Welcome to WebFacing

IBM gives a walk-through of the new WebFacing architecture.

By George Farr and Phil Coulthard

In this article we will explore the role and strategy of WebFacing, the newest IBM application development tool for iSeries™ programmers. A follow-up article will offer a more technical hands-on tutorial introduction to its actual use.

WebFacing and The AD Strategy

As of the V5R1 GA on May 25th, the world of application development for the iSeries platform changed forever. If you missed it, there is now only one product, WebSphere Development Studio (WDS), for application development (both for V5R1 and V4R5). It contains all four priced compilers (RPG, COBOL, C, and C++), all green-screen tools (ADTS), and unlimited licenses to a package of client tools known as WebSphere® Development Tools for iSeries (WDT). For software subscription customers, this new product is a free upgrade from any of the constituent products. For the rest, or for new boxes, it is a delta to the historical price of ADTS plus RPG. Either way, the key is this: You cannot *not* get it. It is the new base for all application development work on iSeries —and if you are not yet at V4R5 or V5R1 … well, it's time to get on with it!

Even though the compiler changes in V5R1 are extensive and exciting, it is the client tools that are catching people's interest. These tools are a packaging of the key Windows-based tools for iSeries development for everything from traditional green-screen to modern e-business applications. The tools include:

- ***CODE/400*** – The modern-day follow-on to SEU, SDA, RLU, PDM, and the system debugger.
- ***VisualAge® RPG*** – For writing Windows client/server GUI applications easily with RPG.
- ***VisualAge for Java™, Professional Edition, plus Enterprise Toolkit for iSeries*** – This tool is for writing all manner of Java code, except for Enterprise JavaBeans, which require an upgrade to the Enterprise Edition of VisualAge for Java.
- ***WebSphere Studio, Professional Edition, plus iSeries Extensions*** – This tool is needed for authoring new Web browser user interfaces that run inside WebSphere Application Server.
- ***WebFacing*** – This tool is the new kid on the block and the focus of this article.

With this set of products, you now have everything you need to use modern tools to write modern applications with a modern user interface. Whether you ease into them or dive headlong is your call!

IBM's essential strategy is to modernize your application development tools, your applications, and most importantly, your application user interfaces. IBM wants more Web browser-based applications running in WebSphere Application Server. These tools are part of that strategy.

To create a Web browser user interface, we give you WebSphere Studio with some exciting iSeries extensions for easily generating all the WebSphere plumbing between that user interface and your batch RPG or COBOL business logic (actually application programming interfaces). This is a fantastic option if your green-screen logic is divorced from your business logic and your business logic is easily accessible using discreet APIs. Many of you are not there, but we encourage you to consider this your long-term goal.

In the meantime, if you have a large interactive application that is not easily re-architected with a formal divorce between the display file I/O and the business logic, we offer WebFacing. This is what we call the "one-two punch." IBM gives you great tools for new applications, and another tool for a "quick fix" to help you deal with existing applications. The idea is that this will buy you time with your customers, or users, or executives, while work begins on the next generation of your application. What if you like WebFacing enough to just stay with it? Well, that is your call too!

What is WebFacing?

We are often asked how WebFacing compares to 5250 intercept products that convert a 5250 datastream to something else on the fly. Many assume that this is what WebFacing is. It is not. There is no 5250 datastream translation involved with a WebFaced application. There is no 5250 datastream, in fact.

WebFacing is a different approach to the problem of re-facing an existing application. WebFacing is a developer's tool. It is a combination of a development-time display file source conversion and a run-time intercept deep in the operating

George Farr ***Phil Coulthard***

system. You convert your user interface source, and with the run-time intercept your existing application code runs as is, still thinking that it is driving the original display files, when in fact it is driving the Web user interface created by the conversion tool at development time. Before dissecting the benefits (and disadvantages) of this approach, we'll dig a little deeper into the process.

First, let's review briefly what happens today when you interact with a display file from a high-level program. The write operations send a buffer of data to the workstation data manager, which merges that data with the record format in the display file to produce a 5250 data stream. The read operations cause the workstation data manager to return to your program all the user input data in the form of an input buffer, and event information in the form of a feedback buffer. The read operation causes your code to block until the user presses Enter or a function key. A noteworthy feature is that the program starts the user-interface interaction by writing that first screen. In a sense, the program drives the user interface.

