enable communication between the integration application and the PDF form displayed in the Web browser, SAP provides the Active Components Framework plug-in, which needs to be installed on every front-end system running a Web Dynpro application that integrates an interactive form.

Adobe contributes:

- **Adobe LiveCycle Designer:** Adobe LiveCycle Designer is Adobe's design-time forms layout tool. It is integrated into SAP NetWeaver Developer Studio and the ABAP Workbench, and it enables the creation of forms that combine high-fidelity presentation with XML data handling. The easy-to-use graphical interface of Adobe LiveCycle Designer enables developers to quickly design forms, maintain form templates, define a form's business logic, make changes, and preview forms before they are deployed as PDF files.
- **Adobe document services (ADS):** The ADS are runtime services that are deployed on the SAP J2EE Engine (the Java stack of SAP Web AS). These services provide a range of form and document creation and manipulation functions used by applications in business operations.

  These services fulfill two basic tasks:

  - Merge XML form templates (created using Adobe LiveCycle Designer) with current SAP application data in XML format, and convert the result to PDF (and to various print formats in the case of back-end printing).
  - Extract data entered into an interactive PDF form by a user, and transfer that data back into SAP applications using XML.
- **Adobe Reader:** Adobe Reader is the free viewer that users need to display and edit interactive PDF forms. While Adobe LiveCycle Designer and the ADS are shipped as an integral part of SAP NetWeaver, Adobe Reader is available for download from www.adobe.com.*

The ADS act as a server-side process for generating PDF-based interactive forms as well as for extracting data from an interactive form during runtime. The ADS were developed using the Enterprise JavaBeans (EJB) technology** and run in the SAP J2EE Engine. To make communication with the ADS easy for both Java and ABAP applications, the services are further exposed as a Web service, an open standard

* The full version of Adobe Acrobat can also be used, but it is not necessary because the Interactive Forms technology enables the necessary editing capabilities in Adobe Reader.

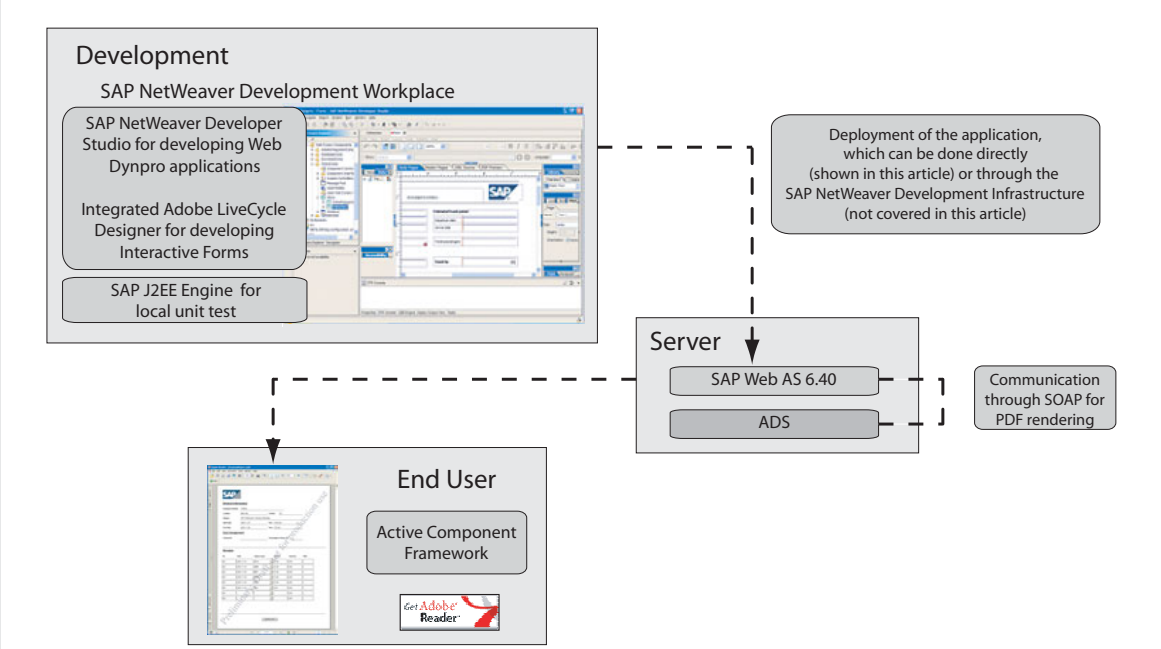
** For more on EJB, see the *SAP Professional Journal* articles "Persist Data for Your J2EE Applications with Less Effort Using Entity Beans with Container-Managed Persistence (EJB CMP)" (July/August 2004) and "The New EJB 3.0 Specification — Why It's Time to Reevaluate Enterprise JavaBeans" (November/December 2005).

communication protocol that enables applications built from both Java and ABAP to talk to the ADS easily using standard SOAP messages over HTTP.***

To further ease the lives of SAP application developers, SAP has encapsulated all the features and functions that the ADS provide in an SAP development object called the PDF Object. The PDF Object is available for both Java and ABAP. It is essentially a wrapper class that allows access to the ADS capabilities from both the Java and ABAP development environments.

In general, the Web Dynpro framework takes care of the communication with the ADS via the PDF Object. This allows for data transfer from the back end into the form and vice versa. Nevertheless, a developer can also call the PDF Object directly from the code to execute certain PDF-related functions on the ADS.

The illustration below shows how these different tools and components work together during design time and runtime. On the development side, you have the SAP NetWeaver Development Workplace, which includes SAP NetWeaver Developer Studio**** and a local SAP J2EE Engine. You can then start the Web Dynpro and Interactive Forms development using SAP NetWeaver Developer Studio integrated with Adobe LiveCycle Designer (as you will learn in this article). On the server side, using the SAP J2EE Engine Visual Administrator tool, you can configure your SAP J2EE Engine to use the ADS running inside an SAP Web AS for generating interactive forms and extracting data. Finally, on the user side, the user downloads the form from the application, views and completes the form, and then uploads the form so the data can be stored in the system.



*** See the article "Extend the Internal and External Reach of Your Applications with ABAP-Based Web Services" (*SAP Professional Journal*, July/August 2005).

**** See the article "Get Started Developing, Debugging, and Deploying Custom J2EE Applications Quickly and Easily with SAP NetWeaver Developer Studio" (*SAP Professional Journal*, May/June 2004).

Prerequisites

To build the sample Web Dynpro for Java application based on SAP NetWeaver ’04, you’ll need specific components on the server and the front-end system.

On the server:

- SAP Web Application Server Java 6.40
- Adobe document services (deployed and configured on the SAP J2EE Engine)
- System Landscape Directory (the central registry for all your systems)
- SAP R/3 Enterprise (SAP R/3 4.7) (with the Travel Expense Manager modules included)

On the front-end system:

- SAP NetWeaver Developer Studio (from SAP NetWeaver ’04, with at least Support Package Stack 10)
- Adobe LiveCycle Designer 7.0 (included with SAP NetWeaver Developer Studio)
- Adobe Reader 7.0.3 or higher (available for free from www.adobe.com)
- Active Components Framework (from SAP NetWeaver ’04, required for displaying an interactive PDF form as part of a Web Dynpro application in Microsoft Internet Explorer)

will leave you well-equipped to explore and take advantage of the additional capabilities offered by Interactive Forms that are not covered here.

Let’s get started!

Creating a forms-based Web Dynpro application

The sample application we want to build has a simple interface as an entry point (refer back to Figure 1). The Download Travel Expense Form button takes users to a view that allows them to download an interactive PDF form for recording expenses; the Upload Travel Expense Form button takes users to a view that allows them to upload the completed PDF form to the back-end SAP R/3 system when they return to the office.

Note!

The Help button on the form does not include any functionality at this point. You can customize the information that will be most useful for your end users when you create the button yourself.

Developing the Web Dynpro application involves the following steps, which we’ll walk through one at a time:

1. Create a Web Dynpro project.
2. Create the required Web Dynpro views.
3. Integrate a UI element into a view.
4. Create a form template in Adobe LiveCycle Designer.
5. Insert form objects into the form.

Web Dynpro basics

- Using Web Dynpro technology enables a clear separation of processing logic and presentation logic for enhanced maintainability and stability. A Web Dynpro application runs on the front-end system and has local or remote access to the back-end system using a service. This means that the display logic is contained in the Web Dynpro application, while the business logic and the persistence of the business objects run in the back-end system.
- Every Web Dynpro application is structured according to the MVC programming model:*
  - The Web Dynpro **model** forms the interface to the back-end system and thus enables the Web Dynpro application to access the back-end data.
  - The **view** is the central logical element of the Web Dynpro application for the layout and processing of the presentation logic.
  - The **controller** is responsible for the data and control flow in the Web Dynpro application. There are different controller types, such as component controllers, view controllers, etc. Since the controller contexts have the same structure, you can process every controller type in the same way.

* For more on the MVC approach, see the *SAP Professional Journal* articles "Build More Powerful Web Applications in Less Time with BSP Extensions and the MVC Model" (March/April 2003) and "Develop More Extensible and Maintainable Web Applications with the Model-View-Controller (MVC) Design Pattern" (January/February 2004).

