Getting Concurrent with Java

Janette Lockhart

impac

Overview

- Reasons to start using Java
  - Powerful
  - Object-oriented
  - Provides classes that make many routine tasks easy – such file manipulation
  - E-Business Suite increasing embraces Java
- Java Concurrent Programs (JCP) are a great way to begin your dive into Java

Objectives

- Learn how to develop and test a JCP from JDeveloper
- Understand how to leverage OA Framework BC4J objects in a JCP
- Learn how to register and run a JCP in the E-Business Suite
- Understand how to call XML Publisher APIs from a JCP

Getting Started

- See Metalink for the correct JDeveloper OA Extension version and download as patch
- For 11.5.10, the base version of JDev is 9.0.3. For R12, the version of JDev is 10g
  - Many different “OA Extension sub-versions” depending on your ATG patch level. 11iRUP4 requires different JDeveloper than does 11iRUP5, etc.
- View the Patch Readme for install steps.

Preparing your JDeveloper Environment for JCP Development

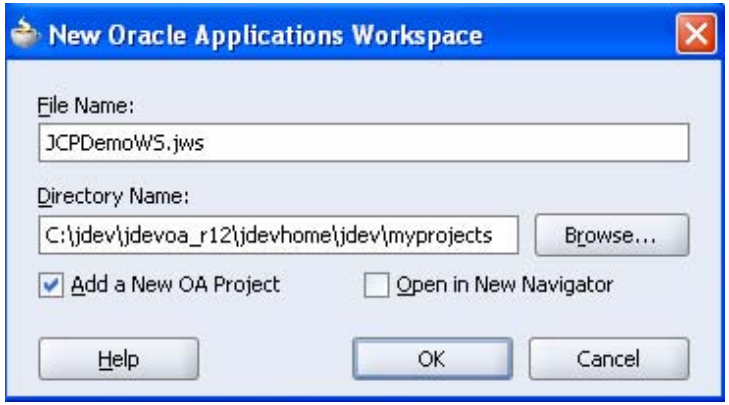
- Download the fnd/cp class files from the middle tier
  - Login to the applications server and set your environment
  - Change directories to $JAVA_TOP/oracle/apps/fnd
  - zip –r cp cp
  - Above command recursively zips the cp directory and creates cp.zip
  - Download cp.zip and unzip into your <jdev_install>/jdevhome/jdev/myclasses/oracle/apps/fnd directory

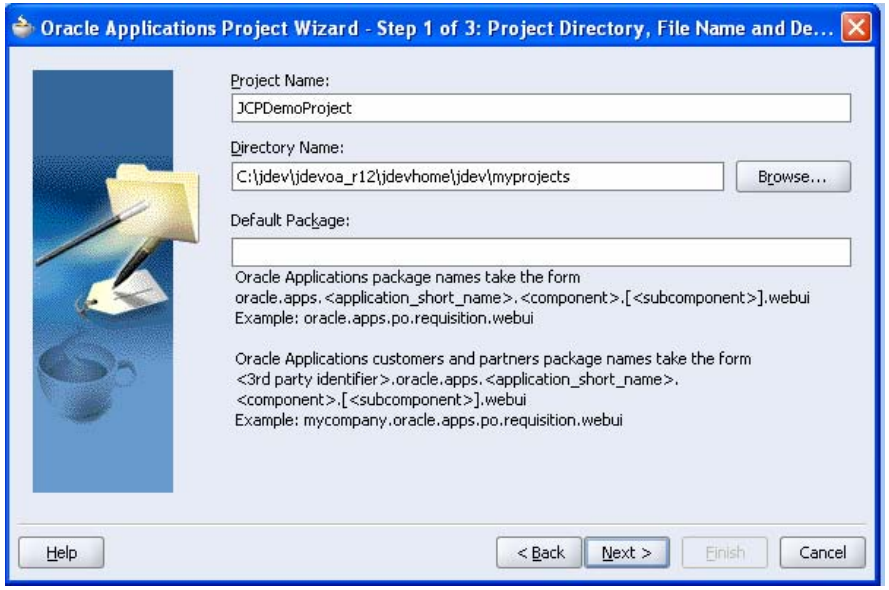
Preparing your JDeveloper Environment for JCP Development

