APEX DIGITAL SYSTEMS

J2EE Architecture Notes

This document was originally an email addressed to Manuel Mattke, Stephanie Liu, and Frank Rusch answering Manuel's J2EE architecture questions.

Jeff Doucette
June 23, 2005

APEX DIGITAL SYSTEMS

1) What are the pieces that make JMS work?

a) Simple example:
http://java.sun.com/products/jms/tutorial/1_3_1-fcs/doc/j2eeapp1.html

This is a simple two-piece scenario, with a sender application and a listening message-drive bean. Both objects import the javax.jms.* libraries to access the JMS API. The steps to getting this up and running, as digested by me:

i) write the sender
ii) write the listener
iii) compile into an .ear file
iv) start the J2EE app server environment; start deployment tool
v) create the queue
vi) point the app server to the .ear file's directory; create application
vii) package the sender and listener (separate)
viii) deploy

So really, there isn't a whole lot to do outside of the app server's GUI. the process flow goes sender -> listener MDB, and the rest is managed.

b) Complex example:
http://java.sun.com/products/jms/tutorial/1_3_1-fcs/doc/j2eeapp3.html

Again, the map found on this page is complicated, but the only objects to be coded are the client app, the 3 MDBs, and the one entity bean. the rest is defined on the app server. this is a neat example b/c it demonstrates an entity bean being collected at the end of its usefulness in the process.

2) Is JDBC analogous to ODBC or SQL*Net?

It depends on which driver is used. Some drivers ("Type 1") simply connect to an ODBC DSN; others ("Type 2") use middle-tier platform-specific libraries but avoid ODBC; others still ("Type 4") are pure 100% Java that can connect directly to the database, and can connect to remote DBs as well as local ones. (See http://java.sun.com/products/jdbc/driverdesc.html for more.)

The Type 4 drivers are intended for applets, while the Oracle OCI driver is the analogous client-server development platform: it uses Net8 to connect to the DB (local or remote, as we already discussed). The enclosed link is a brief explanation: http://technet.oracle.com/doc/java.815/a64685/overvw3.htm#1000908

This link connects to Oracle's JDBC FAQ. The question at this anchor is worthwhile; the rest is implementation bugs.
http://otn.oracle.com/tech/java/sqlj_jdbc/htdocs/jdbc_faq.htm#_59_

3) Servlets vs. JSPs vs. Beans

SERVLETS vs. JSPS

Servlets handle HTTP requests, and are often compared to CGI scripts. JSPs are tag-based dynamic HTML viewers. Both technologies can run processing routines, but in the spirit of separating content and

presentation, it's best to keep as much processing as possible away from the JSP. Which is not to say that processing inside the JSP can't happen too.

I found this article extremely helpful; it discusses the philosophical differences between servlet-based and JSP-based implementations. It's dated 1999 but the discussion is still valid.
http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html

These two GIFs, lifted from the above, illustrate the difference in philosophy perfectly:

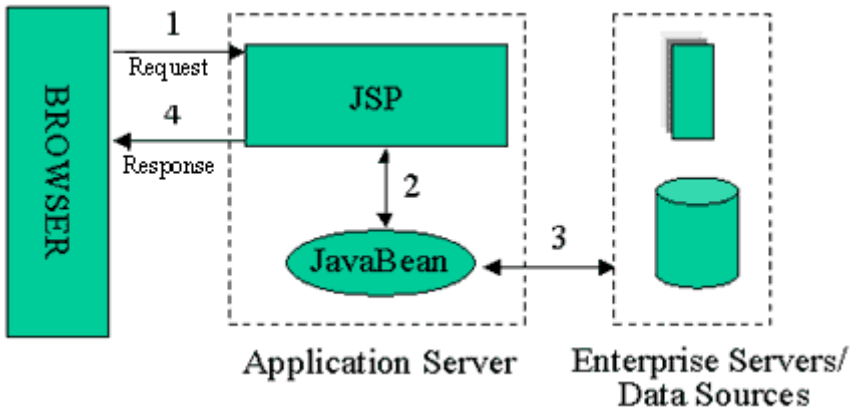


Figure 1: JSP-based model

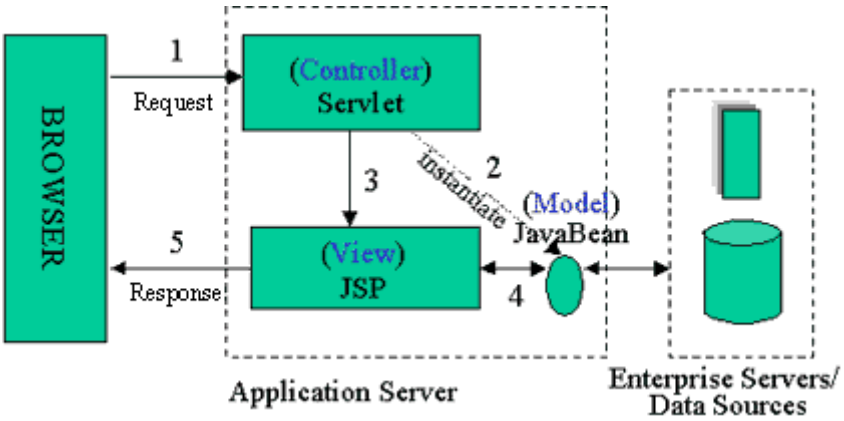


Figure 2: Servlet-based model

For me, the key is this sentence: **"the servlet acts as the *controller* and is in charge of the request processing and the creation of any beans or objects used by the JSP, as well as deciding, depending on the user's actions, which JSP page to forward the request to."** This is what I was trying to verbalize before. The servlet and bean(s) run the processing in tandem, with the servlet deciding what data the JSP needs in order to render itself. By the time the JSP executes, the bean has retrieved the needed data, and the JSP can simply spit out the required information w/o doing a lot of thinking.

JAVABEANS vs. EJBS

A JavaBean is a reusable Java class that implements a "JavaBean" interface. It's analogous to the VB notion of the objects one drops onto a form... e.g. "Button" and its properties would constitute a JavaBean. The concept is called "introspection"... an IDE can analyze a bean based partially on the JavaBean class inheritance and partially on JavaBean programming standards. These beans can be strung together, as in VB, to create a whole. They aren't persistent or independent, really... they are intended as building blocks.

EJBs, meanwhile, are a whole other beast, and unrelated to JavaBeans. EJBs are independent, remotely-accessible chunks of business logic. They run in an EJB container on the app server that manages their security, transactions etc. The applications that an EJB takes part in will share the EJB, rather than including the app's very own copy of the bean code, as seems to be the case with JavaBeans. EJB instances actually persist, containing data that has been given to them by other beans. That data is accessible by future EJB-callers, and killed off when the process is deemed complete. (The complex JMS example above demonstrates this concept pretty well.)

My favorite explanation comes courtesy of JBoss: **"Both JavaBeans and EJBs are units of packaged functionality but they are designed to function in very specific and different environments. JavaBeans facilitate black box reuse of visual and non-visual components within JavaBean-aware IDEs. EJBs on the other hand are non-visual components that can only be deployed in an EJB-compliant Java application server. JavaBeans are basically just classes executed in the JVM, while EJBs are managed objects that are deployed within an EJB application server. JavaBeans may be driven by events, but EJBs are currently just driven by remote method calls."**

For an OK, IBM-polluted explanation, read below:
http://www-900.ibm.com/developerWorks/cn/java/beandiff/index_eng.shtml
Oracle has a nice little EJB summary too which talks more about implementation of EJBs
http://otn.oracle.com/tech/java/oc4j/doc_library/902/ejb/overview.htm#1005620

SERVLETS vs. BEANS

I think we figured this one out for ourselves before you left... servlets are designed specifically to deal with HTTP requests and responses. They're partners much more than competitors though, as the attached GIFs above should demonstrate: servlets can use beans in their own routines if they so choose.

4) JNI: How about languages other than C/C++?

There are options for COM objects in both VB and VC++... BEA WebLogic is on the list, and claims that both directions of integration can happen seamlessly on the following link:
http://www.weblogic.com/docs45/classdocs/API_com.html.

However I'm guessing that Ensemble's C/C++ objects are not running in a COM-able environment. This link lists a few options for integration of Java and COM
http://www.jguru.com/faq/view.jsp?EID=74302

As for Fortran, the following sites make it apparent that Fortran can be wrapped! They all appear to use C as an intermediary though. Regardless, it's within the realm of possibility.

http://userweb.elec.gla.ac.uk/i/iainw/jni/
http://www.math.ucla.edu/~anderson/JAVAclass/JavaInterface/JavaInterface.html
http://www.caip.rutgers.edu/~vincentm/JNI/jni2f.html
http://www-stat.stanford.edu/~naras/java/course/lec5/lec5.html