Note!

In the example, the downloaded PDF form is a pre-generated static PDF, which means that the Web Dynpro application provides an empty form in which the user fills in all the mandatory fields. The Interactive Forms solution provides all the capabilities needed to pre-populate certain fields on the form — with user information, for example.

6. Code the Web Dynpro application to exchange data with the form.
7. Complete the Web Dynpro application and deploy it.

For those of you who are not familiar with Web Dynpro (or for those who want a quick refresher on the fundamental concepts), see the sidebar above.

The first step in the development of our Web Dynpro application is to create a Web Dynpro project using SAP NetWeaver Developer Studio.

Step 1: Create a Web Dynpro project

Figure 2 (which appears on the next page) provides an overview of the complete structure of the Web Dynpro project we want to create.

The **project** is the container for all your

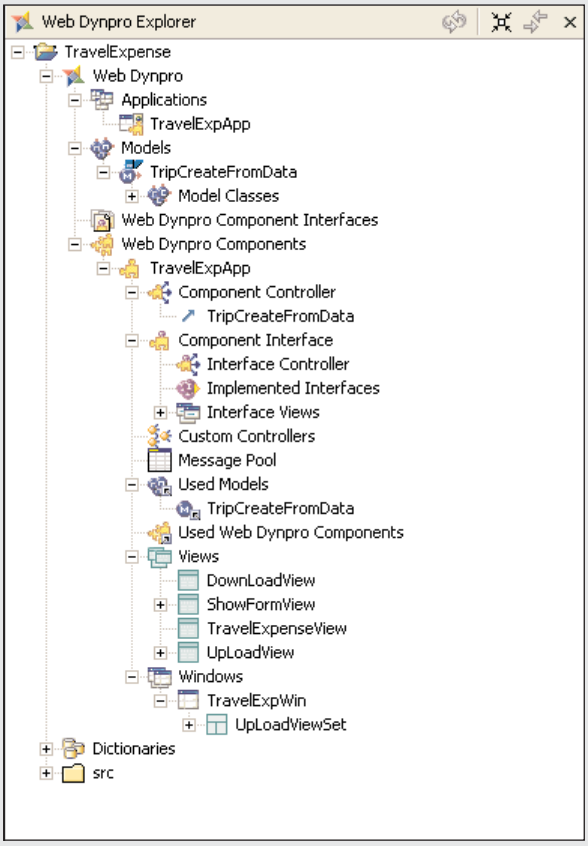


Figure 2 Web Dynpro project to be created

> ***Note!***
>
> As shown in Figure 2, the Web Dynpro Explorer provides an overview of the example Web Dynpro application. For any Web Dynpro application that you create, you will see in the Web Dynpro Explorer all application entities, including Applications, Models, Web Dynpro Component Interfaces, and Web Dynpro Components. The Explorer also includes a Dictionaries folder, which contains the data types and structures defined for the application, and an src (source) folder, which houses the Java packages used in the Web tier of the application, both of which are beyond the scope of this article. Also, as you build the application, the Web Dynpro framework automatically generates the required project files, references files, descriptor files, etc., the discussion of which is also beyond the scope of this article.

Web Dynpro-related development. To create a Web Dynpro project in SAP NetWeaver Developer Studio, follow the menu path File → New → Project and select Web Dynpro Project, which launches the New Dynpro Project Wizard. Fill in the required information — such as the name of the project (TravelExpense in the example), the directory for the project (the default, which we keep for the example, is the SAP NetWeaver Developer Studio workspace directory), and the display language — and then click on Finish. The Web Dynpro framework generates the required file structure for storing the project metadata.

First, we need a **component**, which is the part of the application that contains the actual user interface functionality — in the example, various views with buttons and links, and one view containing an interactive form. If you expand the newly created TravelExpense project node in the Web Dynpro Explorer, and then expand the Web Dynpro node, you'll see a number of generated subnodes (Applications, Models, Web Dynpro Component Interfaces, Web Dynpro Components). Right-click on the Web Dynpro Components node and select Create Web Dynpro Component from the context menu, which launches the New Web Dynpro Component wizard. Enter the required information, such as the name of the component (TravelExpApp in the example) and a component package to contain the generated XML files, which describe the model you create for the application using the visual tools in SAP NetWeaver Developer Studio. The wizard then automatically prompts you to create an embedded **view**, which is the interface the user will interact with when working with the application. You have the option of creating at least one view at this time. You can (and will in this article) create other embedded views. For now, make sure that the Embed new View option is selected (it should be by default). Either accept the suggested view name and view package

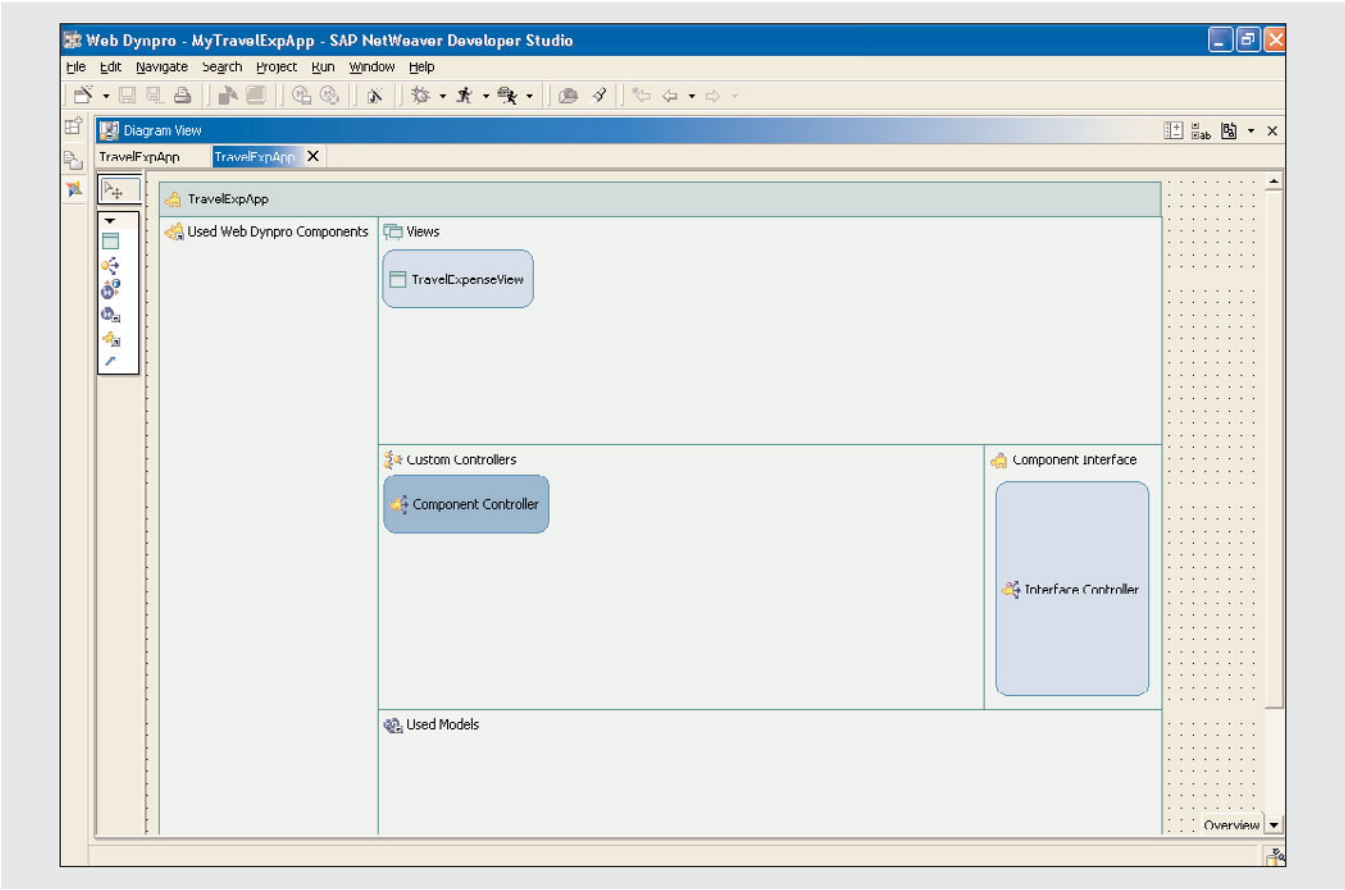


Figure 3 Diagram View of the Web Dynpro component

name (TravelExpenseView and com.adobe.travelexp.add, respectively, in the example) or enter new ones, and then click on Finish. The framework now generates the required subnodes underneath the Web Dynpro Components node, as shown in Figure 2.

Next, we need to define the **model** described in the sidebar on page 33. In the model, you define where your business process (i.e., function module) resides. As you may be aware, defining models is usually the very first step in a model-driven application in the MVC paradigm.[4]

To define the model, we first need to identify an existing BAPI in the back-end SAP R/3 system to serve as the model for the Web Dynpro application. Note that we are not going to build the BAPI here in Web Dynpro, but rather specify an already existing BAPI running in the back-end SAP R/3 system as the endpoint for the business process. Then we can specify the input and output parameters of the BAPI as the input and output of the Web Dynpro application. We do the latter in the model context, which is automatically generated when you create the model.