- Download the DBC file for your environment
  - Login to the applications server and set your environment
  - Change directories to $FND_SECURE
  - Download the .dbc file from that directory
  - Place in your <jdev_install>\jdev\dbc_files\secure directory

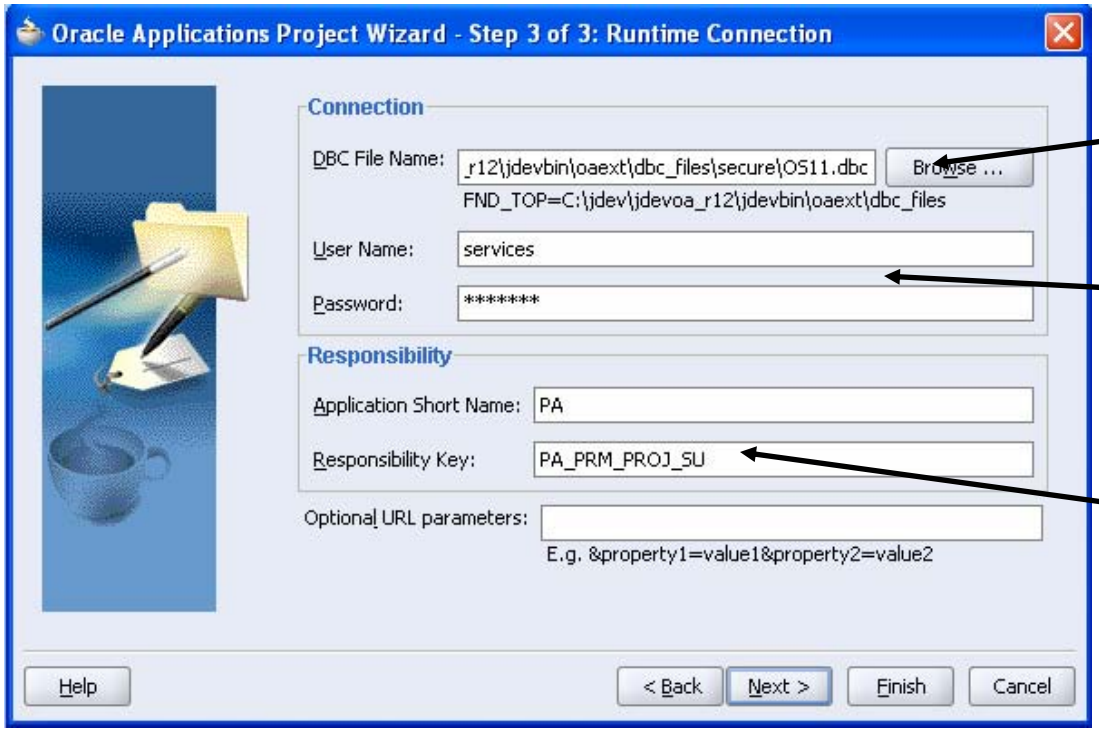
Create an OA Workspace and OA Project

- In JDeveloper's Applications Navigator, right-click the Applications node and select "New OA Workspace"





