

Using IBM Rational Application Developer Version 7.5 to develop a Web 2.0 page that references a session bean

Reference an EJB session bean in a Web 2.0 Web page

Skill Level: Intermediate

[Kathy Endres \(endres@us.ibm.com\)](mailto:endres@us.ibm.com)
Senior software engineer
IBM

18 Nov 2008

This article assists those who are using [IBM® Rational® Application Developer for WebSphere Software Version 7.5](#) in implementing an Ajax-enabled Web 2.0 page. Additionally, it identifies how the RPC Adapter that is provided in the IBM® WebSphere® Application Server Feature Pack for Web 2.0 can be used to invoke an Enterprise Java™Beans (EJB) session bean for database access.

Introduction

[IBM® Rational® Application Developer for WebSphere Software Version 7.5](#) and later (hereafter: Rational Application Developer) provides new Web tools for developers to build asynchronous JavaScript and XML (Ajax) applications for IBM® WebSphere® Application Servers with the Feature Pack for Web 2.0. This article explains how you can use the new functionality in this feature pack to build a simple Web 2.0 page, including the ability to invoke methods on an EJB session bean on the server.

Tools used for this article

The following tools are used in this article:

- Enterprise JavaBeans V2 tools to retrieve or update data in a Derby embedded database.
- Remote Procedure Call (RPC) Adapter tools to expose EJB session bean methods to client-side code. The protocol used in this article is HTTP RPC.
- Dojo Web development tools to create a Web 2.0 page.

The Plants by WebSphere sample application

This article uses the Plants by WebSphere sample application, which is one of many samples provided with the IBM WebSphere Application Server Feature Pack for Web 2.0. This sample, which is also available within Rational Application Developer, demonstrates how a typical Java™ Platform 2, Enterprise Edition (J2EE) application can be enhanced with an Ajax styled architecture without rewriting the entire application (this is the original Plants by WebSphere application that is provided with the IBM WebSphere Application Server product).

The pages that were enhanced to provide users with a more interactive Web 2.0 experience include `index.html`, `orderinfo.jsp`, and `register.jsp`. These pages use Dojo widgets in the JavaScript Dojo Toolkit or custom Dojo widgets (for example, to support dragging catalog items to a shopping cart), or both.

Another feature of the WebSphere Application Server Feature Pack for Web 2.0 is the RPC Adapter that this sample application uses. The RPC adapter for IBM provides an HTTP interface to registered JavaBeans and Enterprise JavaBeans for remote access through the Web (for more information, see the link to "Getting started with RPC Adapter Libraries" in [Resources](#)). The RPC Adapter library can also be used to map a traditional J2EE construct to a lightweight construct, such as JavaScript Object Notation (JSON), which can then be easily used and rendered by JavaScript-based clients by using Dojo. Within the sample application, a POJO (plain old Java object) construct and a POJO RPC Adapter service are used to access methods on an EJB 2 session bean.

This article will identify how a subset of the back-order administration functionality, which was carried over from the original Plants by WebSphere application, can be implemented in a new Ajax-enabled Web page. Additionally, you will learn how you can take advantage of a new feature in Version 1.0.0.1 of the IBM WebSphere Application Server Feature Pack for Web 2.0. The new feature enables you to define an EJB RPC Adapter service for methods in an EJB 2 or 3.0 session bean, negating the need for creating a POJO proxy for the session bean. A subsequent section of this article will show how you can then call the URL for that service within JavaScript to read from or write to the Derby database. In summary, this new Web page will enable a user to view back-order inventory items that may be ordered from a supplier and allow such items to be canceled.

Web 2.0 Feature Pack, Version 1.0 versus 1.0.0.1

Another advantage of using Version 1.0.0.1 rather than Version 1.0 of the Web 2.0 Feature Pack is that it enables you to define services for overloaded methods (in a J2EE construct) that involve the same number of parameters and parameter names, but where the parameter types vary. With Version 1.0, this was a limitation.

Prerequisites

If you would like to follow the steps in this article to enhance the Plants by WebSphere sample application that is available within Rational Application Developer, please ensure you have the following software installed:

- [Rational Application Developer V7.5](#)
- One of the following servers:
 - WebSphere Application Server V6.1 with the Web 2.0 Feature Pack
 - WebSphere Application Server V7.0 with the Web 2.0 Feature Pack

Rational Application Developer provides development tools for these WebSphere servers, and test environment servers are bundled with Rational Application Developer for each supported version of WebSphere Application Server. This article uses WebSphere Application Server Version 7.0 -- more specifically, the WebSphere Application Server Version 7.0 Test Environment package that is bundled with Rational Application Developer.