To create a model, double-click on the TravelExpApp node under the Web Dynpro Components node to display the Diagram View (see **Figure 3**). Note that the TravelExpenseView is automatically created. This is the default entry point into the application.

Right-click in the Used Models area and select

[4] See the *SAP Professional Journal* articles "Build More Powerful Web Applications in Less Time with BSP Extensions and the MVC Model" (March/April 2003) and "Develop More Extensible and Maintainable Web Applications with the Model-View-Controller (MVC) Design Pattern" (January/February 2004).

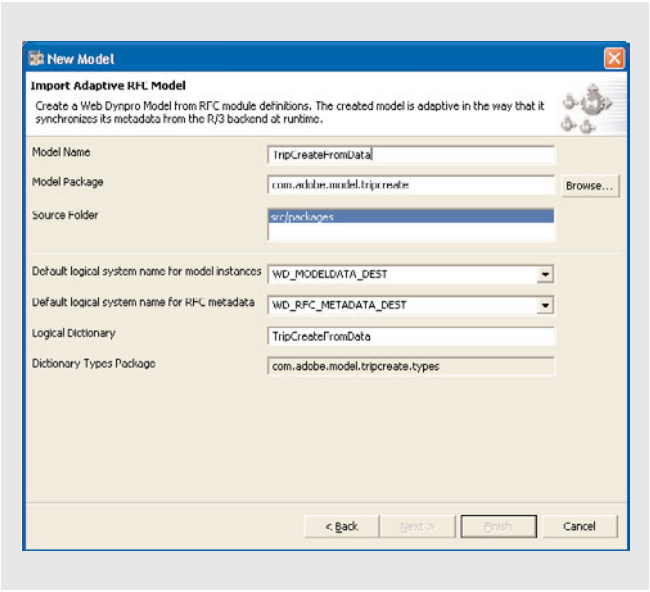


Figure 4 Defining a new model

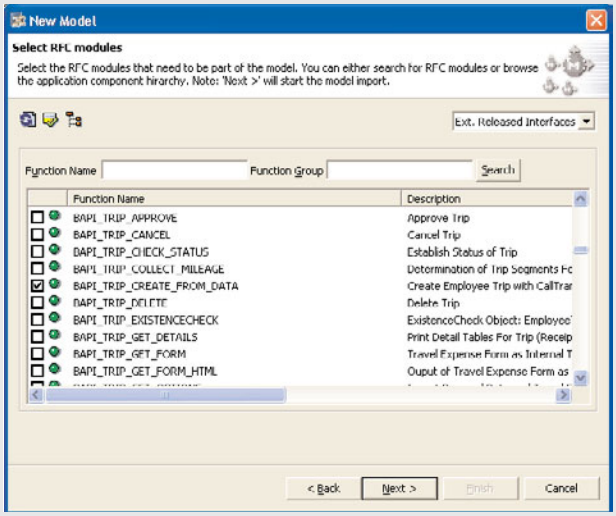


Figure 5 Selecting a BAPI from the back end

Create Model from the context menu, which launches the New Model wizard. We want to use a BAPI from the back-end SAP R/3 system, so select Import Adaptive RFC Model and click on Next. As shown in **Figure 4**, fill in the fields for the model name (TripCreateFromData in the example) and the model package (com.adobe.model.tripcreate in the example). Select a default logical system name for the model instance and for the RFC metadata — for simplicity we use standard SAP options for these names in the example, but it is generally recommended you create your own. Leave the default value for the logical directory, which is derived from the model name and can be changed if you want, as well as for the source folder and the dictionary types package, which cannot be changed, and then click on Next.

On the next screen, enter the logon data for your back-end SAP R/3 system. Then you can search for the BAPI you want to use. For the example, we select BAPI_TRIP_CREATE_FROM_DATA, which is for creating a travel expense report (see **Figure 5**). Clicking on Next starts the import of the model and generation of the associated Java adapter classes, which concludes with a detailed import log message. The Web Dynpro Explorer now shows the generated model context in which the input node corresponds to the input parameters of the BAPI (e.g., the data in the form that is passed to the back end) and the output node corresponds to the return parameters of the BAPI (e.g., the trip number that is passed to the user after the trip is created in the back end).

In the next step, we need to create a link between the TripCreateFromData model and the Component Controller, which is the main controller for the entire application. It contains the component's processing logic and acts as the internal interface between the data obtained from the model and the view (or views). The controller is needed for executing the model. (In a complex application, it makes sense to create additional custom controllers for managing different aspects of your application, but for our example, we'll use only the Component Controller.)

To create the link, click on the small blue arrow () on the Diagram View toolbar. Then drag the Component Controller object down to the TripCreateFromData model object. When you release your mouse button, a line from the Component Controller object to the model object is automatically drawn to indicate the relationship between them and the Edit Model Binding dialog opens. Here you define the relationship and mapping of the Web Dynpro con-

text between the controller and the model. You also specify which input and output elements of the model you need in your application. In our case, drag and drop the nodes from the BAPI (e.g., the top-level nodes Bapi_Trip_Create_From_Trip_Input and Bapi_Trip_Create_From_Trip_Output) from the right column (model) onto the Context root node on the left (controller), and then select the sub-nodes and attributes you want. Once these are defined, there will be a mapping relationship between the controller's attributes and the model's attributes, and the input and output data will flow via these links between the controller and the model. Click on Finish when you have completed your selections.

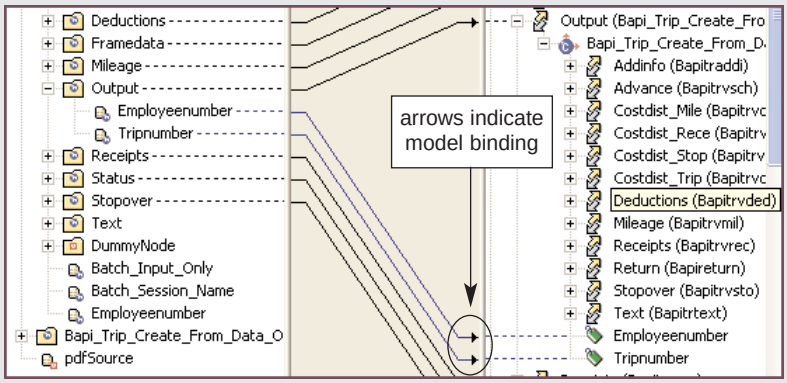
In addition to creating a link between the controller and the back-end data, we also need to manually create one binary-type value attribute in the controller's context that will help pass the PDF form between different views inside the application. This value attribute acts as a placeholder for the uploaded PDF (i.e., the filled and uploaded travel expense form) and gives the different views (and any controller) access to the PDF. To create the required value attribute, double-click on the Component Controller node in the Web Dynpro Explorer, and select the Context tab in the right frame. Right-click on Context, select New from the context menu, and then select Value Attribute. Enter the name you want (pdfSource in

Note!

The context for a controller entity is created by the Web Dynpro Generator by default. This applies to all controller types of a Web Dynpro application (i.e., component controllers, view controllers, and custom controllers). The Web Dynpro tool Controller/ Context Editor provides support for the creation of a context structure through the definition of nodes and attributes. You carry out the structure definition separately for every controller; the procedure for creating the structure tree is the same for all controller types.

Tip!

For a leaner application, try to be more specific in your data retrieval than simply using the top-level nodes. As shown in the screenshot below, you can see that from a large number of nodes under Output on the right side, we selected only Employeenumber and Tripnumber. Blue arrows that lead from the selections on the left to those on the right indicate the model binding.



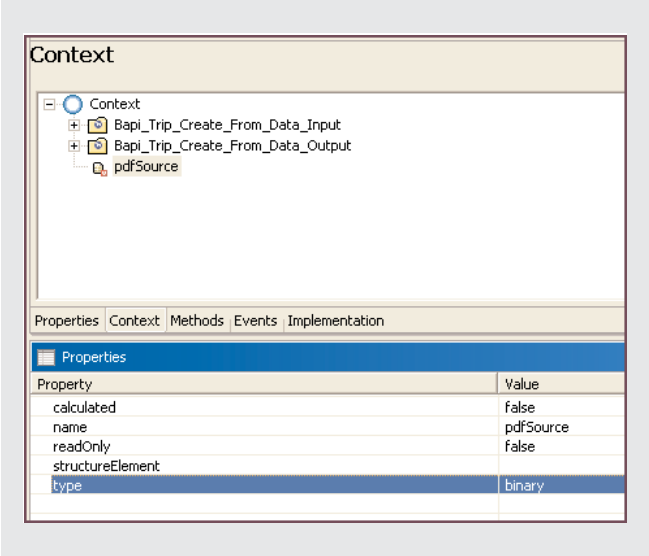


Figure 6 Creating a value attribute in the controller's context

the example). After clicking on Finish, go to the Properties tab and set the type to binary. See **Figure 6** for the correct settings.

