More Web Tools for iSeries Programmers

By Phil Coulthard and George Farr

Here we continue to introduce iSeries programmers to the Web tools that come with WebSphere Development Studio Client (WDSc), which all iSeries programmers have. Previously we described the model-view-controller (MVC) architecture of modern applications, and showed how the Struts framework facilitates architecture to build Web applications. We also started to describe the Web tools that make it easy to build a new Web application on top of Struts, using RPG or Cobol for the three-tier (model) business logic. Now, we get to the fun part where we actually build a running piece of a Web application, leveraging existing RPG code and our existing RPG skills.

The iSeries Web Tools

There are three layers of Web tools within WDSc. The first layer is the Web tools inherited from the base product WebSphere Studio Site Developer, which includes a Web project type and perspective, along with various editors for common Web file types such as HTML, GIF, Java ServerPages (JSPs), and Cascading Style Sheets (CSS). Site Developer also provides the WebSphere Test Environment for testing and debugging your Web application with a local copy of WebSphere Application Server. This first layer of Web tools makes it relatively easy to build Web applications using Java as the business logic language.

The second layer of Web tools builds on the first and is also inherited from Site Developer: the Struts tools. These tools simplify building a Web application that leverages the popular open-source Struts framework, but again uses Java to build the business logic language. This layer includes the Web Diagram Editor, which lets you lay out your Web flow graphically, prior to creating the actual files (see "Better Architecture with MVC and Struts," September 2005, *TUG eServer magazine*).

The third layer builds on the first two layers and is unique to WDSc. This layer adds support to enable the business logic of a Struts Web application to be RPG or Cobol, rather than only Java. We call this layer the *iSeries Web tools*, and our goal is to make it possible for RPG and Cobol programmers to build functioning Web applications with their existing skills.

The iSeries Web tools include two wizards and a palette of Web page parts. The first wizard is the iSeries Web Tools Run-time Configuration wizard, which lets you configure your Web project for a targeted iSeries system containing your RPG or Cobol business logic. You can launch this wizard from the context menu of a Web project or from the toolbar when a Web project is selected. In this wizard, you specify information that applies to all *interactions* in the project. An interaction is a two-Web-page sequence, where a Submit button on the first page invokes business logic (using a Struts action) and then displays the second page. Essentially, dynamic Web sites or applications can be considered collections of interactions, where the output page of one interaction may be the input page of another interaction.

The second wizard in the iSeries Web tools is the iSeries Web Interaction wizard, where you can create a new interaction easily from existing RPG or Cobol business logic. You identify the program or ILE procedure to invoke from the first page, including the input and output parameters. The wizard then generates a Struts action class in Java to call that program or procedure. If you want, it can also create an input Web page that prompts for the input parameters and an output Web page that displays the output parameters. This is a fully functioning interaction; when you click Submit on the first page, your business logic is invoked, and the results appear on the second page. If you prefer, you can specify existing input and output pages. In this case, you map the named fields on each page to input or output parameters in the program or procedure call, and the generated code binds them.

To help you create your own Web pages, WDSc includes the Page Designer—a WYSIWYG tool for Web page design. The Page Designer has an iSeries palette full of Web UI controls that you can drag and drop on your page. These controls have attributes that will be very familiar to you, such as edit code and edit word support for labels and entry fields, and validity checking options. These attributes generate the appropriate JavaScript, so you don't have to become a JavaScript expert just to prevent a user from entering a letter in a numeric field.

A Quick Tutorial

These two wizards and the palette of UI parts constitute the primary iSeries extensions to the Web and Struts tools in WDSc. We will walk through a little tutorial to show you them in action. To follow along, you will need to get the iSeries library containing the RPG business logic we will use. Go to ibm.com/software/awdtools/wds400, click the "Library" link on the left, then click "iSeries Web Application Development," and then click "Tutorials." Under "Building an e-business application with RPG," select the SAVF file, version 5.1 link to download the wsslabxx.savf file to your workstation. If you can't find the link, search the IBM Web site for this file name.

Once you have the file locally, start WDSc and use the Remote System Explorer perspective (the default) to create a connection to your iSeries. Then browse for the .savf file using the Local connection.