If you install a WebSphere Application Server test environment server, be sure that you select the option to install the feature packs for it, as shown in Figure 1. See [Resources](#) for where to find additional information on WebSphere Application Server Feature Pack for Web 2.0, including a download link to facilitate installing it on a remote WebSphere Application Server.

Figure 1. Ensuring install of IBM WebSphere Application Server Feature Pack for Web 2.0



Comment regarding WebSphere Application Server V6.0.2

WebSphere Application Server V6.0.2 with the Web 2.0 Feature Pack supports Java 2 Platform, Enterprise Edition (J2EE) 1.4 applications with Enterprise JavaBeans (EJB) 2 session beans and a dynamic Web project enabled for Web 2.0. However, the version of the sample application that is available within Rational Application Developer, which is listed among prerequisites for this article, uses

one of these two servers.

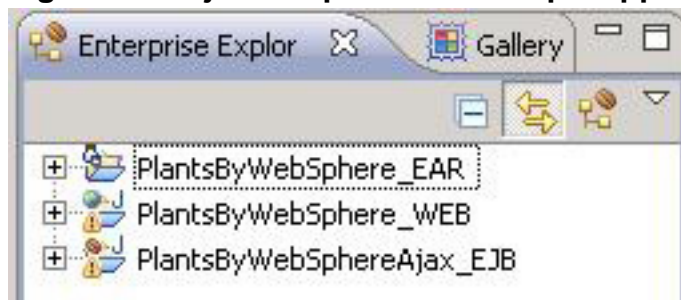
Accessing the Plants by WebSphere sample application

To use the Plants by WebSphere sample application from within Rational Application Developer, follow these steps:

1. Select **Help > Samples** from the menu bar.
2. In the **Samples** document, select Application **samples > Web > Ajax Web application using the Dojo Toolkit**.
3. Select the **Import sample** link.
4. Select the **Finish** button in the Import Sample Application window.

Three projects should appear in the Enterprise Explorer view as shown in Figure 2.

Figure 2. Projects imported for sample application



5. If a **Workspace Migration** window subsequently appears, navigate through its pages and select **Finish** on the last page.
6. Return to the sample Help topic and select the **Setup Instructions** link.
7. Read the tip that helps you determine whether you need to configure the database location. If you do, follow the steps to do so.
8. Follow the steps for running the sample on one of the supported WebSphere Application Servers so that you can get familiar with the sample application.

Implementing Backorder administration support

The BackOrderStock session bean in the sample application provides a method for retrieving all inventory items that have been backordered (ordered but delayed

awaiting resupply. Given that the new page will display only those items that may be ordered from a supplier, you need to extend the BackOrderStock session bean to provide an additional method to allow backordered inventory items to be retrieved by status.

Extending the BackOrderStock session bean

The EJB 2 tooling provided in Rational Application Developer can be used to extend the BackOrderStock session bean by following these steps:

1. Open the Deployment Descriptor for **PlantsByWebSphereAjax_EJB**.
2. Select the **Backorder** CMP entity bean from the Bean tab.
3. Define a new query for this bean, being sure to specify the values listed Table 1.
4. When you are finished, click the **Finish** button.

Table 1. Information to specify to define a new query

Field	Value
Method	New
Method type	Find method
Type	findByStatus
Parameters	status of type java.lang.String
Return Type	java.util.Collection
Sample query	Single Where Predicate
Query	select object(o) from Backorder o where o.status = ?1

5. Close the Deployment Descriptor editor and save the changes.
6. Select **PlantsByWebSphereAjax_EJB** and then **Java EE > Prepare for Deployment**.
7. Update the BackOrderStockBean session bean in PlantsByWebSphereAjax_EJB to use this new query. That is, add a `findBackOrderItemsByStatus(String status)` method in the BackOrderStockBean.java file. The updated version of this class is provided with this article (see [Downloads](#)).
8. Promote the new `findBackOrderItemsByStatus()` method to the

remote interface (from the drop-down menu, select **Enterprise Bean (1.x-2.x) > Promote to Remote Interface**).

9. Close **BackOrderStockBean**, saving the changes.
10. Select **PlantsByWebSphere_EAR** and then **Java EE > Prepare for Deployment**.