With the context defined, we can now go ahead and create additional views for the various pages in the Web Dynpro application.

Step 2: Create the required Web Dynpro views

In addition to the default TravelExpenseView generated at the beginning (refer back to Figure 3), which provides the entry point to the application, we need to create three more views:

- **DownLoadView:** The page that will contain the link for downloading the travel expense form from the back-end system. This will be a static page that only contains a URL link pointing to the file location of the PDF form to be downloaded.
- **UpLoadView:** The page the user will access for uploading the completed travel expense form to the application. This page will contain a simple UI element for uploading the PDF file, and another for using the ShowFormView to display the PDF file and verify the entered data.
- **ShowFormView:** The page that the user will access to display the uploaded travel expense form. Only this page will actually render and display the PDF form.

Because we want to focus on the Interactive Forms integration, we will only briefly touch on creating the views that do not directly integrate the form (i.e., most of our focus will be on the ShowFormView).

To create a view, return to the Diagram View of the project (Figure 3). Right-click in the Views area, select Create View from the context menu, and enter a name for the view (e.g., DownLoadView). You can accept the default value for the package or enter a new name, but you cannot change the name of the source folder. Click on Finish when you are done. Proceed likewise with the other views.

As part of the process of creating a view, you also need to create data navigation links, in this case, between the ShowFormView and UpLoadView, and the Component Controller. The data navigation links enable these two views to access the completed travel expense form (i.e., the uploaded PDF document) in order to display and upload it, respectively. The DownLoadView doesn't need a data link here because it only contains a static URL link for downloading a blank PDF form. To establish the links, proceed as we described in Step 1 (on page 35) when we created the data link between the controller and the model. Do *not* forget to map the pdfSource node from the controller to an identical node in the view context. This will ensure that, when we run the application, what is displayed in the PDF inside the application is really what was entered in the form.

Figure 7 shows the application setup, including the data navigation links between the different parts of the project — between the UpLoadView and ShowFormView, and the Component Controller.

Figure 8 and **Figure 9** show the completed download and upload views. As you can see, Figure 8 contains a link to the travel expense form, and Figure 9 includes the upload and display functions.[5]

[5] As mentioned previously, we will only briefly touch upon the creation of the DownLoadView and UpLoadView because these views have nothing to do with the Interactive Forms capability itself.

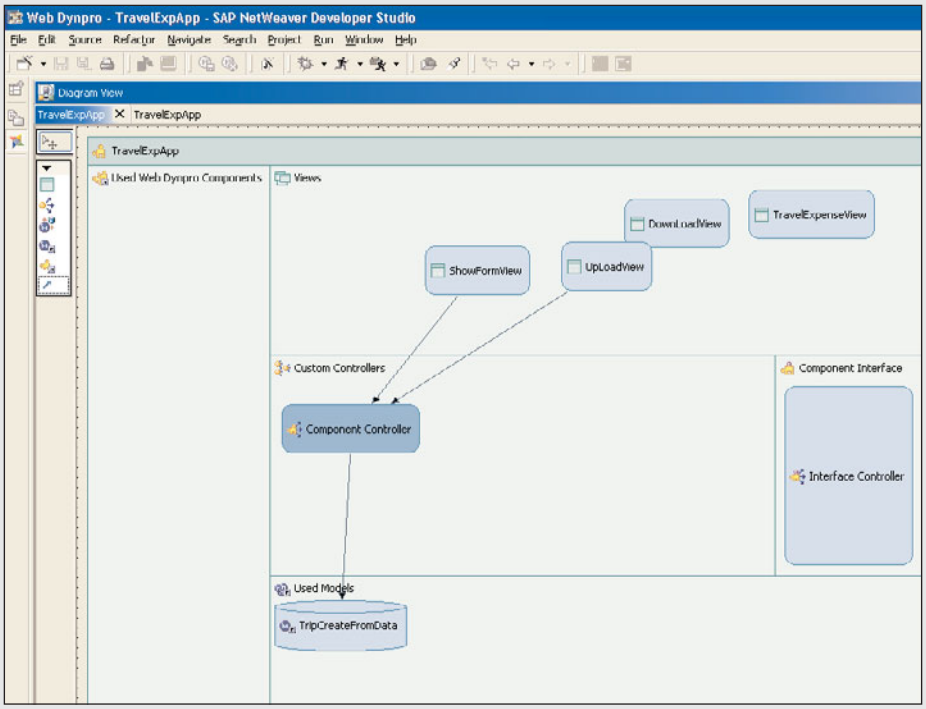


Figure 7 The example application with the new views added and relationships established

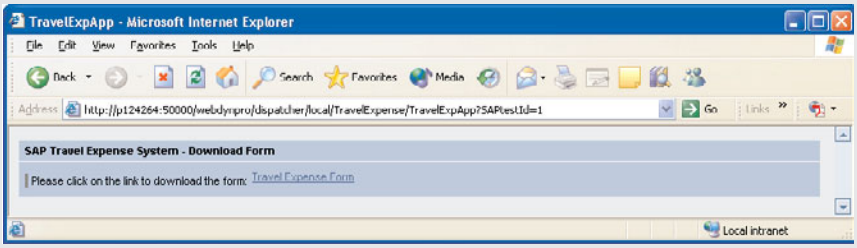


Figure 8 View of the Web page for downloading the travel expense form

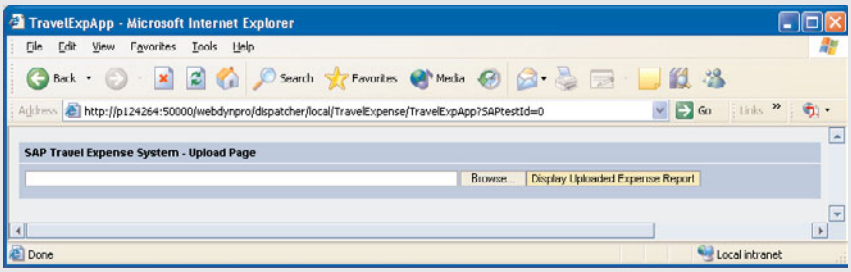


Figure 9 View of the Web page for uploading the travel expense form

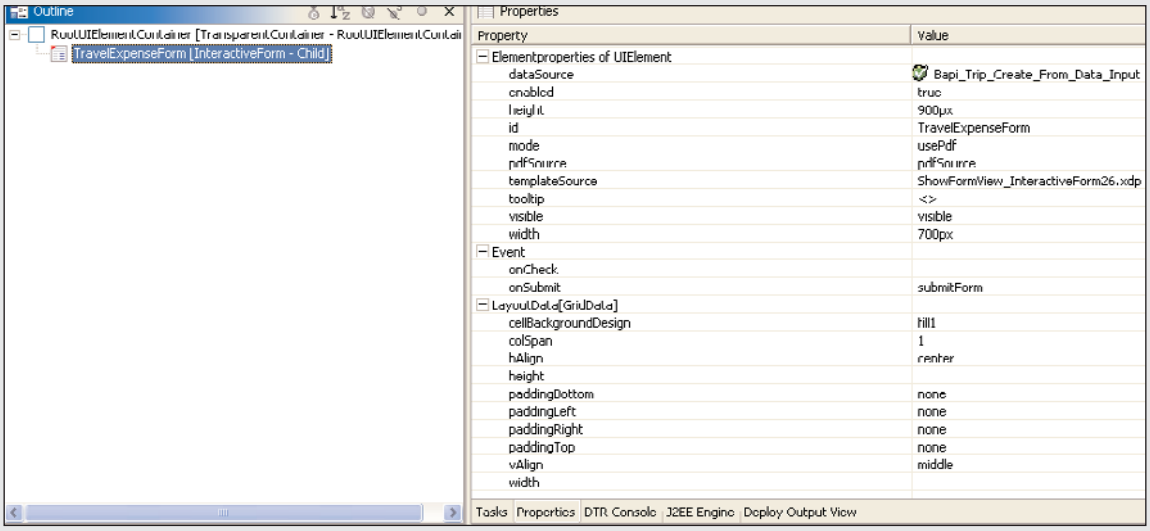


Figure 10 InteractiveForm UI element properties

Now that we have created the views and defined their navigation relationships, we can work on showcasing the Interactive Forms integration.

> ***Tip!***
>
> The navigation flow from one view to another can be visually defined in the Web Dynpro window through the use of inbound and outbound plugs. The details on how to define the navigation between views can be found in the SAP NetWeaver documentation at http://help.sap.com (see footnote 1 on page 29 for the navigation path to the SAP NetWeaver documentation).

Step 3: Integrate a UI element into a view

Each view will have its own UI elements: The DownLoadView will have the FileDownload UI element for downloading the PDF form from the back end; the UpLoadView will have the FileUpload UI element for uploading the PDF form and a Button UI element for displaying the uploaded form; and the ShowFormView will have the InteractiveForm UI element for displaying the PDF form. We will add the UI element to the ShowFormView to illustrate how easy this step is.

