Torry Harris

Distributed e-enterprise Computing

NetDynamics to WebSphere Migration: An Overview

Authors: S. Lingegowda and S. Ramaswamy

Intended Audience: This paper is primarily intended for managers and architects considering options for migrating applications off the NetDynamics platform that is soon to be unsupported.

Further Inquiries and Feedback: info@thbs.com

Notices:

Torry Harris Business Solutions
1090 King Georges Post Rd
Suite 103
Edison NJ 08837
USA

1 Introduction

In this paper, we consider a problem facing many IT departments today – migrating applications off the soon to be defunct NetDynamics application server from Sun Microsystems to the popular WebSphere application server from IBM. The NetDynamics application server was a first generation server that was very popular and rapidly became the platform of choice for web applications. However, its event-based programming model did not become part of the now popular Java 2 Platform Enterprise Edition (J2EE) standards. As a result, the NetDynamics platform has become obsolete. The WebSphere application server on the other hand, is a highly robust and scalable offering from IBM that is based on the J2EE standards.

Torry Harris offers a high-level of expertise in the area of NetDynamics to WebSphere migrations. It has been working very closely with IBM on this problem and has performed some early migrations for large IBM customers. It has also co-developed a migration toolkit aimed at reducing the migration effort. Torry Harris has a large team that is highly trained and dedicated to doing NetDynamics to WebSphere migrations.

2 NetDynamics Applications

NetDynamics is a first generation application server built on a proprietary event-based programming model. NetDynamics was primarily designed for quick generation of HTML interfaces to Database tables. The focus of the product is on using wizards for generating Data objects with all the required SQL code in them and then using wizards to generate Page objects that allow web users to display/fill-in the contents of these Data objects (and consequently, the database records that they represent). In the latest version (ND 5.0), NetDynamics also allows for the creation and deployment of Enterprise Java Beans (SessionBeans only) – these are called BusinessBean objects. The logic in BusinessBeans can be accessed from other NetDynamics objects. Design and development of NetDynamics applications is done only through the NetDynamics Studio.

NetDynamics defines many classes of objects – the important ones are HTML Page objects, Java Page objects, Data objects, Data Source objects, BusinessBean objects, Session objects, Transaction Objects and Security objects. Each object is created using a wizard within NetDynamics Studio and has a set of events associated with it. These events are executed during pre-defined points of the corresponding object's life cycle. When an object is first created using a wizard, a default action is associated with the occurrence of each event. In order to change this default behavior for any of the events, the programmer must explicitly add Java code associated with the events. Note that this event Java code can be quite arbitrary – there do not seem to be any inherent restrictions on what can be added. Furthermore, this event Java code can access a large number of pre-built functions through a set of proprietary NetDynamics APIs.

In a typical project, NetDynamics developers create a number of Data objects to encapsulate the SQL statements needed and then create a number of HTML Page objects that collect user inputs to drive the Data objects as well as display the results of their execution. Default event behavior for Data objects is seldom over-ridden, on the other hand, default event behavior for Page objects is overridden for implementing presentation logic, data validation logic and business logic. Most NetDynamics applications also use User Session objects for tracking user level state. Use of other types of objects is fairly specialized and infrequent.

3 WebSphere Applications

WebSphere is a standards based application server. It implements the highly popular Java 2 Platform Enterprise Edition (J2EE) programming model. This model is based on the Model-View-Controller (MVC) design pattern for user driven applications. In the MVC pattern, a user driven application is considered to have a set of Views, which allow the user to interact with the application. A Controller handles user inputs collected from a View and passes them to the appropriate Model. Once the Model has processed these inputs and produced outputs, the Controller uses them to populate the next View for the user. In the J2EE programming model, JavaServer Pages (JSPs) or HTML pages serve as Views, Java Servlets or logic only JSPs serve as Controllers and Enterprise JavaBeans (EJBs) or Java classes serve as Models. WebSphere applications can be developed using the Visual Age for Java IDE, the WebSphere Studio IDE or any other Java development environment including a simple text editor.