Exposing methods in the **BackOrderStock** session bean in an **RPC Adapter service**

The new Web page will need to call these methods from the enhanced **BackOrderStock** session bean:

- `findBackOrderItemsByStatus(String status)`
- `deleteBackOrder(String backOrderID)`

The Web tooling provided in Rational Application Developer enables you to expose these methods through an EJB RPC Adapter service by following these steps:

1. If you are not in the Web perspective, open or switch to it.
2. Open the **RPCAdapterConfig.xml** file, which is under **WebContent > WEB-INF** in the **PlantsByWebSphere_WEB** project.
3. Select the **Source** tab at the bottom of the **RPC Adapter Configuration Editor**.
4. The `xmlns` attribute on the `rpcAdapter` tag needs to be updated to reference the 1.0.0.1 schema for the RPC Adapter. To do this, append the `xmlns` attribute value with `/1.1` as shown here:
`xmlns="http://www.ibm.com/xmlns/prod/WebSphere/featurepack/v6.1/R"`
5. Close the editor, saving the changes.
6. Reopen the **RPCAdapterConfig.xml** file.
7. Select the **Design** tab in the **RPC Adapter Configuration Editor**.
8. Expand the **RPC Adapter** node in the tree and select the **Services** node.
9. Select the **Add** button.
10. From the **Add Item** dialog, select **EJB Session Bean** and then click **OK**.
11. To identify the 2 **BackOrderStock** session bean, specify the information

shown in Table 2.

Note:

The Name, Home Interface, and Remote Interface values are identified in the deployment descriptor for PlantsByWebSphereAjax_EJB and the EJB reference (ejb/BackOrderStock) that is defined in the deployment descriptor for PlantsByWebSphere_WEB identifies the JNDI value.

Table 2. Information to identify the BackOrderStock session bean

Field	Value
Name	BackOrderStock
JNDI	plantsbyajax/BackOrderStockHome
Type	Stateless
Home interface	com.ibm.websphere.samples.plantsbywebsphereejb.BackOrderStockHome
Is remote	true
Remote interface	com.ibm.websphere.samples.plantsbywebsphereejb.BackOrderStockHome

12. Select the **BackOrderStock** node in the tree and then click **Add**.
13. Select **Methods** in the Add Item dialog and then click **OK**.
14. Select the new **Methods** node in the tree and, for the Filter field in the Details section, select **White Listing**.
15. Select the new **()** node in the tree and then identify the findBackOrderItemsByStatus method by the values in Table 3.

Table 3. Information to identify the findBackOrderItemsByStatus method

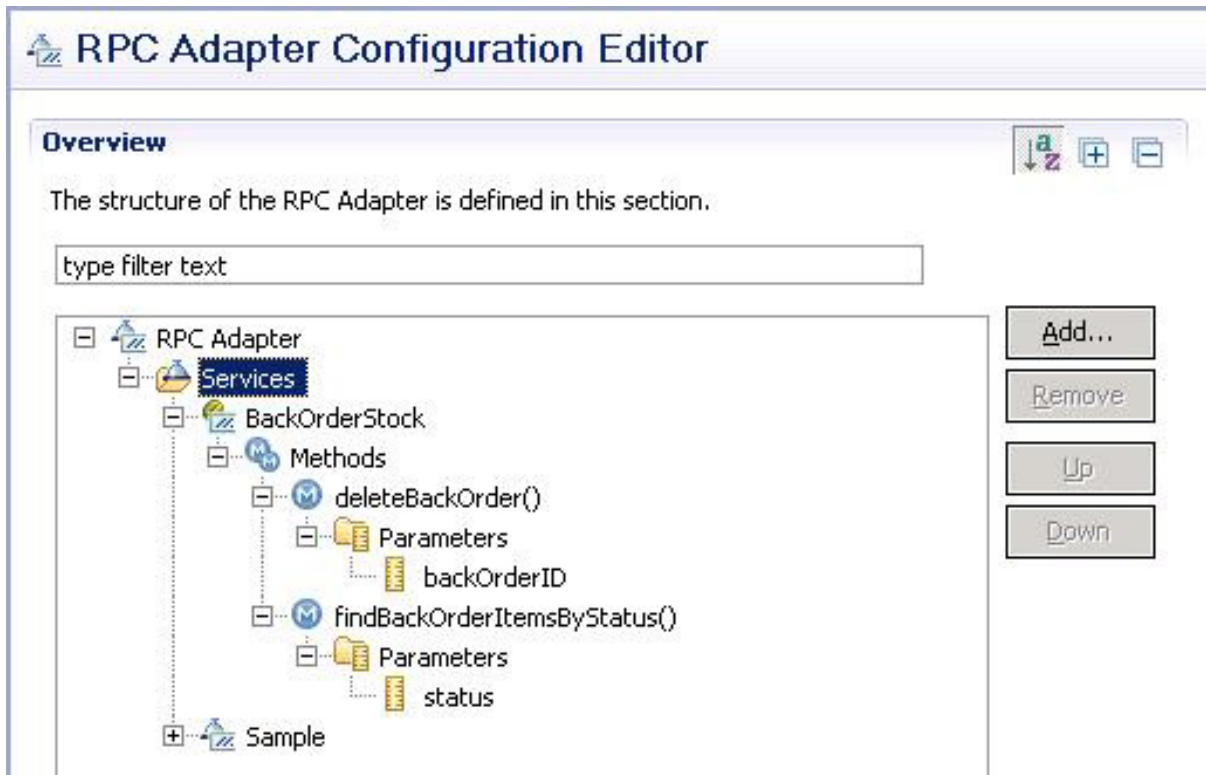
Field	Value
Name	findBackOrderItemsByStatus
Description	returns back order inventory items by status
HTTP Method	GET (since this method just retrieves data)