In the Web Dynpro Explorer, double-click on the ShowFormView of the TravelExpense project. In the Web Dynpro UI Element Library, click on the Adobe Library to display the Adobe-related elements, and then drag and drop the InteractiveForm UI element onto the form. This InteractiveForm UI element is the container that will hold the objects that will constitute the interactive data-entry form.

Let's take a look at the properties of this UI element using the Properties tab in the lower-right frame of SAP NetWeaver Developer Studio. **Figure 10** shows the properties of the InteractiveForm UI element.

Many of the values are set by default, but you can change them if necessary (e.g., you can change the height and width of your InteractiveForm UI element). The following are key properties for which you need to provide (or verify) values:

- **dataSource:** The data source encapsulates the data you can manipulate in the form at runtime. For the dataSource property, you need to specify a context node providing the input or output data you want. In the example, we reference the corresponding context node we defined in the context structure of the model, which is based on the BAPI we selected. The structure and attributes of the context node, which are referenced by the dataSource property, will later be displayed in the Data View tab of the Adobe LiveCycle Designer when we edit the PDF form. We will later define the UI elements on the PDF that can be bound to the attributes in the context node corresponding to the BAPI's input and output, as defined during the model generation.
- **mode:** At design time, you specify how the PDF form is created. In the example, we select the usePdf mode,[6] because we are in a process that uses a pre-generated PDF form without prefilled values (i.e., we do not generate the form on-the-fly based on a template and current system data).
- **pdfSource:** This property specifies the context attribute that stores the PDF form. You need to bind this property to a context attribute of the binary type. This is the pdfSource value attribute we created in Step 1 (on page 37). In the example, the interactive form will be stored in this attribute during runtime. This property allows the application to access the form and transfers the data to and from the form through the Web Dynpro context.
- **templateSource:** This property specifies the name of the form template stored in the back-end system. In Web Dynpro, the name of the template is automatically generated when you insert the InteractiveForm UI element into the view.

Note!

The default value of the enabled property is true, which allows for the interactive functionality of the form, and the default value for the visible property is visible, which allows the form to be displayed. The id property takes its value from the name of the form. The tooltip property, which functions like any screentip or tooltip, is optional.

[6] See the "Tips and Tricks" section at the end of this article for information on the generatePdf mode, which is used for online scenarios vs. the offline scenario developed in this article.

Step 4: Create a form template in Adobe LiveCycle Designer

Next, we create the actual form template. Right-click on the TravelExpenseForm [InteractiveForm - Child] item in the Outline frame (see Figure 10) and select Edit from the context menu. This launches the Adobe LiveCycle Designer inside SAP NetWeaver Developer Studio. When the application starts, the Adobe LiveCycle Designer workspace provides the tools you need to design the form you want.

Using **Figure 11** on the next page, which gives you a preview of the layout of the example travel expense form, let's first have a look at the Adobe LiveCycle Designer workspace. The Layout Editor in the center is the main area where you create and maintain the form layout. (The Layout Editor is empty when you launch Adobe LiveCycle Designer inside SAP NetWeaver Developer Studio.) Tabbed palettes around the Layout Editor provide easy access to the respective tools.

Important!

Using the features of Adobe LiveCycle Designer goes beyond the scope of this article. We introduce you to Adobe LiveCycle Designer here in Step 4. In Step 5, we show you how to insert form objects, but we do not show you how to insert *all* of the form objects shown in Figure 11. Once you understand how to insert some of the form objects, however, it should be relatively easy to figure out the rest.

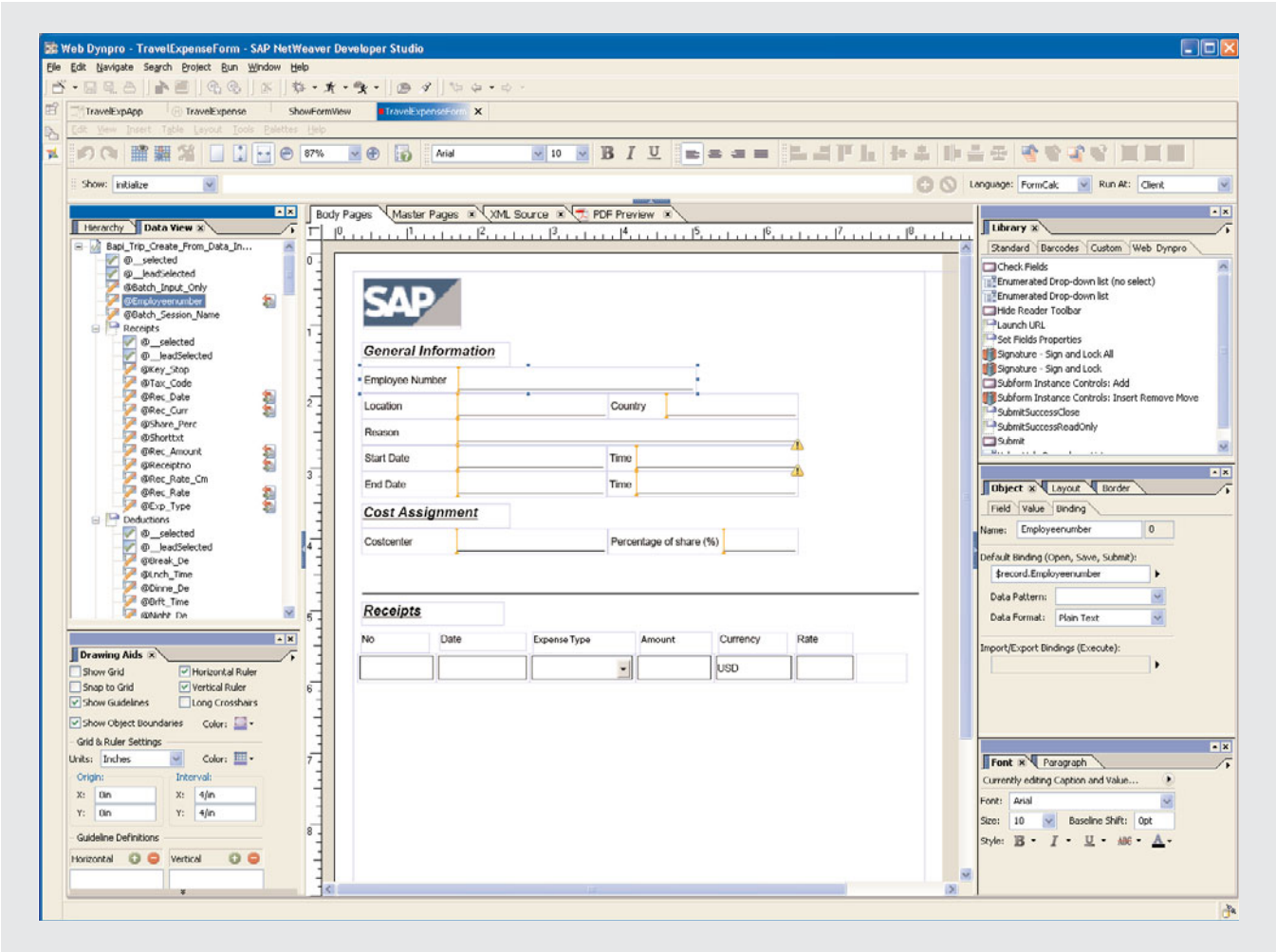


Figure 11 Adobe LiveCycle Designer workspace integrated into SAP NetWeaver Developer Studio

The Layout Editor contains four tabs:

- **Body Pages:** Body pages contain a form's content (e.g., a table with data from the back-end system).
- **Master Pages:** Master pages specify the layout and background for the form. Here you add objects that will occur in the same position throughout the form, such as a logo.
- **XML Source:** This tab contains the XML code that defines all aspects of the form layout. As you build a form template, Adobe LiveCycle Designer writes the corresponding XML code in the XML Source tab. There is nothing that needs to change on this tab. It just displays the coding that has been done for you.
- **PDF Preview:** Use the PDF Preview tab to view and test the operation of a form or template as if it were a PDF file.

The palettes, each of which may contain several tabs, logically group features and functions you need when you design the form template. Here's a list of the most important palettes and their special focus:

- The **Hierarchy** palette at the upper left is a graphical representation of the contents in the Body Pages and Master Pages tabs — i.e., the form objects from the Library palette.