When you find it, right-click it and select the Restore on iSeries action, taking all the defaults except changing the "Restore to library" field at the bottom to be "WSSLABXX." This creates a WSSLABXX library on your iSeries system for you. If you're not running V5.1.2, you must follow the instructions on the Web site to restore this save file.

You should now follow the instructions from "Web Tools for iSeries Programmers – The Details" (November 2005, *TUG eServer magazine*), which will leave you with a new dynamic Web project that has been configured. Or just use File|New|Other to create a new Web project of type Dynamic Web Project, with whatever name you prefer. On the Features page, specify the Struts and iSeries Web Components features. The project contains (or will contain, if you create it now) a Web diagram depicting a single interaction: one input page (input.jsp), one action (InquiryProcedure), and one output page (results.jsp). The diagram will also include connections from the input page to the action, and from the action to the output page. For your convenience, in V5.1.2, an empty Web diagram is now started for you when you create a Struts-enabled Web project, so you can quickly start designing your flow by dragging and dropping from the palette.

Now that we have designed our first flow, we can "realize" it, but first we need to configure the project. Right-click the Web project name and select "Specify iSeries Web Tools Run-time Configuration." In the resulting wizard, add WSSLABXX to your library list. Because there is more function in V5.1.2, this wizard spans two pages, and library list information is now on the second page. It is also important to specify your iSeries host name, user ID, and password on the first page.

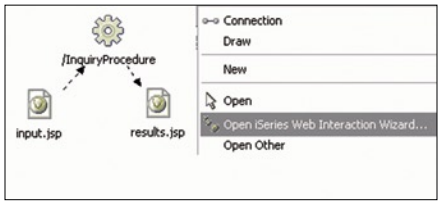

Figure 1: Launching the Web Interaction wizard

Returning to the Web Diagram Editor, it is now time to "realize" our little Web diagram. Right-click the InquiryProcedure action, and select the Open iSeries Web Interaction Wizard menu item (**Figure 1**). We want to generate the action class to call some RPG logic, and generate those input and output pages.

Click Next on the first two pages to take the defaults. On the page called "Specify Input and Output Parameters for your iSeries Host Program," specify INQUIRY for the program alias, GETDATA for the program object, and WSSLABXX for the library, and then click OK. This information identifies the program to call when the Submit button is pressed on the input page. Notice that we could instead call a procedure in a service program by specifying the service program and entry point or procedure.

This RPG program accepts one input parameter (customer ID) and updates two output parameters: an externally described data structure and a feedback field containing error information. Now we need to specify these parameters.

The first one is a seven-character customer ID that is input to the program, so enter CUSTNO for the parameter name, leave the data type as character, enter 7 for the length, and select input for the usage. Then click OK.

The second parameter is an externally described record that is output from the program. Right-click in the tree on the left, and select Add Database Reference Structure. Expand your previously created iSeries connection, or if none exists, create one first (expand New Connection and fill in the information), and then expand it. Expand "Work with libraries..." and enter WSSLAB* for the library name. Click OK, and in the resulting list, expand library WSSLABXX and logical file CUSTOML3, and select the record format CUSTOM01 from the list (**Figure 2**). Click Add, and then Close.

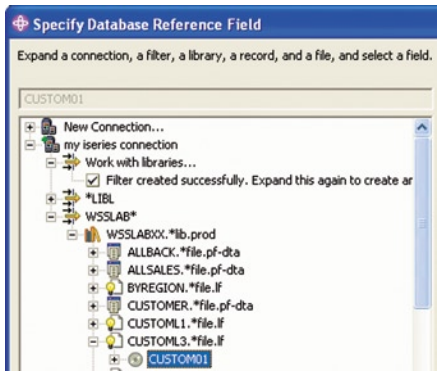

Figure 2: Selecting a database format

You have added the structure to the tree, but now you need to define the parameter that uses the structure. Select INQUIRY in the tree on the left, right-click, and select Add Parameter. Enter CUSTDATA for the parameter name, select structure for the data type and CUSTOM01 for the structure name, select output for the usage, and then click OK.