In a WebSphere application, you have a different architecture, but with some similarities. First, the user starts the interaction by calling up a Web page, so it is the user interface that drives the program, not the other way around. A Web page is defined as a JavaServer Page (JSP) file, which is simply an HTML file with embedded JSP tags. The HTML tags define the constant part of the page, while at run time the JSP tags are used to extract the run-time data out of a Java bean, on an output operation. Note that this is a simple Java bean, *not* an Enterprise Java Bean. A JSP is called from a servlet, which is a small Java program that manages the flow of the JSPs. The servlet will put the data into the Java bean and pass it to the JSP. Your user typically calls your servlet first, which then calls the initial JSP, resulting in the first page.

All input fields in a JSP are captured in standard HTML forms, which contain various input widgets such as entry fields and radio buttons. Each form is associated with a server side program such as a servlet that is called when the user presses the Submit button. Each form has one →

Submit button, which is therefore very similar to the Enter key. In addition, a form can have other buttons that can be coded to call the servlet, similar to the function keys in a display file. When a servlet is called from a form, it reads the input entered into the form by the user, and uses it to populate a Java bean. It then calls business logic to do the requested task, passing in the user data. The business logic returns new data, which is put into that same bean. The servlet then decides which JSP to show next and calls it, passing in the Java bean for the run-time data. This is the basic flow of a simple Web application, and this is what WebFacing uses.

Now we can discuss how WebFacing maps the old world into the new world. The WebFacing Tool is a development environment (IDE) for creating WebFacing projects. This IDE contains a wizard for identifying the input information, such as the display file DDS source members, and a project action for converting or reconverting that DDS source into a Web user interface.

The conversion tool generates the following for each record format:

- ***Two JavaServer Pages*** – one containing the user interface format, and one containing purely JavaScript routines for applying conditioning, and so on. JavaScript is a language that is interpreted and run by the Web browser itself.
- ***Three JavaBeans*** – one for the input and output fields, one for the feedback information, and one for the view information such as indicator settings.
- ***One Cascading Style Sheet*** – for all the files converted in this project. By changing this style sheet, you can affect all the JSPs in your application.
- ***A Single index.html File*** – to act as the front door to your application. Users typically start with it, and it has a link that starts your application by calling the WebFacing servlet.

You then use the tool, which runs on Windows, to publish these output files to WebSphere Application Server along with the WebFacing servlet. This can be WebSphere on iSeries, or indeed, any WebSphere Application Server. For example, you many choose to test it on WebSphere Application Server on Windows first, or the WebSphere Application Server that comes with VisualAge for Java (for testing and debugging.)

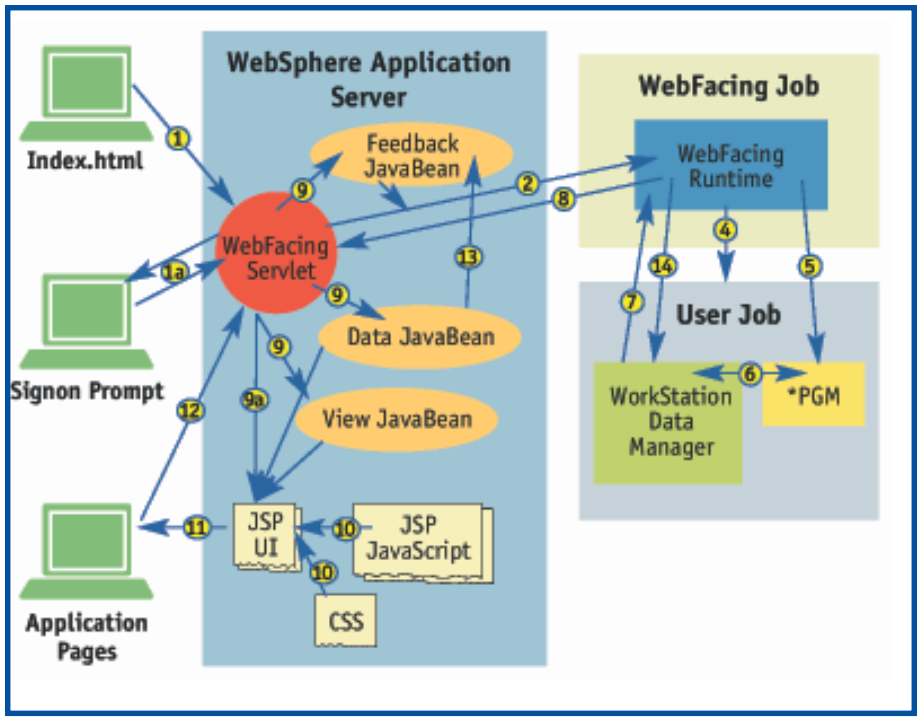

Figure 1: The WebFacing Tool allows Web-browsing users to access legacy applications through a combination of JSP and servlets.