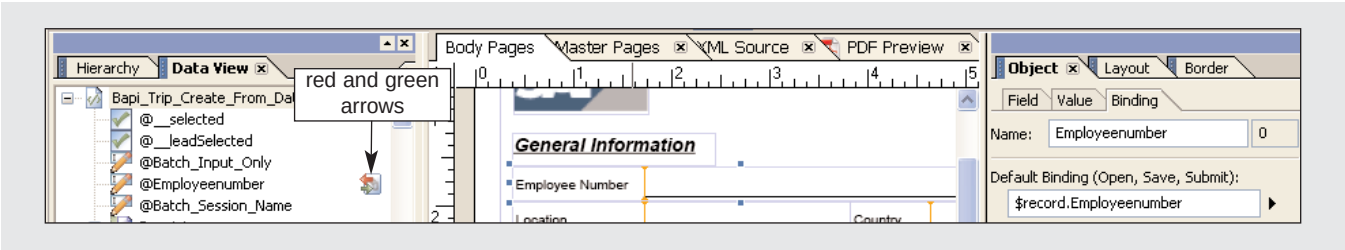


Figure 12 Binding a form object to a data source attribute

- The **Data View** palette, also at the upper left, displays the hierarchy derived from your data source. As you learned in Step 3, you define your data source in the Web Dynpro context. When you launch Adobe LiveCycle Designer, the Web Dynpro framework automatically generates an XML schema, which is the basis for what is displayed in this palette. In other words, the nodes and attributes you defined in the Web Dynpro context and specified in the InteractiveForm UI element's dataSource property appear here. You will see the same tree structure that you defined in the InteractiveForm UI element's context.
- The **Library** palette at the upper right contains all the objects (layout elements) that you can add to a form layout, such as text fields, images, etc. Adobe LiveCycle Designer also offers a number of Web Dynpro-specific objects (e.g., buttons, drop-down lists, etc.) specially developed for SAP, that can interact directly with the Web Dynpro framework.
- The **Object** palette at the middle right is used to modify properties that are specific to the selected form object.

Adobe LiveCycle Designer allows drag-and-drop operations for many of the items contained in the palettes. Usually, these operations are used for placing form objects on the form (from the Library palette) and binding them to fields in the back-end system (from the Data View palette). We want a number of different form objects on the example form:

- An image object for the display of the SAP logo
- Text objects for static, read-only texts
- Date/time fields
- Text fields, which are placeholders for text that can change every time the form is rendered or used
- A drop-down list for predetermined selection values
- A button for triggering a specific action

Step 5: Insert form objects into the form

You can place different elements on a form by adding them to the form template. As shown in Figure 11, there is an image (the SAP logo), three headings, ten text field objects (eight under the General Information heading, two under Cost Assignment), and a table object under the Receipts heading.

For example, to create a text field, such as the Employee Number field, go to the Data View tab, select the @Employeenumber node, which represents the corresponding field in your back-end system, and drag it onto the form. You will notice that a small icon with a red arrow and a green arrow pointing in opposite directions appears in the Data View (see **Figure 12**). This is indicates that the form object for the field is now bound to a data source attribute in the Web Dynpro context — in this case, the field containing the employee number. (There are several text fields in the example, including Location, Country, and Reason, all of which are created the same way.)

You can always check (or change) the binding of a form object by selecting the Binding tab on the Object palette. You can also use the Object palette to change the caption of an object, for example, by selecting the

Field tab. By default, a form object's caption is derived from the name of the node shown on the Data View palette. On the Value tab of the Object palette, you can specify, for example, whether a field is used for calculating and displaying values or intended for mandatory or optional user input. The Layout and Border palettes provide additional options for positioning and formatting the form object.

To add an image object (like the SAP logo), you would use the Library palette to drag an image object from the Standard tab, and drop it onto the empty form in the Layout Editor. Then you would use the Object palette for the image object to specify the URL of the location of the graphic. To guarantee that the image is stored inside your form template, so you don't need to worry about access rights to the original location of the graphic, you need to make sure that the Embed Image Data checkbox is selected.

There are other objects in the example form; however, as noted earlier, using all the features of Adobe LiveCycle Designer goes beyond the scope of this article. Therefore we do not provide the steps for creating a table in the form, adding the drop-down list feature or Submit to SAP button to the form, the Help button to the initial view, the URL link to the download view, the Browse and Display Uploaded Expense Form buttons to the upload view, and so on. But, as you can tell, working in Adobe LiveCycle Designer is quite easy and intuitive, and adding these and other features to the views should not be difficult.

Step 6: Code the Web Dynpro application to exchange data with the form

The Web Dynpro framework interacts with the integrated PDF form in two ways. First, as we've seen through the example, the Web Dynpro context is used to define the data retrieval. The Web Dynpro context constitutes the basis for the XML schema that the framework generates. In Adobe LiveCycle Designer, the framework offers all the attributes (e.g., employee number, receipts, etc.) defined in the context through the Data View palette, so they can be used on the form. Second, buttons (created using the Web Dynpro Library attributes) on the form can trigger certain actions. You can use buttons for a wide variety of actions, but to facilitate development, the Web Dynpro framework provides two SAP-specific buttons: Submit to SAP and Check Fields, which correspond to the onSubmit and onCheck actions.

In the example, the Submit to SAP button appears at the bottom of the form. When the user clicks on it after uploading the PDF form to the Web Dynpro application, the onSubmit event establishes the connection to the ABAP back end and submits the data entered in the form to the corresponding fields in the back-end system.

How do we know which action is triggered? In the properties of the interactive form (see the Event section in Figure 10), you can define an action for the onSubmit event, which always corresponds to the Submit to SAP button on the form. An action in Web Dynpro is basically a Java method being generated for you on the Implementation tab, based on the action that you assign to the onSubmit event. You add your coding to the generated method, but this goes beyond the scope of the article. In the example, we execute the model we generated earlier that corresponds to the BAPI we want to call.

Step 7: Complete the Web Dynpro application and deploy it

Once all of the views are created, and the corresponding UI elements added, we can create the Web Dynpro application, which means to encapsulate the Web Dynpro project into a deployable and executable unit.

In the Web Dynpro Explorer, right-click on the Applications node, and then select Create Application from the context menu, which opens the corresponding wizard. Enter a name for the application (e.g., TravelExpApp) and a package name for the XML files to be generated. On the next screen, select the corresponding component (e.g., TravelExpApp), the interface view (e.g., TravelExpWinInterfaceView), and the startup plug (e.g., Default) from the respective drop-down menus.

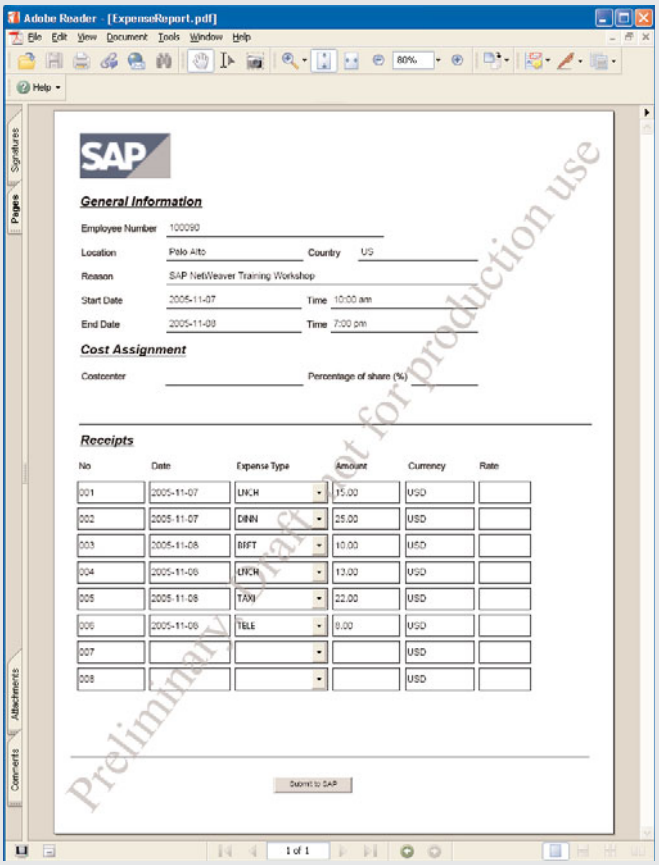


Figure 13 Completed PDF form with entered travel expense data

Save all the changes. During the save process, SAP NetWeaver Developer Studio will compile all of the code for the application and return any errors (e.g., type mismatches, undeclared classes, etc.) in the task pane. Once the application is successfully compiled, we can deploy and test the example:

1. **Launch the Web Dynpro application.** To launch the Web Dynpro application, right-click on the TravelExpApp application in the Web Dynpro Explorer (TravelExpense → Web Dynpro → Applications → TravelExpApp), and select Deploy New Archive and Run. You now see the simple initial view of the application (the SAP Travel Expense System Web page as shown in Figure 1), which is the entry point from which users can access the download or upload view.

2. **Download the PDF form.** Click on the Download Travel Expense Form button, which takes you to the download view containing the URL link to the PDF form. Click on the Travel Expense Form hyperlink, which prompts you to save the PDF form to your hard drive. Save the PDF form, go to its location on your hard drive, and open it in Adobe Reader.[7] Now you can fill in the form even though you are offline. Once you have completed the form, save it. **Figure 13** shows an example of a completed form.

3. **Upload the completed form.** Return to the application's entry screen and click on the Upload Travel Expense Form button, which takes you to the upload view. Click on Browse, navigate to the form on your hard drive, and click on OK.

[7] When Adobe Reader opens, the Interactive Forms technology activates certain features in the Adobe Reader that allow you to edit the form.