Create an OA Workspace and OA Project, continued

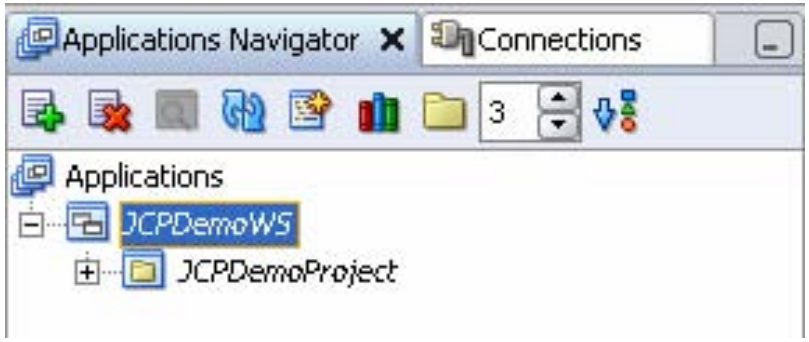


DBC File

Applications Login

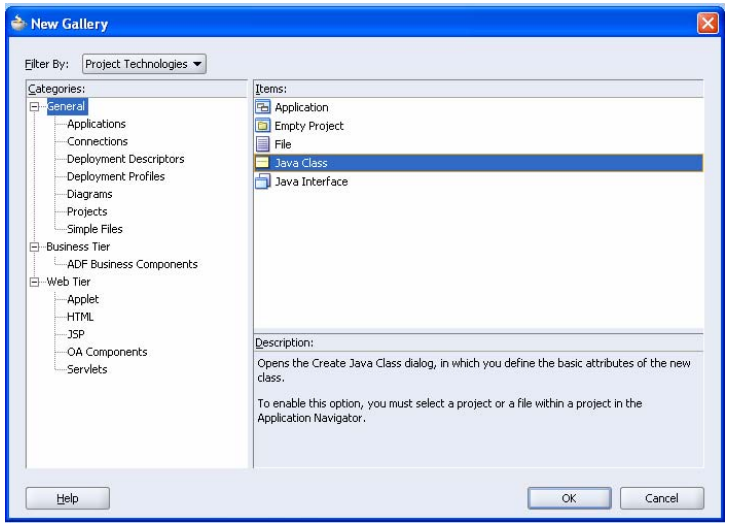
Responsibility Application and Key

Workspace and Project Setup Complete



Your First Java Concurrent Program

- Right-click on JCPDemoProject and choose “New” from the context menu
- In the “New” Gallery, select General => Java Class. Click OK.



Your First Java Concurrent Program, continued

- In the Create Class window
  - Enter JCPDemoClass for Name
  - Enter vis.oracle.apps.xxvis.cp.demo for Package
  - Leave other defaults

Editing the Generated Code

- Add import statements

```
import oracle.apps.fnd.cp.request.ReqCompletion;
import oracle.apps.fnd.cp.request.JavaConcurrentProgram;
import oracle.apps.fnd.cp.request.CpContext;
```

- Add “implements JavaConcurrentProgram”

```
public class JCPDemoClass implements JavaConcurrentProgram
```

- Add program logic

```
public void runProgram(CpContext ctx) {
ctx.getLogFile().writeln("Starting to run Java concurrent program",
    0);
ctx.getReqCompletion().setCompletion(ReqCompletion.NORMAL,
    ""); }
```

Code Explanation

- We have created a class named JCPDemo
- Class resides in a directory structure (package) named vis.oracle.apps.xxvis.cp.demo
- Import statements refer to other Oracle/Java classes that we will reference
- Our class implements JavaConcurrentProgram
  - Which is an interface (or a template)
  - Meaning we promise to provide the code for the methods specified by the interface: runProgram in this case
- The runProgram takes one argument – ctx which is of type CpContext