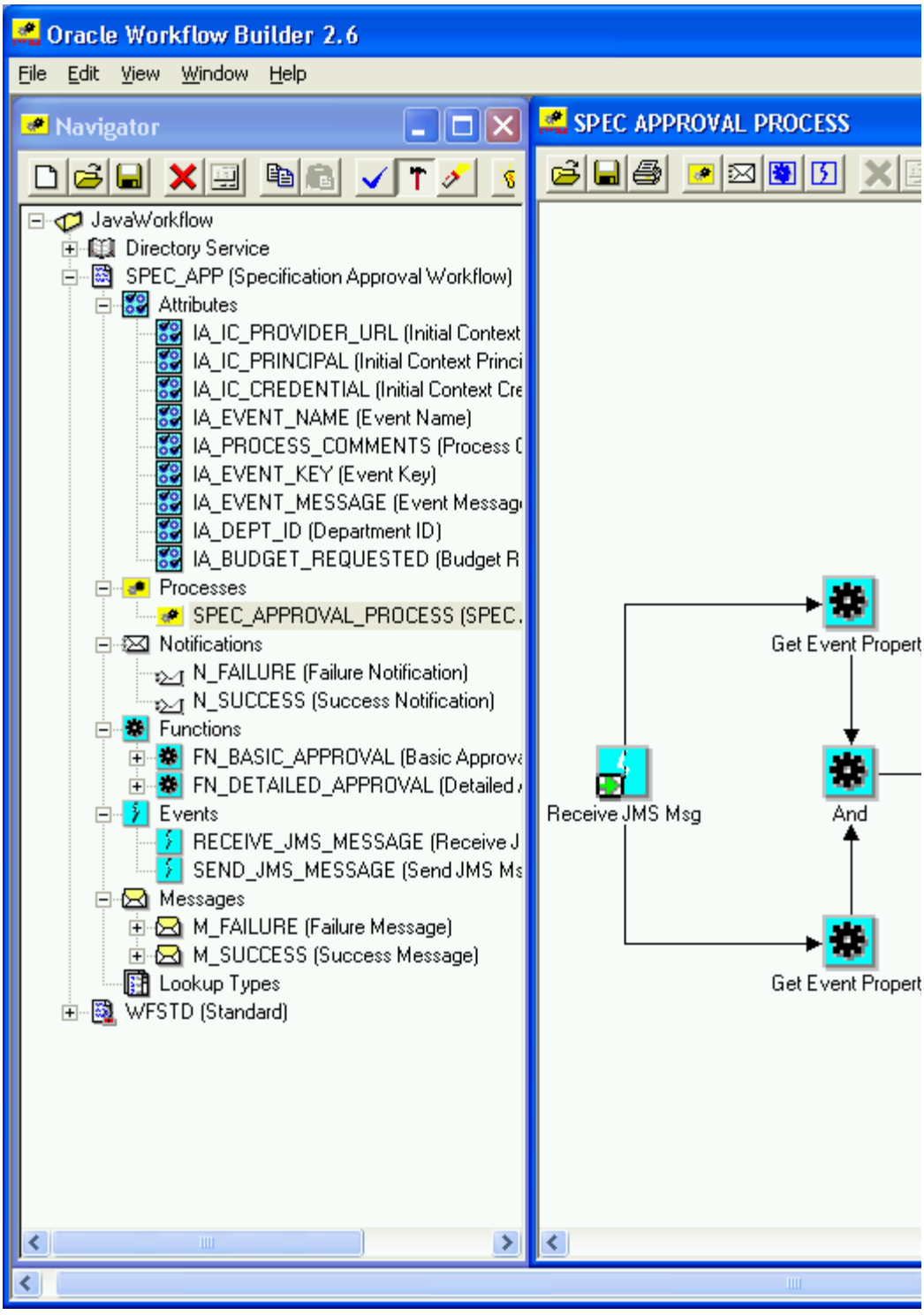
Sun's general JNI tutorial is here... it's brief by Sun standards
http://java.sun.com/docs/books/tutorial/native1.1/TOC.html#concepts

5) Workflow systems: What can they do, and how?

Using (surprise) Oracle's workflow system as an example, the user can define business processes by linking together process items and integrating the resulting workflow with core technologies like JMS and

EJBs. WFs can be instantiated just like any other object, and there are applet- and web-based tools to monitor each instance.

Here's what I don't get... what exactly is being linked together? The fact that something tactile is deployed leads me to believe that the workflow is composed of J2EE objects, but there is a distinct lack of J2EE vocabulary found in the Oracle Workflow documentation PDF. (http://otn.oracle.com/docs/products/ias/doc_library/90200doc_otn/integrate.902/a95265.pdf) Does the workflow definition take the results of Java/SQL processes and make decisions based on those outputs? e.g. if an SP call comes back with a particular code, do something, else do another thing? The web and Usenet have not been particularly helpful to me in this regard.



Oracle claims that its integration with PL/SQL as a major advantage, because it can use the results of stored procedures to make workflow decisions (that makes some sense, provided that the same result couldn't be inferred based on bean data using other platforms).

This white paper describes the idea in some detail... and mostly in English!
http://otn.oracle.com/products/ias/workflow/workflow_j2ee_wp.doc

A) APPENDIX: J2EE/9iAS Best Practices

http://otn.oracle.com/products/ias/pdf/best_practices/903iASBestPractices.pdf
I noticed this document during my Shen summit. It's about best practices in Oracle 9i App Server, but most of the ideas apply across platforms. Chapter 3 is particularly insightful regarding architecture components, including things like clustered J2EE apps, remote EJB communication, etc.