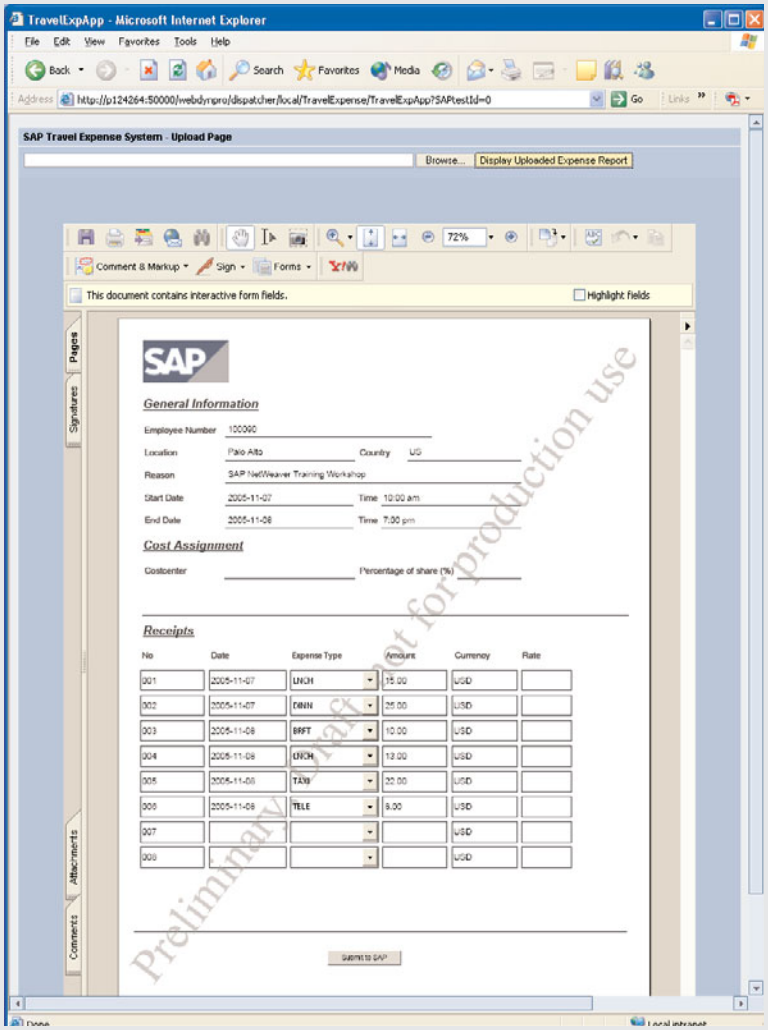


Figure 14 Upload view of the completed PDF form displayed in the browser

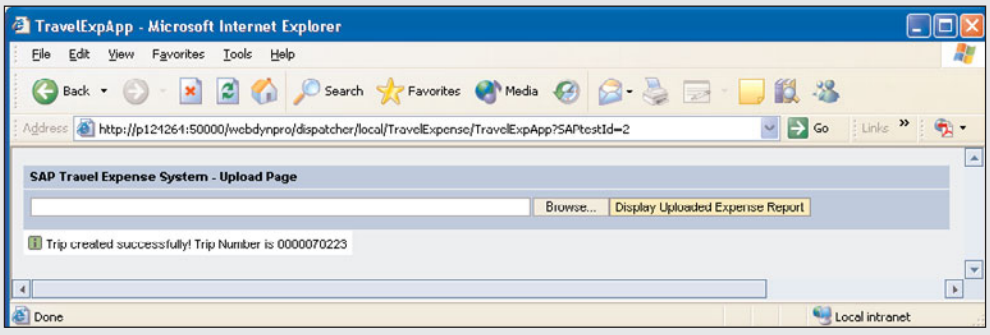


Figure 15 Success message with trip number displayed in the browser

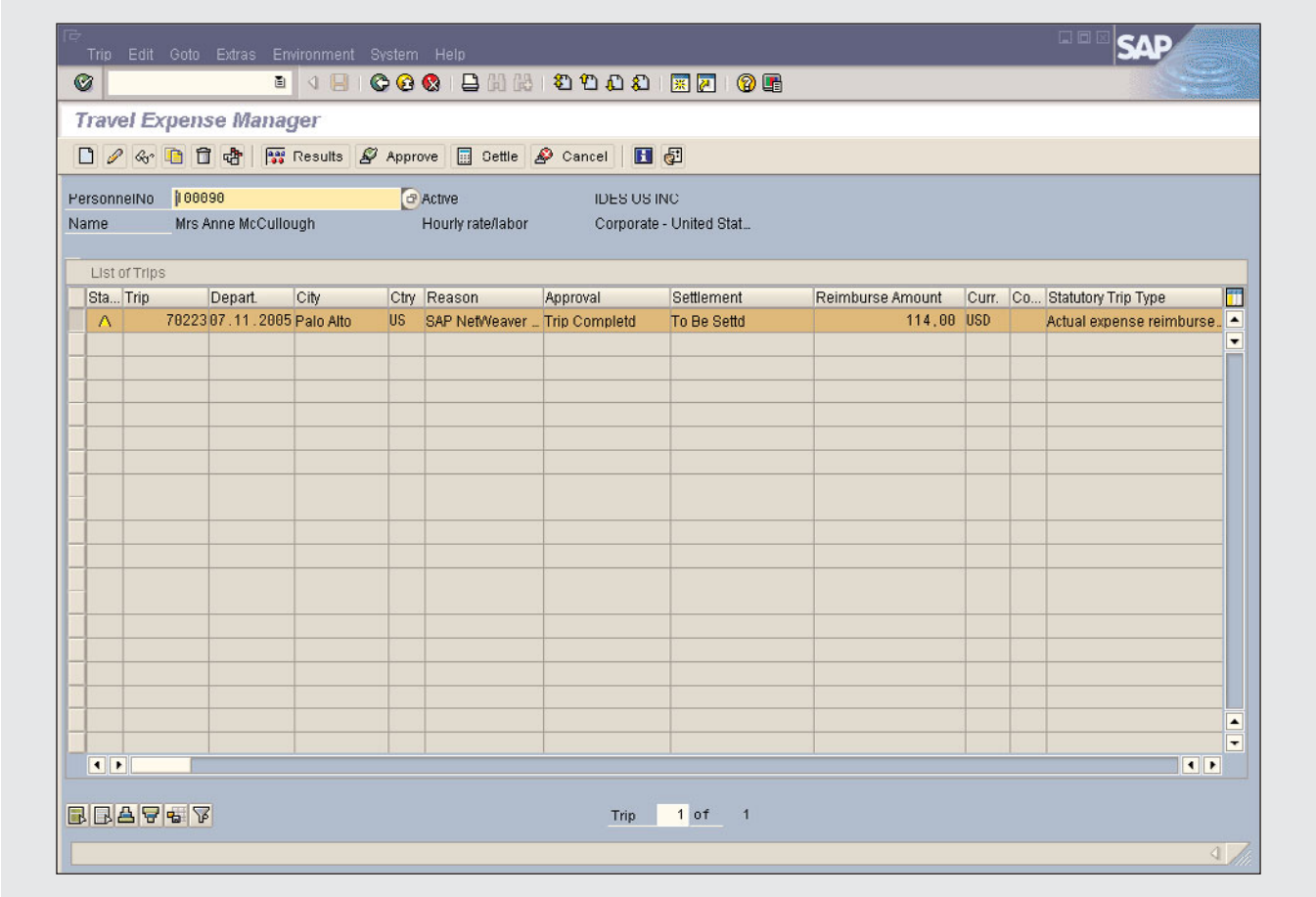


Figure 16 Travel Expense Manager

Before you submit the data to the back-end system, you want to ensure that you uploaded the correct form containing the correct information. Click on the Display Uploaded Expense Report button (Figure 9), which uses the ShowFormView, and the Web Dynpro application displays the form you just uploaded (see **Figure 14**). At this point, the form's content is still local (i.e., on your computer), but it is wrapped in the Web Dynpro application. The data has not yet been submitted to the back-end system, so you can still make changes to the data you entered offline in Adobe Reader.

4. **Validate and submit the form data.** Once you have verified the accuracy of the data in the form, click on the Submit to SAP button, which triggers the corresponding action in the Web Dynpro application. The application establishes a connection with the SAP R/3 system, calls the BAPI, and passes the data from the form to the BAPI.

   If the submit process is successful, the SAP R/3 system creates a new trip based on the data it receives and sends the success message with a corresponding trip number to the Web Dynpro application, which displays in your browser, as shown in **Figure 15**.