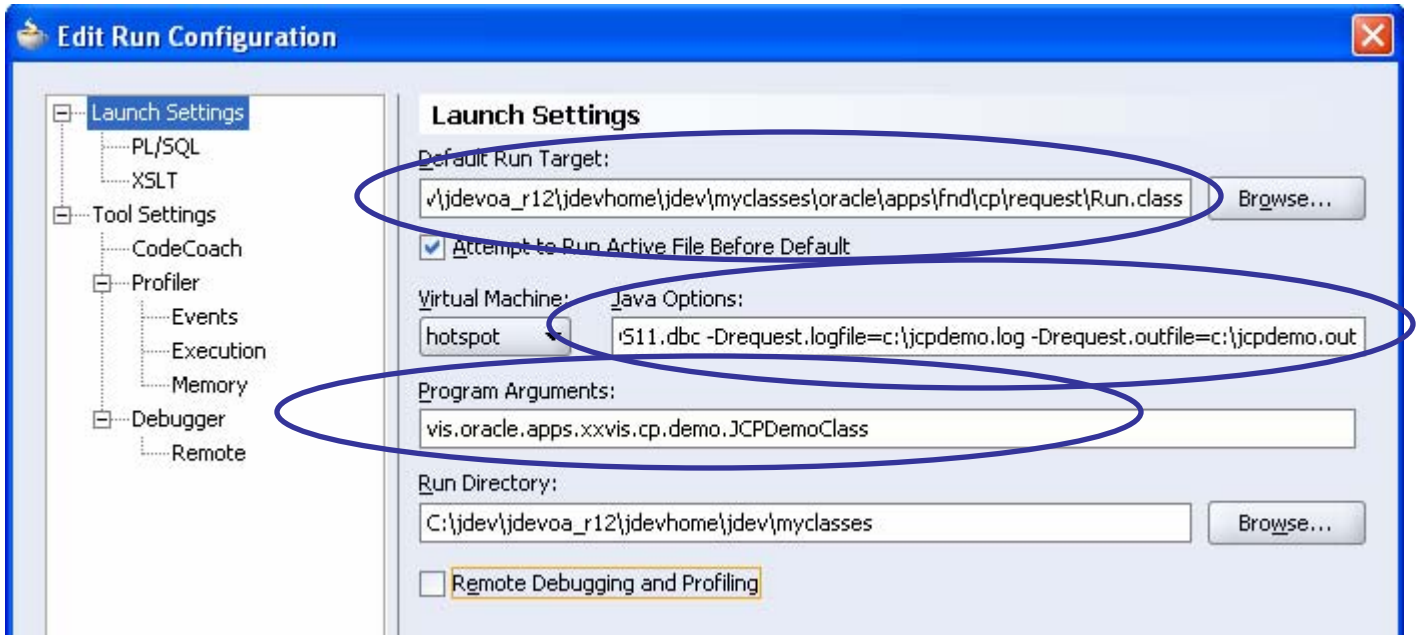
Code Explanation, continued

- CpContext provides us with “context”
  - Access to concurrent request log and output files
  - Access to profile and global values
  - Access to a JDBC connection
- So far, we’ve used CpContext to get a handle on the log file for our request and write some text to the log
- Also, we’ve used CpContext to set the program completion status

Running the JCP from JDeveloper

- Why? Convenient not to have to deploy to middle tier for every test
- We need to set Java Options so JVM can find our DBC File, our log file, and our output file on the desktop
- Right-click the JCPDemoProject
  - Choose “Project Properties”
  - Choose “Run=>Debug”
  - Click the “Edit” button for the default configuration

Running the JCP from JDeveloper, continued



- Java Options:

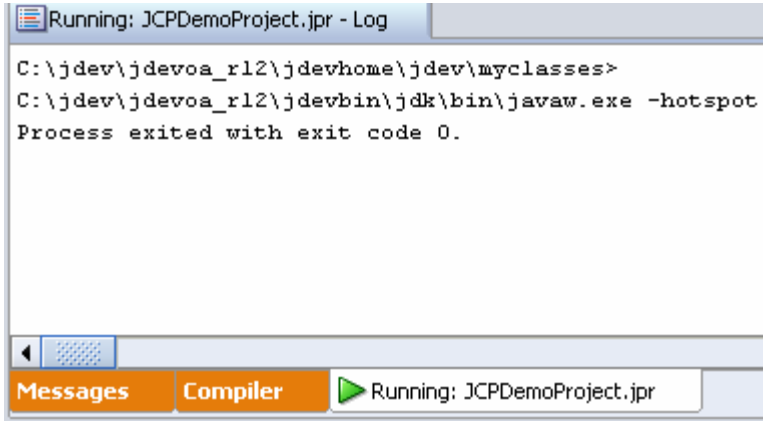
  -Ddbcfile=<jdev_install>\jdevhome\jdev\dbc_files\secure\OS11.dbc