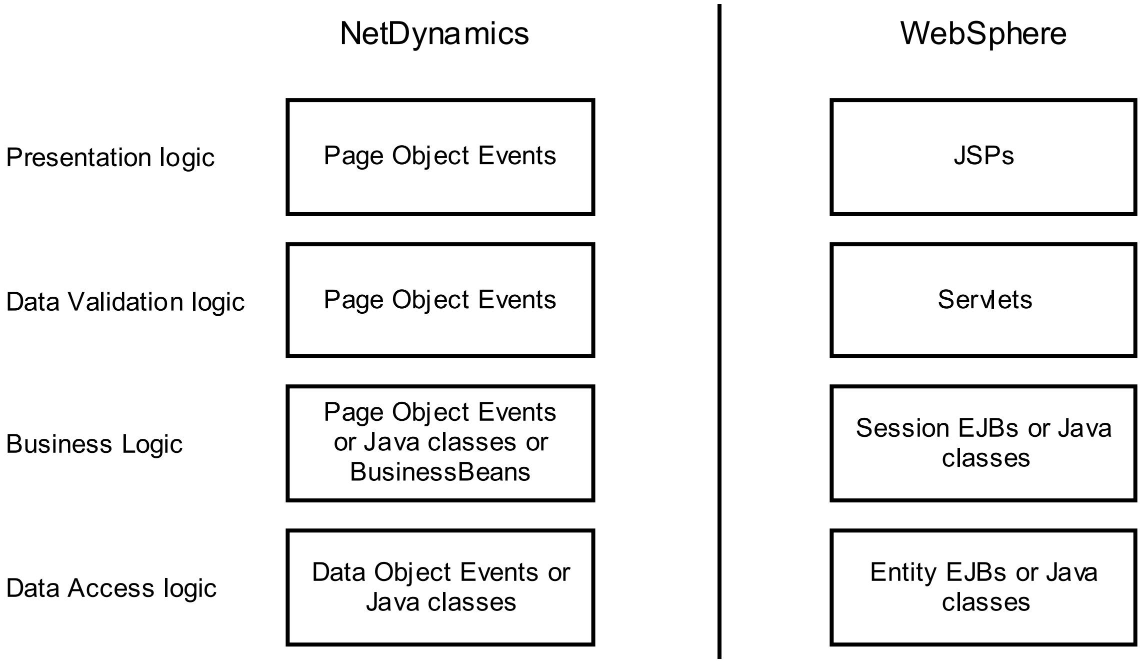
WebSphere applications consist of a set of Java classes conforming to the J2EE standards and using the allowed set of J2EE APIs. Applications have to be deployed into the WebSphere container using a set of tools provided with the product.

In a typical project, WebSphere developers create a set of Entity EJBs or Java classes to encapsulate their SQL statements and other data access logic, a set of Session EJBs or Java classes to encapsulate their business logic, a set of Servlets that contain data validation logic, and, a set of JSPs that contain presentation logic. WebSphere developers typically use popular J2EE APIs like the Java Database Connectivity (JDBC), the Java Messaging Service (JMS) and the Java Transaction API (JTA).

4 Migration Methodology

4.1 Application Logic

In terms of the application logic layers, the picture below illustrates the migration strategy for a typical NetDynamics application:



	NetDynamics	WebSphere
Presentation logic	Page Object Events	JSPs
Data Validation logic	Page Object Events	Servlets
Business Logic	Page Object Events or Java classes or BusinessBeans	Session EJBs or Java classes
Data Access logic	Data Object Events or Java classes	Entity EJBs or Java classes

4.2 NetDynamics Objects

In terms of key NetDynamics high-level objects, the typical migration looks as follows:

- HTML Page objects: these objects contain both visual elements as well as Java code that potentially implements the presentation, data validation and business logic layers. When migrating to WebSphere, the visual elements can be implemented in a JSP and the logic layers can be spread across a combination of Servlet, EJBs and Java classes. Servlets can be used for data validation logic and Session EJBs or Java classes can be used for business logic. WebSphere studio can be used for JSP creation and Visual Age for Java can be used for creating the other logic components.

- Data objects: these objects contain all the data access logic. Typically, static SQL constructs are used in data objects with parameters being filled in prior to their execution. However, in NetDynamics, developers can also create dynamic SQL statements by overriding the default event behavior of data objects. In WebSphere, the equivalent functionality can be implemented using either Data Access Beans (a special type of JavaBean), Entity EJBs or by Java classes. Data Access Beans and Entity EJBs can be created using WebSphere Studio/Visual Age for Java wizards similar to those in NetDynamics.

- Data Source objects: these objects represent connections to data sources. In WebSphere, the Java DataSource interface that is part of the JDBC standard can be used to provide similar functionality. In WebSphere, DataSource objects can be configured using the administrative utilities.

- Java Page objects: these objects are meant for creating pages that have active behavior at the client using the Java Applet specification. For WebSphere, a similar applet can be created using Visual Age for Java. In case the NetDynamics…

- BusinessBeans objects: these objects correspond to the Session EJBs as defined in an early version of the EJB standard. WebSphere supports a later version of the EJB standard and can run BusinessBeans with some minor changes.