5. **Verify that the data was posted.** Last, but not least, let's check to make sure the trip was actually created in the SAP R/3 system. Log on to the back-end system, go to the Travel Expense Manager (transaction PR05), and enter the employee number you used on the form (Figure 13). Highlight the trip in the list of trips, as shown in **Figure 16**, and

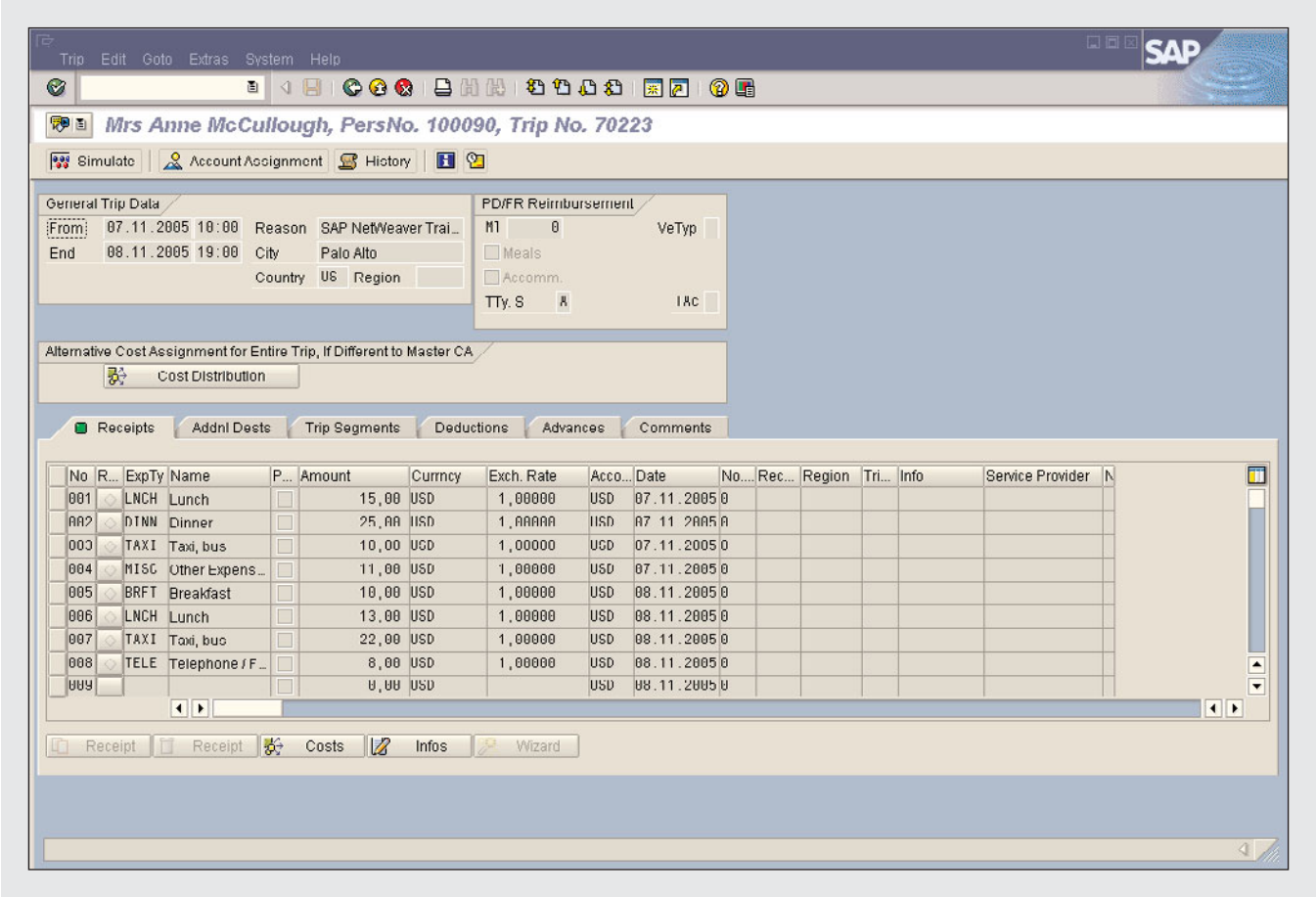


Figure 17 Displaying the trip created in the back-end SAP R/3 system

then click on the display button ( ). The result should resemble **Figure 17**.

Tips and Tricks

Here are a few hints to make your Web Dynpro/ Interactive Forms development project more successful:

- **Create both an online and offline scenario to ensure optimal usability of your application.** In the online scenario, the user logs onto the company portal, for example, and at the click of a button triggers the generation of a PDF that is automatically populated with relevant system data, such as the user's personnel information, trip data (assuming the trip has been approved and the data is known to the system), etc. This generated PDF can then be provided for download, similar to the example scenario described in the article, the difference being that the PDF in our example is generic and empty. In the online scenario, the mode property of the InteractiveForm UI element would be generatePdf; in the offline case, it would be usePdf (as you saw for the example in Figure 10).
- **When you create the data binding in Adobe LiveCycle Designer, try to use "implicit" binding as much as possible.** Instead of dragging and dropping individual fields onto the form from the Data View ("explicit" binding), use groups of fields to build the form. This ensures that the binding of subordinate nodes to the corresponding XML

elements is automatic, which improves performance during the generation of the document.

- **Be aware of the effects of scripting in a form.** Although it is technically possible to transfer much of the business logic into a form, there are two reasons to be careful. First, you want to keep your business logic in your business application (i.e., inside SAP) as much as possible. Second, scripting has detrimental effects on the time it takes to generate a PDF. (This may be irrelevant, depending on the particular scenario, but it is important to consider.)

- **Use subforms to structure form content.** Subforms in Adobe LiveCycle Designer provide anchoring, layout, and geometry management for form objects. The objects in a subform can be arranged in rows, columns, or some other kind of balanced arrangement, and thus make control of your form content (data) easier.

- **Use the Hierarchy View to select form content.** Although Adobe LiveCycle Designer does an excellent job of previewing what a form will look like, some objects on a form are not readily seen without inspecting the hierarchy. The form layout may also make it difficult to select a specific object. When you work with the hierarchy, operations such as resizing or copying are much easier.

Conclusion

As you can see from the sample application detailed in this article, Interactive Forms based on Adobe software in SAP NetWeaver provides capabilities not available in the SAP world before: You can run parts of your SAP-based business processes outside of your system, providing greater convenience to your users. At the same time, you need not worry that incorrect information will be transferred to the back-end system during the offline step, because the process to achieve this transfer is fully automated and runs on the server.

SAP Web AS contains all the tools (SAP NetWeaver Developer Studio, Adobe LiveCycle Designer) you need to set up forms-based processes using PDF documents, and it provides a runtime environment that integrates technology exclusively provided by Adobe (i.e., the Adobe document services) and shipped as an integral part of SAP software.

Integrated into SAP's design-time environments, Adobe LiveCycle Designer is a user-friendly forms design tool that allows business experts to create the form layout most suitable for their business process. Software developers can therefore focus on the more complex tasks of modeling and developing the data retrieval and implementation of the process.

The Adobe document services are fully integrated into SAP's runtime environment in the Java stack and provide a wide range of form and document creation and manipulation functions. You can generate interactive PDF forms that contain data from your back-end system and data your users need to pass back to the system.

Although the first version of Interactive Forms in SAP NetWeaver '04 provides everything you need to create interactive forms-based scenarios using Web Dynpro for Java — or non-interactive output forms in the ABAP Workbench — SAP NetWeaver 2004s adds more types of integration into SAP software and important new functions.

In SAP NetWeaver 2004s, Web Dynpro for ABAP will take advantage of the ABAP integration of Interactive Forms and provide all the well-known benefits of the Web Dynpro framework in the ABAP back end as well. In addition, Guided Procedures, the process component of the Composite Application Framework, leverages the capabilities of the Adobe-based forms solution for the handling of offline business task management.

Additional new features include:

- A wizard that simplifies the creation of tables in forms
- The ability to dynamically add elements, such as additional table rows, to the form after it has been generated
- A zero-install integration of Adobe Reader into the Web Dynpro environment, which uses the

HTTP protocol instead of ActiveX components and therefore provides true cross-platform support

If you want to familiarize yourself with Interactive Forms, but aren't ready for a full SAP NetWeaver installation, there's help waiting on the Internet. Log onto the SAP Developer Network (SDN) at http://sdn.sap.com, go to the Downloads section, and download the Java Edition of the SAP NetWeaver '04 Sneak Preview. It includes the complete SAP Java stack and Interactive Forms. To get off to a good start with your Web Dynpro development, also get the Interactive Forms tutorials available on the Web Dynpro page at SDN. And don't forget to check the Interactive Forms home page in SDN at http://sdn.sap.com/irj/sdn/adobeforms for all the latest information.

Since joining SAP in 1998, Markus Meisl has held a variety of positions on product management and rollout teams. In his current role, he works in the SAP NetWeaver Product Management organization and is responsible for the rollout of SAP forms technology (particularly Interactive Forms based on Adobe software). You can reach Markus at markus.meisl@sap.com.

Marc Chan joined SAP Labs in Palo Alto, California, in 2001. As a Product Manager and a Regional Implementation Group (RIG) Consultant in the US, Marc worked on numerous SAP Web Application Server (SAP Web AS) development projects. One of these projects was the development of the Adobe document services, on which he assisted both the Adobe and SAP development groups. Now based in Hong Kong, Marc is currently a Senior Consultant in the SAP NetWeaver RIG for the Asia-Pacific Region, where his primary focus is rolling out the technical and implementation details of the SAP Web AS technology. He is also responsible for assisting with customer implementations, and relaying the results and challenges of these projects to the development organization to help with future improvements. You can reach Marc at marc.chan@sap.com.