The WebFacing run time is a piece of code running on the iSeries. It is the heart of the magic. The servlet communicates with it through Sockets calls, and the runtime, in turn, communicates with the WorkStation Data Manager. When you run your WebFaced application, here is what happens:

1. The user presses the generated link to start your application. (If the user has not signed on yet, a sign-on prompt will be displayed.)
2. The WebFacing Servlet gets control and it calls the WebFacing run time.
3. The WebFacing run time spawns a thread for this user, and then starts an AS/400 job. It either users a hard-coded user ID and password, or tells the WebFacing servlet to prompt the user for these. (This is a decision you made at conversion time.
4. The run time tells WorkStation Data Manager that you are in WebFaced mode.
5. The run time calls your application (using the CL command and parameters you specified at conversion time), which starts running.
6. Your application does a number of write operations to the display file, followed by a read operation.
7. The WorkStation Data Manager, knowing that you are in WebFacing mode, simply passes the application data directly on to the WebFacing run time, which in turn passes it on to the WebFacing servlet, which is waiting on a socket. The WebFacing servlet puts the application data into the generated Java bean for this record format and passes it to the generated JavaServer Page for this record format. The JSPs for all the writes are merged and finally sent as a unit on the read operation. The user sees this as his next page. At this point the RPG program is still blocked, waiting for the read operation to be completed.

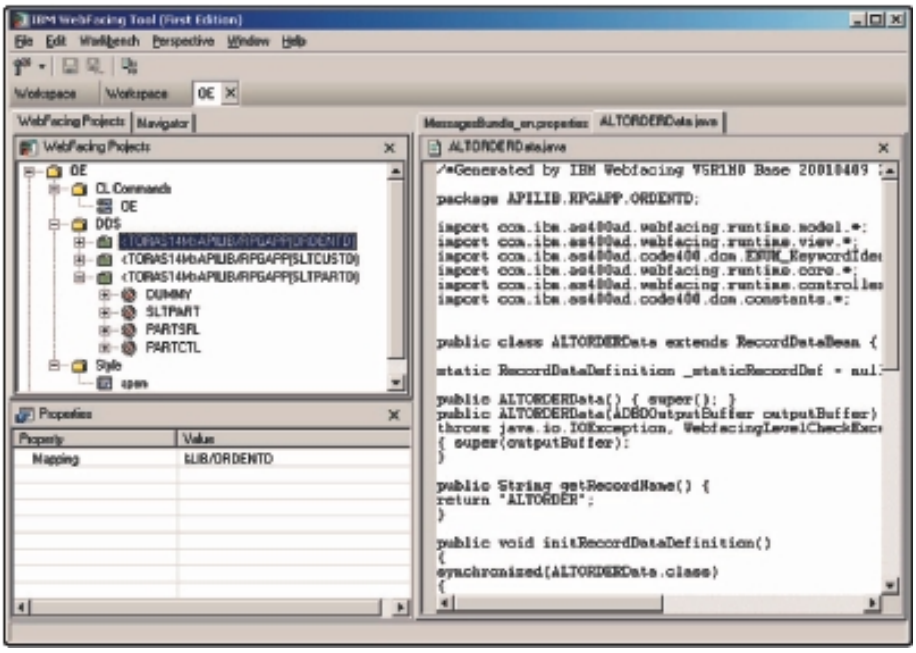

Figure 2: A WebFacing Tool wizard lets you customize the look and feel of your DDS screen.

8. The user enters data and presses Submit or a push button representing a function key.
9. The WebFacing servlet gets control again, reads the user input data, and passes it and the feedback information to the WebFacing run time. The run time passes the data and feedback information to the WorkStation Data Manager, which uses them to respond to the RPG program as usual.

This continues until your application exits. The WebFacing servlet recognizes the exit and cancels the WebSphere session. Of course, there are all those little details that have to be dealt with, like library lists, break messages, user timeouts, the browser Back button, and so on. Not to worry, we have done the worrying for you. Including the biggest worry of all: subfiles!