To define the final parameter, enter FEEDBACK for the parameter name, leave the data type as character, specify 20 for the length and output for the usage, and click OK. Your definition should now appear as in **Figure 3**.

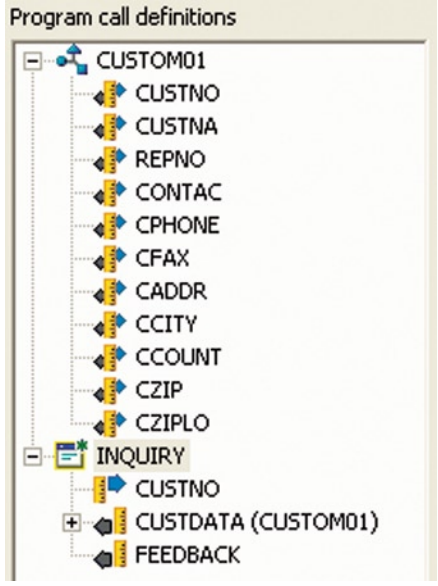

Figure 3: Program parameters

Next Article in this RPG to J2EE Series:

March 2006 – "Web Services Tools for iSeries Programmers"

Click Next to see the wizard page where you can affect the generated input Web page. Select the CUSTNO input parameter in the upper left panel, and in the lower left panel, change its Label property to Customer number, being sure to press Enter after typing. Select the Page tab at the bottom left, and then change the Page title property to Customer Inquiry.

Click Next to see the wizard page that lets you affect the generated output Web page. Change the Page title property to Customer Details. Note that the default is to generate each output field as a label. But you can change that here if you want to allow the input fields to be editable and thereby allow this page to become the input page to another interaction. Select the Fields tab at the bottom left, and then select the FEEDBACK field in the upper left. Deselect that field's check box, so it does not appear in the output page. Click Finish. Your icons now have color in your Web diagram, and the lines are solid. This tells us that they are now *realized*—that they exist physically.

Before we actually run our new little Web application, we need to check something. If you have named your project something with embedded blanks, such as "My Web Project," then you must right-click the project name and select Properties, and then select Web, and change the context root to something without blanks, such as "MyWebProject." Press OK and answer Yes to the message. WebSphere does not like blanks in its path names, and blanks are hard for users to enter as URLs.

Now you want to see your first Web interaction running, so in the Project Navigator right-click the input.jsp file within the WebContent folder of your Web project, select Run on Server, and click Finish to take the defaults (but ensure "Create a new server" is selected). This starts the built-in copy of WebSphere Application Server, which runs your Web application locally, where it calls your RPG program remotely.

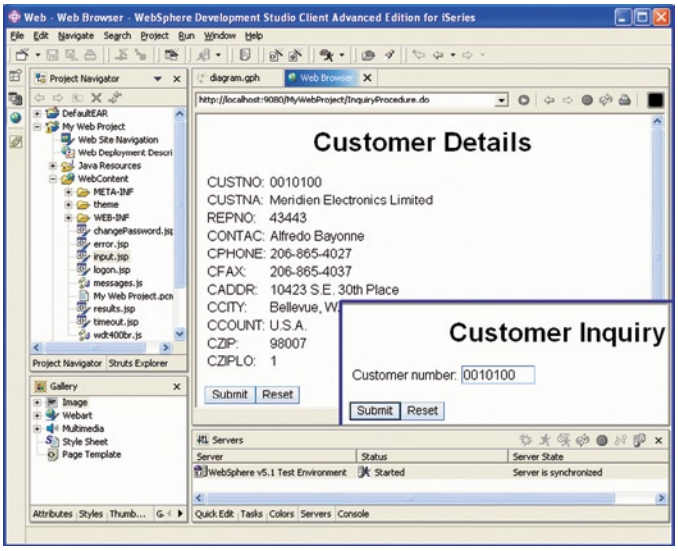

Figure 4: Result of running an interaction

When your first page shows up in the Web browser, you are good to go! Enter 0010100 for the customer number, click Submit, and you should see the results in the output page (**Figure 4**).