  -Drequest.logfile=c:\jcpdemo.log

  -Drequest.outfile=c:\jcpdemo.out

Running the JCP from JDeveloper, continued

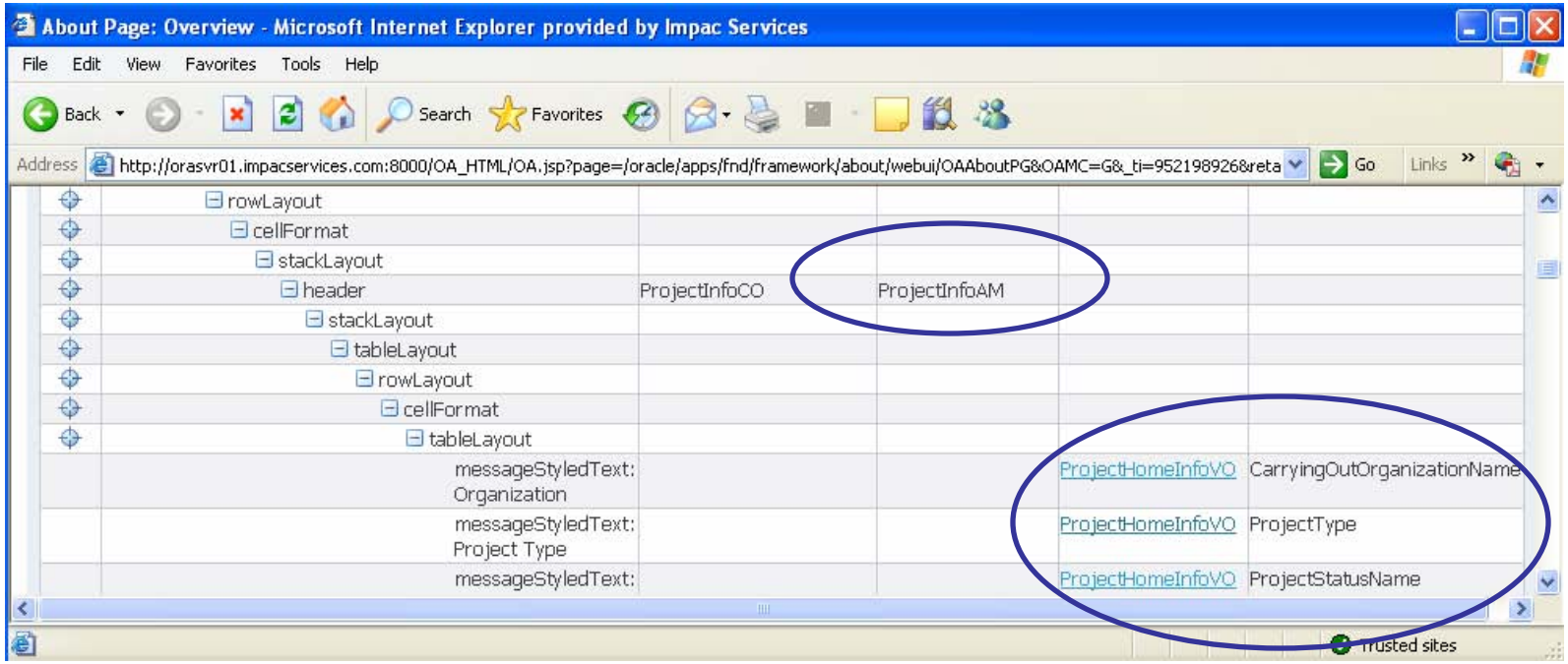
- In the Applications Navigator, right-click the JCPDemo project and select “Run”
- You should get a successful completion message
- Go look for your log file and verify that the text you specified was written



Leveraging OA Framework BC4J Objects in a JCP

- Use “About this Page” to learn about BC4J Objects underlying an OA Framework web page

