Döcu Content

DocuWebNews

IBM Notes Domino Designer | Oracle JDeveloper

Knowledge Transfer Documentation

Foreword:
Get ready to be Happy, this Knowledge Transfer documentation can be used in conjunction with prior tutorials submitted to general public via our website to maintain, code, design applications. You will be using IBM DB2 database, IBM Lotus Domino Designer (DDE), and Oracle JDeveloper IDE to completely step through Apps mentioned in this document. Should you need to familiarize yourself with either DDE or JDeveloper, please consult the specific IBM or Oracle website by searching Google or favourite search engine, or visit our website: www.dokollsolutionsinc.com, contact us for support.

System Requirements:

- Windows 7, 8, 10 Operating System
- IBM DB2 Express-C 9.x
- http://www.ibm.com/developerworks/downloads/im/db2express/
- IBM Lotus Notes Domino Designer 8.5.3, 9.x
- https://www.ibm.com/developerworks/downloads/ls/dominodesigner/
- Oracle JDeveloper 12.x
- http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html

Application Launch: Elements Navigator

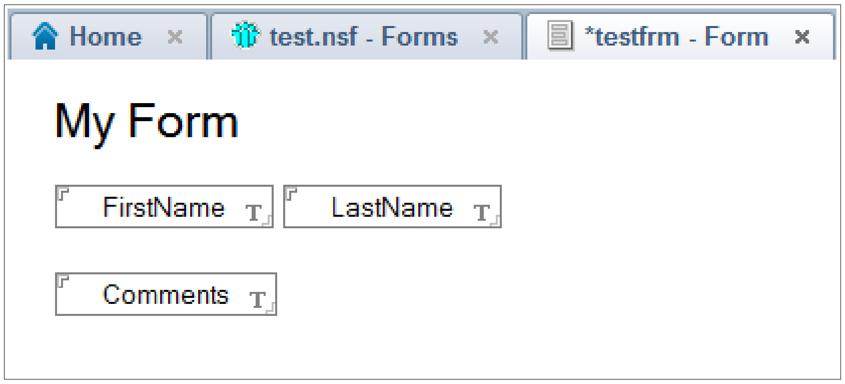
IBM Notes Domino Designer

Step # 1:

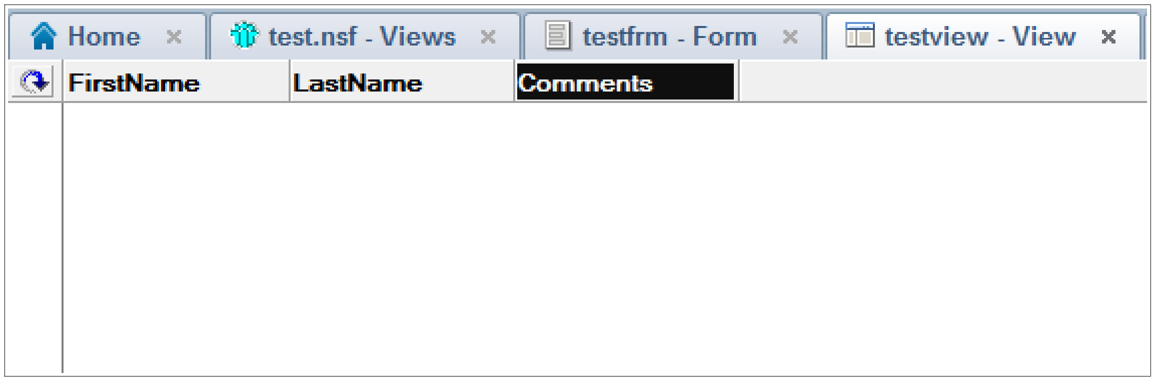
Open and Run DDE software to look at the application from a code and design standpoint. Before going in too deep, you should first make a backup copy of the current App; right-click on App name in Application Navigator, at left of your environment, choose new copy in the menu, name your copy by appending either a date or text to existing name (example: **2016.09.110.6.38.AM_docucontent.nsf**), wait for it to complete saving.