Note the initial run is slower as it page-compiles the JSP files on first touch. Also, there is the overhead of signing on and starting the job, which is only done on the first interaction of your Web application.

The idea behind the Web Interaction wizard is to prevent the need to learn great details about Web programming to build functional Web applications with RPG and Cobol. To this end, there is quite a bit of power in this wizard, especially in V5.1.2. For example, on page 2 of the wizard (and subsequent pages), you can now choose to generate the Web pages following an IBM-supplied or user-defined page template, so that all your pages have the same professional look and feel. You can also specify multiple output pages, with multiple connection lines coming out of the action, before running the wizard. Do this when you want to generate multiple output pages, but only show one at runtime, depending on some condition. The assumption is that condition is based on the value of one of the output parameters of your program, which we call a *flow controller* parameter. In this case, an additional wizard page prompts for the flow controller parameter and allows you to map values to output pages. All the conditioning logic is generated for you.

Another common requirement is to do error checking of the input parameters in your RPG program, and somehow bubble the error back to the Web page. This is quite doable by designating one of your output parameters as a message flow parameter, using a parameter property in the property sheet in the Result Form wizard page. You map parameter values to error messages in the application message file or resource bundle, specified in the runtime configuration wizard. If the parameter has any of these values at runtime, the input page is shown again with the mapped message.

You can also add function with this wizard to Web pages that already exist. For example, in the Web Diagram Editor, you can drag and drop existing pages to the diagram and connect them to actions as you do to Web pages that are not yet realized. If you do this, then the wizard detects that the pages exist and lists all the input fields in the input page so you can map each one to an input parameter. Then, it lists all the output fields in each output page so you can map each one to an output parameter before generating the binding code.

Page Designer and Examples

To actually create your own page, you can simply right-click the WebContent folder in the Project Navigator and select New|JSP File. You simply give the file a new name, click Enter, and you're in Page Designer, which supports both WYSIWYG editing like SDA and source editing like SEU.

Perhaps the best way to see what you can do is to start with an example. Click File|New|Example, and select iSeries to see a greatly enhanced list of shipped sample projects in V5.1.2. For example, select the Telephone Directory with Web Tools, and click Next and Finish to have it create a project that represents a very useful telephone directory application. This is similar to the application shipped in WebSphere Express, except that it is implemented entirely with RPG and the iSeries Web tools, and all the source is included. Another example is Order Entry with Web Tools, which implements a typical order entry application. **Figure 5** shows one of the JSP files from this application open in Page Designer.

