iSeries Application Modernization – An Update

by David Slater;
Market Manager,
iSeries Application Development Products;
IBM Software Group

Application modernization is extremely important to IBM in the context of its e-business strategy. IBM believes that e-business is the future of the IT industry. Having a Web-enabled, browser interface is a requirement for many new iSeries solutions. Many existing iSeries solutions are good enterprise ready business application but they still have a 5250 green screen interface. In this paper, we will review some of the key modernization strategies in the iSeries marketplace.

A History Lesson

Before the introduction of ILE RPG, most 5250 applications were written as single programs to avoid the performance penalties associated with external calls. The user interface code, the business logic and the database access code were all included in monolithic large RPG/400 programs. This made application modernization a difficult process because the entire application had to be redesigned to make minor modifications to either the user interface or the database.

Application Modernization – An Architectural Approach

Separating all the user interface logic and the database access logic from the business logic using good modular design concepts is a basic first step in improving the ability for an application to be more easily maintained or modernized. This is also consistent with moving these applications more toward an object oriented footing.

By updating applications into a modular architected format, developers could more easily change the user interface or the database without making radical changes to the other components of the application. If a developer separates the user interface from the business logic and then moves the business logic into service programs, he has accomplished several objectives:

- The user interface can be updated independently of the business logic

- The business logic can be reused by other programs and applications

This architectural modernization strategy delivers significant e-business benefits. Now that the business logic is modularized into callable batch programs, WebSphere Studio for iSeries can be used to create a Web interface to this application logic. During the modularization process, the business logic is normally broken down into re-useable modules. This significantly improves the maintainability of the business logic and makes this logic available to both traditional and Web applications. When business logic is accessed via a Web interface created by WebSphere Studio, the cost penalty associated with interactive application execution is also avoided.

There is a degree of isolation of the user interface from the application logic that is inherently part of the iSeries system. The display file DDS describes the format of the 5250 interface and this display file DDS is separate from the business logic of the application. IBM has leveraged/used the Display file DDS to create the Web interface using the WebFacing Tool.

WebFacing – A Shortcut to the Web

Although most architects would agree that we should modularize our applications and separate the UI and Database code from the business logic, the world of e-business doesn't wait. The "need for speed" is very important in the Web world. If you can't submit a bid to satisfy a customer requirement because you don't have a browser interface for your application, you need a quick fix.

The WebFacing Tool creates a Web interface to an existing host application quickly, easily and cost effectively. The WebFacing Tool creates the Web interface based on the display file DDS, using standard Java components, JavaServer Pages, Java beans and servlets; there are no proprietary interfaces. These Java components can be customized and enhanced with any Java development tools such as VisualAge for Java, WebSphere Studio or the WebFacing Tool. The same application logic can drive both the 5250 and Web interface – no dual maintenance. The Java components will run in any application server that supports JSPs, Java beans and servlets, like WebSphere Application Server, Standard Edition or the open source Apache HTTP server with the Tomcat plug-in.

Using the WebFacing Tool is a modernization tactic that appeals to many application developers. The WebFacing Tool is a component of WebSphere

Development Studio for iSeries. There is no separate development tool cost. The Java components created by the WebFacing Tool will run in WebSphere Application Server, Standard Edition, a no-charge feature of OS/400. The conversion process is quick and simple. A developer can convert up to 500 5250 screens in a couple of hours and publish them to a WAS unit test environment for review and test.

This is the first release of the WebFacing Tool. Not all of the DDS keywords are supported. The development team has converted about 60% of the DDS keywords representing approximately 90% of the DDS keyword usage. The development team is continuing to expand the number of DDS keywords that are supported. So iSeries developers have a choice. They can change / revised their applications to use only the supported keywords or they can delay their conversion until the DDS keywords critical to their applications are supported. Please note that some DDS keywords will never be supported because either the usage of the keywords is extremely low or the DDS keywords make no sense on a browser interface.

We have several goals for the WebFacing Tool:

- Increase the DDS keyword support so that most applications can be converted easily with no changes to the DDS.
- Make it easy to enhance and extend WebFaced applications using WebSphere Studio for iSeries
- Convert the WebFacing runtime support so that WebFaced applications run in batch mode and avoid the cost/performance penalties associated with interactive execution.
- Update the generated Web interface to be able to support wireless devices from WebFaced applications.