Let's pay a quick visit to the Design of a basic Form and View, get a taste of main back-end elements involved in creating DDE applications. We will also look @ an established application to see forms and views and how they interact with Xpages... areas of interest, as usual, will be highlighted for your convenience

Form Design



View Design



Note: If you are creating an App from scratch, it is a good idea to name your Views similarly to your Forms, so you can easily match them up, it can get really confusing, especially when you have dozens of Forms and Views to look though. You might want to do as the following <u>Form Name</u>: 'Company', or 'frmCompany" <u>View Name</u>: 'CompanyView' or 'vwCompany', however you want, but at minimum, reference similar text in both naming conventions.

<u>Steps to already established application</u>
Form, View and Xpage

Döcu Content

Forms Element

1. Go to the Application Navigator,
2. expand or double-click the Forms element, to reveal a list of forms in the middle pane,
3. choose the form design from the list of forms
4. double-click to open form named DB2Report...
5. right-click on DB2Report form to see Form Properties
6. drill down to various options, example: colour, table options, and security settings...

Views Element

1. expand or double-click Views element from the App Nav, to launch list in middle pane-
2. choose view that matches the form you were looking at prior, example DB2Log.
3. select the View Design and look at its properties below,
4. you should be able to see the Form associated with that view
5. selecting a column in the View Design up above,
6. right-clicking on that column will bring up its column properties,
7. here again, you can drill down to all options for the View column
8. Repeat 5, 6, 7 for other columns

Xpages Element

1. Double-click to open Xpages element,
2. here you have the option to either look at the design or the source of an Xpage
3. look bottom left of the middle pane after launching your Xpage
4. With the Xpage showing, look below for its properties
5. You can see the form or view it is associated with
6. Look at right, see Xpages controls, buttons, labels, etc...

Food for Brain:

Xpages applications can be written by simply dragging and dropping components or controls onto Xpages forms (associated to Views/Forms in the background). However, to fully experience the power of this framework, and implement better coding practices, developers should write own code, separate from the generated XSP code; examples of code that can be written to enhance your Xpages applications are Server Side JavaScript (SSJS) and Java. If writing SSJS code, it is advised to also keep code separate from the design, example adding code to a Script Library, as opposed to including it within the XSP design.

For this exercise, we are using Java to communicate with Xpages, if you know the background of Xpages, you are already aware the Xpages framework is built on top of Java Server Faces (JSF). While Xpages and JSF may differ in some aspects, you will find both work hand in hand with Java, thus making Java, in our opinion, the preferred language to use with Xpages, for its robustness, easy of use, security features, and so on...

We are now going to traverse the coding portion of the current application, for the purpose of the Knowledge Transfer documentation, by briefly discussing the XSP source, then going to Package Explorer where all Java code, and other technologies reside.

Application: Design, Source Navigator, Package Explorer

IBM Notes Domino Designer

Step # 2:

In IBM Notes Domino Designer when Xpages forms are created they generate a skeletal XSP code, bare-bones, that performs almost no function until it is tempered with (in a positive sense of the term) and given some purpose. An Xpage form can also be created by referencing a form or view from the NSF back-end which helps create DataSources via the Drag and Drop feature, to interact with Xpages. With respect to a form, the Xpages controls are linked to back-end form or view fields (DataSource fields), example document1.FieldName, and so on... When coding in Java, the DataSource name shown here is therefore ClassName.fieldName; in some cases, Java code is made available on Xpages form by calling a Script, beforePageLoad or afterPageLoad.