Leveraging OA Framework BC4J Objects in a JCP



- Application Module is ProjectInfoAM
- View Object is ProjectHomeInfoVO

BC4J Quick Overview

- In Business Components for Java (BC4J)
  - Application Module is
    - A “container” object that gives access to other objects in the model, like the ProjectHomeInfoVO
    - Provides context (similar to CpContext)
  - View Object is
    - An object that queries the database
    - It is defined with a query whose where clause maybe manipulated at run time
    - It returns rows which have attributes representing the “columns” of data returned from the database query
    - Provides a writeXml method

Adding a Method to Build XML

- Name our new method buildXml
- Takes CpContext as an argument
- Creates an Application Module Factory
- Use the “factory” to create the ProjectInfoAM Application Module
- Use the findViewObject method on the AM to get the ProjectHomeInfoVO view object
- Set the VO where clause and bind variable
- Execute query and writeXml to our out file

Adding a Method to Build XML, continued

```
private void buildXml(CpContext ctx)
  {
    String amName = "oracle.apps.pa.project.server.ProjectInfoAM";
    String voName = "ProjectHomeInfoVO";
    OAApplicationModuleFactory amFactory = new OAApplicationModuleFactory();
    OAApplicationModule am = amFactory.createRootOAApplicationModule(ctx,
    amName);
    OAViewObjectImpl vo = (OAViewObjectImpl)am.findViewObject(voName);
    vo.addWhereClause("project_number = :1");
    vo.setWhereClauseParam(0, "B100"); //temporarily hard-coded
    vo.executeQuery();
    XMLNode root = (XMLNode)vo.writeXML(0, XMLInterface.XML_OPT_ALL_ROWS);
    BlobDomain result = new BlobDomain();
    try
    {
     root.print((OutputStream)result.getBinaryOutputStream());
     ctx.getOutFile().writeln(result.toString());
    }   …
```

Adding a Method to Build XML, continued

- Compile and run the project
- Look for the output file which now has the XML
- Notice the root element has the same name as the VO.