How about Windows-based event-driven GUIs

Some developers like the rich Windows-based event-driven GUIs that were popularized by client/server applications. This GUI is not quite as popular as it was in the past because it features a fat client interface. The fat client has several problems in the Web world. For client/server applications, there are the system management difficulties in trying to manage the code on the clients. When there are only a few client workstations, keeping the clients are the same version and release level of the application is manageable. As the number of clients and the

number of supported client operating systems expands, the system management issues become daunting. The system management issues can be addressed by providing the Windows GUI as an applet that is downloaded from the server into the browser interface. This means that there is only one version of the client interface to maintain. However, this solution forces the user to download a fat client GUI to execute the application. This can result in unacceptable performance if the user is operating with a standard modem connection to the server. This option should not be dismissed however. If the developer minimizes the size of the applets (i.e. puts most of the logic on the server), and the user is connected over a high-speed cable modem connection, this e-business option offers a rich GUI option for users in a B2B world.

VisualAge RPG allows users to create rich event-driven Windows-based GUI applications using the RPG IV language that is used for ILE RPG. Because the same language is used on both the client and the server, RPG developers cut and paste business logic between the client and the server. This makes it easy for VA RPG developers to minimize the logic on the user interface. There are several capabilities in VisualAge RPG that increases its usefulness for e-business development.

- You can generate Java applications or Java applets from VisualAge RPG source.
- You can create batch RPG applications with no user interface with VisualAge RPG.

By generating Java applications, the user interface can be executed on any client that supports a JVM ...and every strategic client has JVM support.

If developers update and modularize their RPG applications, the business logic is put into service programs. These service programs can be moved into VisualAge RPG and then Java applications can be generated from the RPG source. This would allow you to convert RPG application logic to Java and run it on any server with a Java Virtual Machine. This could drastically increase the number of deployment platforms available for RPG solutions.

What about VisualAge for Java

If e-business is the future of the IT industry and Java is the language of e-business, why isn't VisualAge for Java the premier legacy modernization tool?

VisualAge for Java is the tool of choice for writing new, portable, object oriented e-business applications. You can use VisualAge for Java to move legacy solutions to an e-business footing in two ways.

1. *You can choose to rewrite your solution in Java*. This may be an option for some critical applications that need cross-platform execution. However, this is very expensive from both time and resources and is not a viable business proposition in many cases. It usually violates the Web imperative "The need for speed!"

2. *You can put a Java interface to your existing applications*. This was a very popular option in the initial days of e-business. To use this approach, you must first re-architect your application to separate the UI logic from the business and database logic. This is typically not a simple first step. Also, these Java front ends are typically fat. The Java interface either must be served up as the client in a client/server solution or downloaded as an applet. The concerns with these types of e-business implementations have already been covered in the section titled "How about Windows-based, event-driven GUIs?" If you have separated you UI logic from the rest of your application, you can access your existing business logic using WebSphere Studio (as outlined in the section Application Modernization – An Architectural Approach) The learning curve associated with this product is smaller that of VisualAge for Java.

Although VisualAge for Java fulfills an important e-business development role, it is not a key tool for legacy application modernization.

Web Services in the iSeries market

Web services aupports is a key capability that is being delivered along with the new Eclipse Integrated Development Environment in the next release of WebSphere Development Studio for iSeries. This new support will allow developers to easily consume and construct Web services. Developers can browse the UDDI registry, a "yellow pages" of Web services, to locate existing, published Web services. They will be able to generate a JavaBean Proxy to utilize these Web services in their application solutions.

Developers will also be able to create Web services using existing applications and data. Wizards will be provided to wrapper existing iSeries programs as Java beans. Developers will then be able to create new Web services using these JavaBeans or using existing databases. Developers will be able to deploy these Web services to WebSphere Application Server or Tomcat for testing with the built-in test client. Developers can then publish these newly created Web services to the UDDI registry.

Web services offers a new modernization strategy for legacy applications. Web services has the potential to rapidly expand the development and deployment of new e-business applications and will allow developers to assemble new solutions from available (from UDDI) Web services. The base of these Web services will be a combination of existing legacy applications and new application components.

This is a dramatic enhancement to the concept of creating/assembling solutions from components - a new component assembly model. The Web services model may revolutionize the way companies integrate their IT solutions in the business to business marketplace.