Let's pay a quick visit to the Design and Source of a normal Drag and Drop Xpages form, then look at a different Xpage with Java code present, areas of interest, again, will be highlighted for your convenience

Xpages Skeletal Code

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
        <xp:label value="Hello World" id="label1">

        </xp:label>

</xp:view>
```

Döcu Content

Xpages Design, Source

1. Open SiteFinal Xpages Form in the Döcu Content App series, look for the following XSP code tag <xp: dominoDocument var="document1" formName="Site"></xp:dominoDocument>
2. Follow the variable DataSource name document1 to find all fields connected to that DataSource name, example: SiteName, SiteNumber, and so on...
3. <xp:inputText value="#{document1.SiteName}" id="siteName1"></xp:inputText>
4. Scan the rest of the XSP code to find the button control, in this case, <xp:button ...>
5. The button runs a piece of JavaScript code 'EventHandler' to perform a Submit function- Same code is used in Java, but with a slight twist...

Xpages Design, Java Packages

1. Now, Open a different Xpages form with Java code as DataSource, example 'xpaddversiondata'
2. Look at the Source code, find Java class name 'SendAppNewUserIssuesJavaBean'
3. Then find userName field, in this case, the first letter is lowercase (Will explain why later)
4. <xp:inputText value="#{SendAppNewUserIssuesJavaBean.userName}" id="siteName1"></xp:inputText>
5. Continue down to the button Tag <xp:button …/> and you can see a dramatic difference
6. EventHandler includes an action Script, calls Java code, performs submit <xp:this.action><!

[CDATA[#{javascript:SendAppNewUserIssuesJavaBean.submitVersionData()}]]></xp:this.action>

7. Go to Package Explorer (Window + Show Eclipse Views + Package Explorer)
8. You want to look for the userName field/variable and the button above (submitVersionData)
9. Pin the Explorer to your area of choice, Hint: put it @ right, away from Application Navigator
10. Expand docucontent.nsf App, then WebContent/WEB-INF/src from Package Explorer
11. You should now see all Java Packages, containing Java classes for this Application
12. Expand com.dokoll.solutions.inc.developement.Utils Package
13. Scroll down to find and double-click **SendAppNewUserIssuesJavaBean** class
14. Search for userName field, in this case here it is called a variable
15. You will find that the variable is referenced in declarations as private String UserName;
16. Resulting to Getter and Setter methods such as **public String getUserName** and **public void setUserName**, UserName = userName is what sets the variableName on the Xpages form
17. Drill down to the button method submitVersionData() take a look at the code and the referenced userName variable, submitDocument.appendItemValue("userName", "UserName");

Application: facesconfig.xml Package Explorer/ WebContent

IBM Notes Domino Designer

Step # 3:

Xpages forms, when created, generate a configuration resource file called '**facesconfig.xml**' located further below in the Package Explorer Window; an additional WebContent folder is created to allow other technologies to be referenced in Xpages Apps, example HTML code, CSV files. For now, you are mostly concerned with the **WebContent/WEB-INF** directory which contains the facesconfig.xml. This XML configuration is extremely important because it registers Java classes such as SendAppNewUserIssuesJavaBean to work the right Xpages form. The package name containing the Java class is plugged into ManagedBean tags in the configuration file, if this process is not included, you will receive a Browser error, stating the package is not recognized by the Xpages form in question.

Use below steps to take a closer look at how the facesconfig configuration file works, areas of interest will be highlighted for your convenience.

Döcu Content

Xpages Design, facesconfig.xml

1. Go to Package Explorer
2. Expand docucontent.nsf App, then **WebContent/WEB-INF/src** from Package Explorer
3. Look further below the packages for an additional **WebContent/WEB-INF/** directory
4. Expand **WebContent/WEB-INF**
6. You should now see all facesconfig.xml, double-click to Open it
7. Search for the Java class mentioned in this Knowledge Transfer documentation
8. **<managed-bean-name>**...**</managed-bean-name>** contains the name of the class
9. **<managed-bean-class>**...**</managed-bean-class>** calls the full Java package name to make

available Java code used on the Xpages form

10. thus, com.dokoll.solutions.inc.developement.Utils.SendAppNewUserIssuesJavaBean

Future Design, or enhancement tips

With the above steps, you can pretty much update an Xpages application without much effort. Example, you would first update the form in question by adding a new field, say it were 'UserRole', then you would go to the view your form is connected to, add a column on the Xpage, drag or copy an inputText control... in the Java class you would declare a new variable for UserRole, make available Getter and Setter methods, and processes to create a new document'
submitDocument.appendItemValue("userRole", "UserRole");

At first glance, you may be thinking, Geez, how the heck am I going to do this. Rest assured, Step by Step instructions are available throughout prior tutorials. If support is needed to navigate Forms, Views, Xpages elements in DDE, watch https://www.youtube.com/watch?v=Ah8j3Ae25f8 for support. If you prefer, do a quick search in Google, or use your favourite search site for 'How to create an Xpages Application'; as always, you can refer to our website (www.dokollsolutionsinc.com), contact us for additional support.

Application: Design, Source, Package Explorer

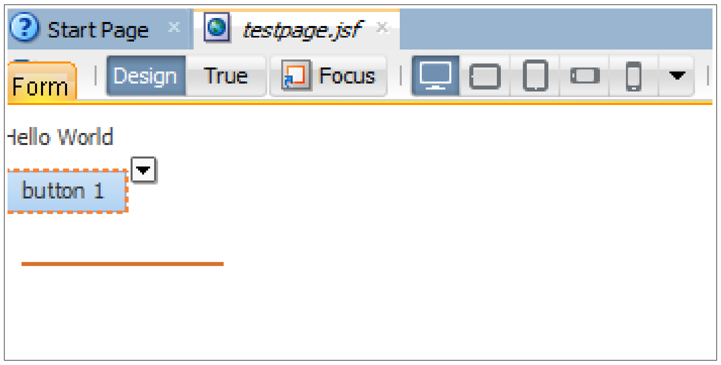
Oracle JDeveloper

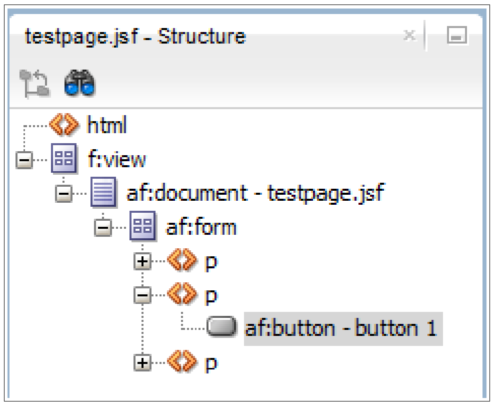
Step # 1:

Open and Run JDeveloper software to look at the application from a code and design standpoint. Before going in too deep, you should first make a backup copy of the current App. There are a number of ways to go about this, for now just go to File + Save As in the above menu, name your copy by appending either a date or text to existing name (example: **2016.09.110.6.38.AM_docuwenews.jsw**), wait for it to complete saving.

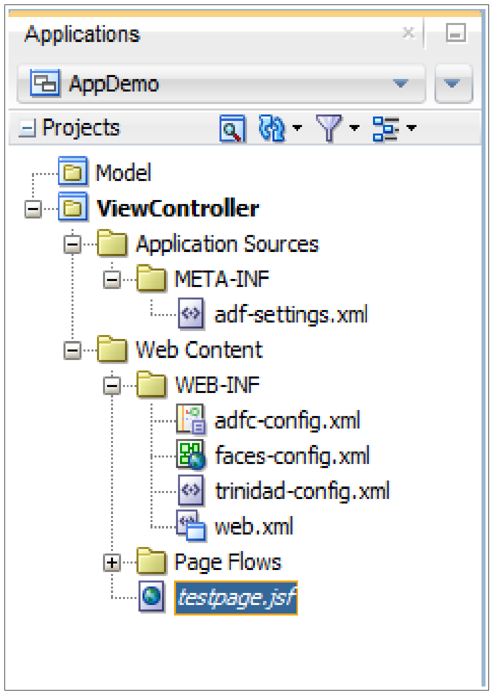