Another wizard is also part of the tool. This wizard helps you create your own styles, as we call them: the look and feel that all your Web pages will have. Basically, we put the converted screen in the center of the page, but you are free to put what you want in each edge, making it easy to add corporate logos, for example. Use the style to customize the colors and fonts of the application area, and position the function key buttons. The wizard makes it reasonably easy to generate new styles, but to get you going, the tool comes with a number of predefined styles to choose from. Enterprising business partners might want to consider marketing additional styles as well. That is WebFacing, at a glance.

Why Use WebFacing?

The advantages to this approach over the traditional approach of converting the 5250 datastream on the fly are twofold. First, the conversion is done at development time so should you decide you don't like the output we generate, you are free to change it. The JSPs and beans are industry standards, and there are several tools for editing them, including WebSphere Studio, which ships with WDT, (Websphere Development Tools.). The key is that you make all your changes at development time on source code, as opposed to writing run-time macros. The second advantage is that there is no 5250 datastream created simply for the purposes of subsequently translating it to

something else again at run time. This provides an opportunity for better performance.

Regarding the editing of the output – it sounds great, but there is a caution. Once you change what WebFacing generates, it will be your responsibility to merge your changes back in after a subsequent regeneration (WebFacing backs up your changed file first.) Future releases will offer merge tools, but in the meantime you are on your own. To minimize the need for this, we offer the ability to tailor the output prior to the conversion. The CODE Designer tool is the SDA follow-on inside CODE/400. If you use it for your display file DDS source, you will find that there is a properties dialog for fields where you can specify properties that only pertain to the conversion code. For example, you can elect to hide a file from the generated Web page even though it will still show up in the green screen.

You can also insert any HTML you like into your green screen, such as a little image beside each menu item. Again, this does not affect the green screen, as WebFacing stores these properties as DDS comments. You can even have a field be rendered as an image, where the name of the image file is deduced from the value of the field at run time. For many, this capability, along with the style support, will be sufficient to get a good-looking Web page that the boss will like. However, what is key is that if it is not sufficient, you have all the generated code at your disposal and you are free to do what you want with it.

"It's okay. One of the routers must have gone down and we had a brief broadcast storm."

At the ultimate extreme, you may even decide to alter your RPG code to optimize the screen flow for the Web. To help you do this, we offer an API to query whether you running on the Web or as a green screen. Doing this type of surgery today is a bit tricky, though. We will make it much easier in future releases.

As a final advantage for WebFacing, it is cost effective. The tool is part of WDT that eventually everyone will have. The AS/400 run time is an OS/400 PTF.

Why Not WebFacing?

To use WebFacing, you must WebFace the entire application. You cannot switch between Web and green screen randomly. Not all screens can yet be converted. For example, we don't handle UIM yet, and hence system screens (we will be doing UIM, with UIM help panels the first priority). In fact, the GA product for V5R1 does not handle all DDS keywords yet. We prioritized them based on scanning the source of 120 business partner applications, so we hope that we have coverage for what you use (the conversion report will tell you). If not, our Web site will offer frequent updates to the tool and the conversion throughout this release cycle. Watch it carefully. At this writing the URL was subject to change, but this one will definitely get you there by way of links: www.ibm.com/software/ad/wdt400.

WebFaced applications only run on V4R5 or higher of OS/400, and they require both HTTP Server and WebSphere Application Server to be installed, configured, and running.

Something for Everyone

WebFacing is a new option for you to convert your green screen applications into Web-enabled applications, quickly, easily, and cheaply, and with a number of options for refining the resulting output. We think that for many it will be just perfect, and in fact that is what we are hearing. For others, it will take a little more evolution before it fits their bill. Still others will choose to simply install and turn on a 5250 intercept product so as to easily Web-enable every screen of every application (assuming they like the output). Yet others are digging in and aggressively rewriting and redesigning their applications as native Web applications. This is all good. The only wrong thing to do is nothing!

As a side note, we should point out that WebFaced applications still count as interactive applications and therefore still require the interactive feature. In a follow-up article we will go through the tools and show you a little more of the mechanics of actually doing a WebFacing conversion. TUG

George Farr *and* ***Phil Coulthard*** *are coauthors of the Midrange Computing books Java for RPG Programmers and Java for S/390 and AS/400 COBOL Programmers. George can be reached at farr@ca.ibm.com, and Phil can be reached at coulthar@ca.ibm.com.*

This article first appeared in the June-2001 issue of Midrange Computing magazine. (Reprinted here with permission.)