- Session objects: these objects are meant to track state at the global, user and page levels. In NetDynamics, global state is visible to all applications within a cell. In WebSphere, tracking global state that is visible to all applications within a domain will require special code; however, tracking global state visible throughout an application can be implemented by using the Java ServletContext interface in JSPs or Servlets. In WebSphere, page level state can be tracked using local variables in JSPs or Servlets and user level state can be tracked using the Java HttpSession interface in JSPs or Servlets.

- Transaction objects: In NetDynamics, these are typically used to collect the execution of multiple data objects (and consequently data accesses) within a single transactional scope. In WebSphere, similar functionality can be implemented using the Java Transaction API (JTA) to create a transactional scope around the execution of a set of data accesses.

- Security objects: these objects are used to implement authentication and authorization policies for a NetDynamics application. WebSphere also provides for authentication and authorization checks for applications that can be set up administratively.

4.3 Back-end Systems

In addition to the set of objects mentioned above, NetDynamics provides Platform Adapter Components (PACs) that can be used for connecting to back-end systems – PACs are provided pre-built or can be custom built. In WebSphere, functionality equivalent to PACs will need to be provided either through custom coding or by using connectors available for popular back-end systems like CICS, MQSeries, SAP, etc.

Finally, NetDynamics also allows programmers to connect to back-end CORBA compliant servers and use their services within NetDynamics programs. WebSphere also permits such connectivity as it supports connection to external ORBs.

5 Migration Effort

Torry Harris along with IBM and one of its other business partners – Wavebend have built a toolkit that automatically migrates HTML Page objects and Data objects from NetDynamics to their equivalents in WebSphere. However, this automatic migration is restricted currently to objects that have been built solely using the NetDynamics wizards. In practice, this restriction limits the extent to which the toolkit can reduce the overall migration effort. The actual savings in effort is dependent on the extent to which wizards were used when building the NetDynamics application being migrated.

Most of the manual effort in any NetDynamics to WebSphere migration process will be in re-implementing the logic contained within the event Java code for the NetDynamics objects. This Java code can be arbitrarily complex and make extensive use of the proprietary NetDynamics APIs. While re-implementing the event code logic in WebSphere, the event execution order specified in NetDynamics must be kept in mind so as to create the correct resulting functionality. Also, basic JDK APIs will need to be used to replace the NetDynamics API references. In case a suitable replacement cannot be found, a special utility class may need to be constructed.

Manual effort will also be required in cases where PACs or calls to CORBA-compliant services are used. These cases will need to be examined on an individual basis and appropriate action will need to be taken. In some cases, especially with PACs, the effort may be very significant.

It is our opinion that most migration efforts will result in a sizeable fraction of the initial effort used to build the NetDynamics application. The general application architecture, page layouts and page flows are all reusable. However, all the event code that uses NetDynamics APIs has to be essentially rewritten. We believe that this code is probably what took most of the effort for the NetDynamics application. In cases where the NetDynamics application does not make extensive use of the proprietary APIs and implements all its logic in a well-defined set of Java classes with the event code merely invoking methods on these Java classes, the migration effort will be somewhat easier.

For the purposes of migrations to WebSphere, a classification of NetDynamics applications along the following lines is useful:

A. Applications built totally using wizards:
   - Re-architecture not necessary.
   - Automation is a big win.
   - Overall effort will be small.

B. Applications implementing only presentation logic:
   - Re-architecture not necessary.
   - Automation would help to a limited extent.

  - Overall effort may be sizeable – extent dependent on complexity of logic.

C. Applications implementing presentation and business logic:
  - Re-architecting necessary – extent dependent on initial architecture.
  - Automation of very little help.
  - Overall effort will be sizeable.

6 High-level Migration Plan

The general steps involved in any ND to WebSphere migration effort are as follows:

1. Detailed study of NetDynamics applications involved
2. Architecture of corresponding WebSphere applications
3. Definition of a detailed implementation plan of WebSphere applications along with testing requirements, performance requirements and deliverables.
4. Implementation of WebSphere applications
5. Testing of WebSphere applications
6. Deployment of WebSphere applications into production environment and performance tuning

The duration of each step will depend heavily on the particular application being migrated.

7 Summary

In this paper, we have outlined the details involved in performing a NetDynamics to WebSphere migration. As described, the migration effort is highly dependent on the way the NetDynamics application is architect and built. In some cases, the migration effort could be considerable and approach the original effort.

Torry Harris has all the necessary skills and experience required to carry out NetDynamics to WebSphere migrations. It has in depth knowledge of the programming models used by both platforms and has developed a process for migrating between the models. Furthermore, Torry Harris can reduce the cost of migrations significantly by using the migration toolkit co-developed with IBM as well as by using its offshore development center for carrying out the bulk of the migration effort.