Let's pay a quick visit to the Design and Source of a simple Drag and Drop ADF/JSF form, then look at the ADF/JSF Source without XML data but with Backing Bean code generated by JDeveloper, areas of interest will be highlighted for your convenience

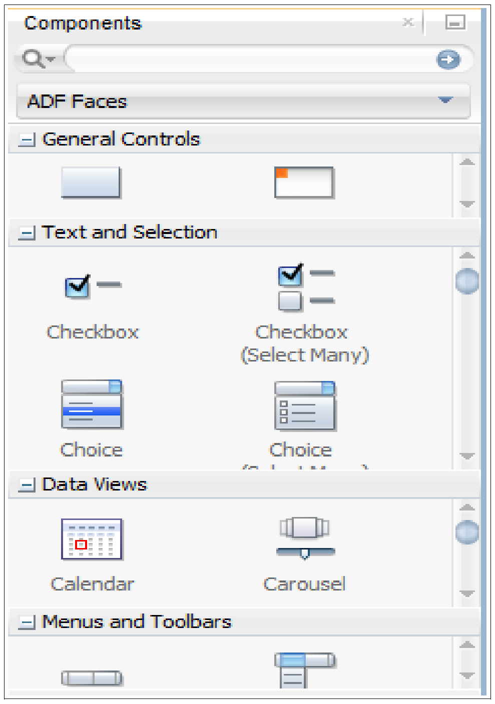
Form/Page Design and Structure





Applications Explorer Window/Components Palette





Now, look into the steps taken to create the above ADF/Faces application:

Döcu Content

Page Views

1. Go to the Applications Explorer, left of the JDeveloper IDE
2. Expand or right-click on ViewController
3. Create a Facelets Page, name it appropriately as Blank Page Layout
4. Look at right for the components Palette, expand, browse, choose from sections...
5. Drag and Drop needed controls, bind inputText, outputText, button, etc...
7. Right-click and run the page in your Browser

Note: At this point JDeveloper creates a backing bean file, named similarly to the ADF/JSF page you just created. However, since this is a basic Drag and Drop page without database interaction the page is static in nature. In this exercise, we wanted to demonstrate how to create a page from scratch using the features in the JDeveloper IDE.

Food for Brain:

ADF/JSF Apps in Oracle JDeveloper are designed similarly to those in IBM Notes Domino Designer. An ADF/Faces components palette is available to drag and drop controls, data views, layouts on pages to complete full applications. The difference here is that most of the underlying configuration is done behind the scenes to allow for rapid application development with minimal intervention by the developer. Further, you can connect to a database and bind entities to page controls to create a data-driven application from scratch within minutes.

Application: Design, Source, Components Explorer (Continued...)

Oracle JDeveloper

Step # 2:

In JDeveloper, developers can choose to drive their applications using Java Server Faces (JSF), Application Developer Framework (ADF) or both at the same time, forms are created, a skeletal JSF and Backing Bean code are generate, information that be enhanced to interact with back-end databases; Form controls such as inputText, button, etc... are therefore linked to Backing Bean code to populate database columns on the fly, when ADF/JSF page runs in Browser. To better understand this scenario, you can create a complete Drag and Drop ADF/JSF application with database entities; we will cover Drag and Drop applications as ADF/JSF samples in other tutorials. In this exercise, we are mainly interested in using a skeletal ADF/JSF (although we are using the Drag and Drop features to some degree), we will attempt to actually write code that drives the current sample by modifying its Backing Bean to interact with a DB2 database back-end, and XML data.

Let's pay a quick visit to the Design and Source of a simple App, created with simple Drag and Drop ADF/JSF features to form, then we will look at the Source of the established ADF/JSF design, peek at the Backing Bean code to be modified... areas of interest will be highlighted for your convenience

JSF Skeletal Code (Example)