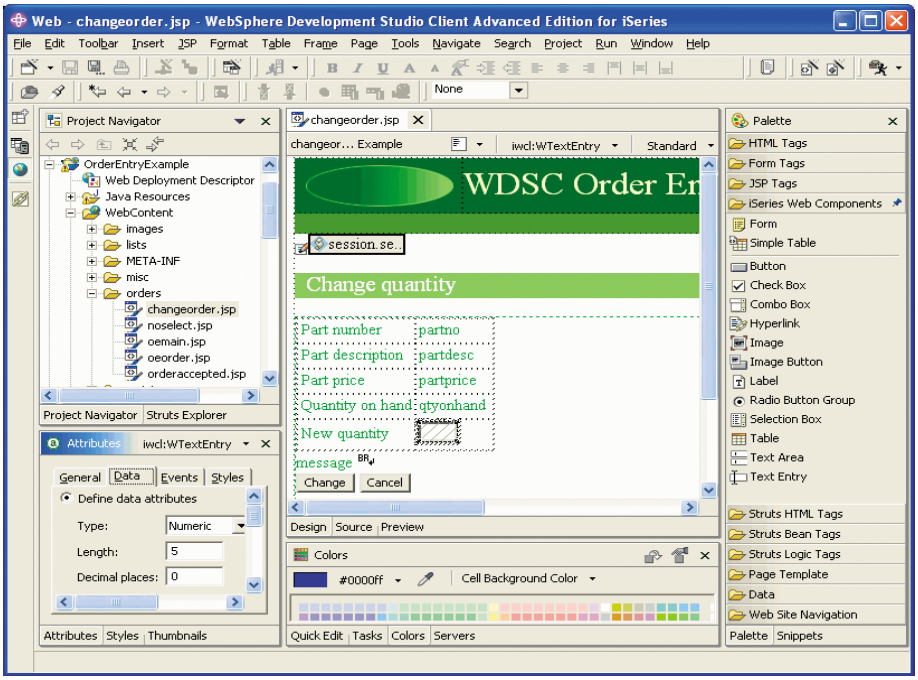

Figure 5: A sample application open in Page Designer

This order-entry application uses one of the supplied page templates. You can see on the right that a palette is associated with the editor, and an iSeries Web Components drawer has numerous Web controls that are optimized for iSeries development. An entry field is selected in the designer, and in the lower left panel are the attributes for the field, including the type, length, and decimals, which are used to generate client-side JavaScript for validity checking. If you scroll down, you see a Formatting button, where you can specify edit code and edit word information. Notice the Table part in the palette; this is a modern subfile. **Figure 6** shows a simple example of a table.

The attributes for the Table part allow you to specify an ILE service program that is used to populate it and deal with scrolling events. You can add other parts into the subfile for the cells, such as a combo box or hyperlink. For the hyperlink, you can substitute the values from other hidden columns, into the hyperlink or JavaScript value. Ultimately, the goal of all these controls is to make it possible for RPG or Cobol developers with SDA skills to create new Web pages with very little learning curve, while still allowing more advanced capabilities for those with some JavaScript skills. Of course, after you create your own pages, you can drag and drop them to the Web Diagram Editor. You can also run the Web Interaction Wizard to bind their fields to program parameters and the Submit button to RPG logic. You don't need to do this for tables, unless you want to make the cells editable.

For more advanced users, in V5.1.2, you can now change the font and color attributes of all the iSeries palette parts in your Web application, including those in generated Web pages, by expanding WebContent\theme\iSeriesWebComponents and double-clicking the iSeriesWebComponents.css file.

Other Capabilities

WDSc V5.1.2 has other Web tools capabilities, some of which are inherited from WebSphere Studio. For example, new tools help you create Faces pages that are based on the new JavaServer Faces industry support that may someday replace Struts.

Select	Part number	Description	Quantity	Unit Price	Extended Price
○	000001	Websphere redbook	12	$30.00	$360.00
○	000002	Radio Controlled Plane	23	$96.86	$2,227.78
○	000007	Feel Good Vitamins	50	$29.81	$1,490.50

Figure 6: A subfile-like table component

This approach makes it very easy to create database-centric Web applications that access the database directly, as opposed to going through business logic to get the data. If you need such an application, you will find these tools very compelling. They support any database, including UDB DB2, although they do not support logical files there, so you may have to create SQL views instead.

Support is also available in V5.1.2 for a new business logic language, if you want something portable but easier to learn than Java. This is Enterprise Generation Language (EGL). EGL is easy to learn for RPG and Cobol programmers, and it generates into Java or (with WDSc Advanced Edition) ILE Cobol on the iSeries. The tools for EGL are tightly integrated into the IDE.

The Bottom Line

The iSeries Web tools make it possible to build functioning Web applications with existing skills, by writing only RPG or Cobol business logic. They let you participate in the modern e-business world with your existing skills. And by combining Java with RPG or Cobol, they bring the best attributes of both to your new highly competitive applications. WDSc now contains numerous working examples to teach the tools and technology, and if you follow the Library link at ibm.com/software/awdtools/wds400, you'll find several additional tutorials. There is also a link for Larry's Tips, which provides many quick tips on how to fully exploit these tools. In our next article, we will look at similar tools for creating Web services from RPG or Cobol logic.

Phil Coulthard *works at the IBM Toronto lab, where he is the lead architect for application development tools and languages on iSeries.*

George Farr *works at the IBM Toronto lab, where he is the technical development manager for the RPG and VisualAge for RPG languages, as well as the new RPG and Cobol tools in WDSc. Phil and George are frequent speakers at many conferences and user groups worldwide, and their books Java for RPG Programmers, 2nd Edition and Java for S/390 and AS/400 COBOL Programmers, are available from Penton and MC Press.*

This article was first published in iSeries News magazine.