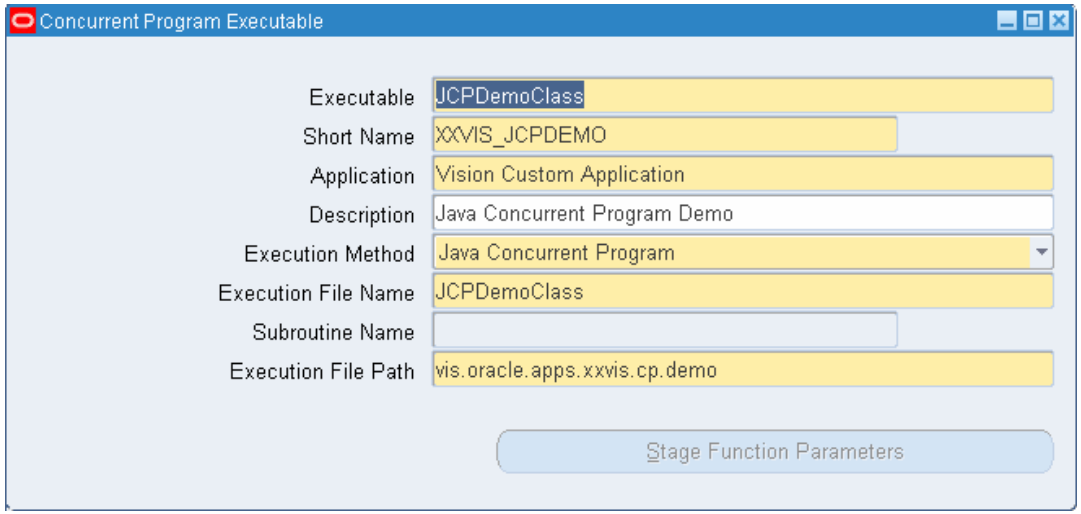
jcpdemo.out - WordPad

File Edit View Insert Format Help


```
<ProjectHomeInfoVO>
   <ProjectHomeInfoVORow>
      <ProjectId>3445</ProjectId>
      <ProjectType>Construction-Cap Int</ProjectType>
      <PersonId>53</PersonId>
      <PersonName>Marlin, Ms. Amy</PersonName>
      <StartDate>2004-01-01</StartDate>
      <CompletionDate>2009-06-30</CompletionDate>
      <CarryingOutOrganizationName>Services-Construction</CarryingOutOrganizationName>
      <ProjectStatusName>Approved</ProjectStatusName>
      <Description>New Building Construction with Capitalized Interest</Description>
      <ProjectName>Building 100 Construction</ProjectName>
      <ProjectNumber>B100</ProjectNumber>
      <ProjectLongName>Building 100 Construction</ProjectLongName>
      <StartDateSrc>A</StartDateSrc>
      <CompletionDateSrc>S</CompletionDateSrc>
   </ProjectHomeInfoVORow>
</ProjectHomeInfoVO>
```

Registering the Concurrent Program in the E-Business Suite

- Register Concurrent Executable
  - Execution Method: Java Concurrent Program
  - Execution File Name: Class Name (without .class)
  - Execution File Path: Package name



Registering the Concurrent Program in the E-Business Suite

- Register Concurrent Program
  - Set Output Format to “XML”
- Assign to Request Group
- Test!

Calling XML Publisher APIs from JCP

- Demo requirement is to
  - Programmatically apply XMLP template
  - Burst (split) the output by project number
  - Email resulting PDF files
- Note: in R12 bursting comes out of the box, no need for a custom JCP
- Bursting operations are driven by an XML control file
- Download the oracle/apps/xdo class files

Bursting Control File

```
<?xml version="1.0" encoding="UTF-8"?>
<xapi:requestset xmlns:xapi="http://xmlns.oracle.com/oxp/xapi">
<xapi:request select="/ProjectHomeInfoVO/ProjectHomeInfoVORow">
<xapi:delivery>
<xapi:email server="vansmtp.impacservices.com" port="25"
    from="jlockhart@impacservices.com" reply-to
    ="jlockhart@impacservices.com">
<xapi:message id="123" to="jlockhart@impacservices.com"
    attachment="true" subject="JCP Demo Test">Please review the
    attached document.</xapi:message>
</xapi:email>
</xapi:delivery>
<xapi:document output-type="pdf" delivery="123">
<xapi:template type="rtf" location=
    "xdo://XXVIS.XXVIS_JCPDEMO.en.US/?getSource=true" >
</xapi:template>
</xapi:document>
</xapi:request>
</xapi:requestset>
```

Calling the Bursting API from our JCP

- Add a new method to our class
  - Get the Custom Top path where our bursting control file resides
  - Get the request output file path and name, which is our XML data source
  - Instantiate a Document Processor object
  - Call the process() method on the Document Processor

Calling the Bursting API from our JCP

```
private void emailOutput(CpContext ctx){
    //get the path to our custom top directory
    String sCustomTop = ctx.getAppsEnvironmentStore().getEnv("XXVIS_TOP");
   //concat the directory path and file name for the bursting control file
    String sCtlFile = sCustomTop + "/jcpdemo_ctl.xml";
    try {
      //get the file name for the conc request output file
      String sXmlFile = ctx.getOutFile().getFileName();
      //call the document processor, passing control file and xml file
      DocumentProcessor dp = new DocumentProcessor(sCtlFile, sXmlFile);
      dp.process();
    }
    catch (Exception e)
    {
      ctx.getLogFile().writeln(e.getMessage(), 0);
      mReqStatus = ReqCompletion.ERROR;
    }
  }
```

Conclusion

- We've learned how to
  - Write and run a JCP from JDeveloper
  - Instantiate OA Framework BC4J objects and use them to generate XML output
  - How to register a JCP in the E-Business Suite
  - How to call XML Publisher APIs from a JCP
- We were able to do a lot with very little code!

Note: The complete program code is in the Appendix in the paper that will be available on the OAUG web site