```
<html lang="en"
      xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Facelets Hello World</title>
    </h:head>
    <h:body>
        #{hello.world}
    </h:body>
</html>
```

ADF/Faces Skeletal Code (Example)

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE html>
<f:view xmlns:f="http://java.sun.com/jsf/core" xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
  <af:document title="testpage.jsf" id="d1">
    <af:form id="f1">
      <af:activeOutputText value="Hello World" id="aot1"/>
    </af:form>
  </af:document>
</f:view>
```

Döcu Content

Application Development Framework (ADF/JSF) Design, Source

1. Open DocuWebNews ADF/JSF Form **WebContent** for Döcu Content App series,
2. Look @ ADF/JSF code tag <af:document title="**webnewsdbfeeds.jsf**" id="d1"
3. ADF/JSF tag binds Backing Bean code with ADF/JSF page, via variable/control item binding="#{backingBeanScope.backing_webnewsdbfeeds.d1}">
4. You can scan the rest of the ADF/JSF to find the button control, in this case, <af:button …/>
5. Button 'b1_action' runs the Backing Bean, performs a Submit function using - <af:button text="**XML Data to DB**" id="b1"
   binding="#{backingBeanScope.backing_webnewsdbfeeds.b1}"
   action="#{backingBeanScope.backing_webnewsdbfeeds.b1_action}"/>
6. Let's take a closer look, see how this actually works in the full Backing Bean class
7. Go to Applications Explorer Window, use the Drop Down to find your AppName...
8. You may have to Expand Projects to reveal Model and ViewController sections
9. Expand ViewController, drill down package com.dokoll.solutions.inc.DocuWebNews.Model
10. See Backing Bean **Webnewsdbfeeds.java** class for the button method '**b1_action()**'
11. Read Backing Bean method to see what it does with XML URL from IBM Notes Domino App
12. Venture into the Query used to Submit XML records to DB2

Future Design, or enhancement tips

With the above steps, you can pretty much update an ADF/Faces application without much effort. Example, you would first update the ADF/JSF design in question by adding/dragging a new inputText control, you can name it, as one example, 'UserRole', then you would go to its Backing Bean, bind it with already declared variable for UserRole; Getter and Setter methods would have been generated by JDeveloper on your behalf... run it and test in your Browser, make sure code compiles okay

At first glance, you may be thinking, Geez, how the heck am I going to do this. Rest assured, Step by Step instructions are available throughout prior tutorials or Journal Entries. If you prefer, do a quick search in Google, or use your favourite search site for 'How to create an ADF/Faces Application'; as always, you can refer to our website (www.dokollsolutionsinc.com), contact us for additional support.

Conclusion:

You are now able to step through Xpages and ADF/JSF Apps using Drag and Drop features and Object-Oriented programming with JavaBean, Backing Bean to submit XML data to DB2 database via Oracle JDeveloper, from a URL located on IBM Notes Domino App. For all Questions and comments, please add a Quick note to our Contact form, or visit our social media networks.

Contact
http://www.dokollsolutionsinc.com/apptrendscontactemail.php

Facebook
https://www.facebook.com/Dököll-Solutions-Inc-233555900032117/

Google+
https://plus.google.com/u/0/+DököllSolutions/posts

Twitter
https://twitter.com/DokollSolutions

YouTube
https://www.youtube.com/channel/UCSImDTpK0oe7QrPsYOE4nww

Dököll Solutions, Inc (Download Döcu Content)
http://www.dokollsolutionsinc.com/DownloadNew.html
Or
http://www.dokollsolutionsinc.com/2016.09.16.9.21.PM_docucontent.zip

OpenNTF (Download Döcu Content)
https://www.openntf.org/main.nsf/project.xsp?r=project/D%C3%B6cu%20Content
%20V3.0/releases/B18525B1223AC8FD86258031000B857D

Dököll Solutions, Inc.
version: 2016.09.21.8.51.PM