16. Use the editor to define the **status** parameter (which is of type java.lang.String) for findBackOrderItemsByStatus.
17. Define a method in the RPC Adapter Configuration editor for deleteBackOrder(), but specify **POST** for the HTTP Method field, because this method results in an update to the database.

The BackOrderStock EJB session bean service should appear in the Overview

section of the RPC Adapter Configuration Editor as shown in Figure 3.

Figure 3. BackOrderStock EJB session bean service defined in the RPC Adapter Configuration Editor



18. Close the RPC Adapter Configuration editor, saving the changes made.

Defining an EJB RPC Adapter Service in a new Web project

If you need to define an EJB RPC Adapter Service in a new or existing dynamic web project in which no RPCAdapaterConfig.xml file exists, follow these steps:

1. If the Web perspective is not currently open, open or switch to it.
2. Open **Properties** on the Web project and select **Project Facets**.
3. Ensure that the **Web 2.0 > Server-side technologies** facet is enabled and set to a version of **1.0.0.1**
4. Create a dummy POJO in a dummy package under **Java Resources: src** in the Web project and define a public method in this class.
5. From the Services view, select the **RPC Adapter** node

and then **Expose RPC Adapter Service** from the drop-down menu.

6. Browse to the dummy POJO class that you created in Step 4 and select the method that you defined in it. After you complete this wizard, an `RPCAdapterConfig.xml` file will appear in the Web project.
7. After you define the EJB RPC Adapter Service, remove the POJO RPC Adapter service that you created earlier and delete the dummy POJO class that it referenced. These are no longer needed.

Testing an RPC Adapter service

After you define a RPC Adapter Service, you can use Rational Application Developer to test it. To test the `findBackOrderItemsByStatus` method in the `BackOrderStock` service and to view the results in JavaScript Object Notation (JSON) data format, follow these steps:

1. If the `PlantsByWebSphere_WEB` project shows any of the following .jar (Java Archive) files under `WEB-INF > lib`, delete them, because they are old (**Note:** The 1.0.0.1 version of JAR files for the Feature Pack for Web 2.0 are provided as a shared library in the `PlantsByWebSphere_EAR` project. To see this, open the deployment descriptor for this EAR project and tab to the Deployment page. You should see a **Web 2.0 Feature Pack Library** entry under the Shared Libraries section):
 - `JSON4.jar`
 - `retroweaver-rt-1.2.3.jar`
 - `RPCAdapter.jar`
3. Select the **PlantsByWebSphere_WEB** project and then select **Run As > Run on Server** from the drop-down menu.
4. Select one of the WebSphere Application Servers identified previously in the Prerequisites section and then click **Finish**.
5. In the Web browser that appears, you should see a URL similar to this: `http://localhost:9080/PlantsByWebSphereAjax/`. To see a directory of deployed services for this Web project, update this URL by appending it with `servlet/RPCAdapter/httprpc/` and then press the **Enter** key (Note: This suffix maps to the URL mapping for the `RPCAdapter` servlet, which is defined in the deployment descriptor for the `PlantsByWebSphere_WEB` project).

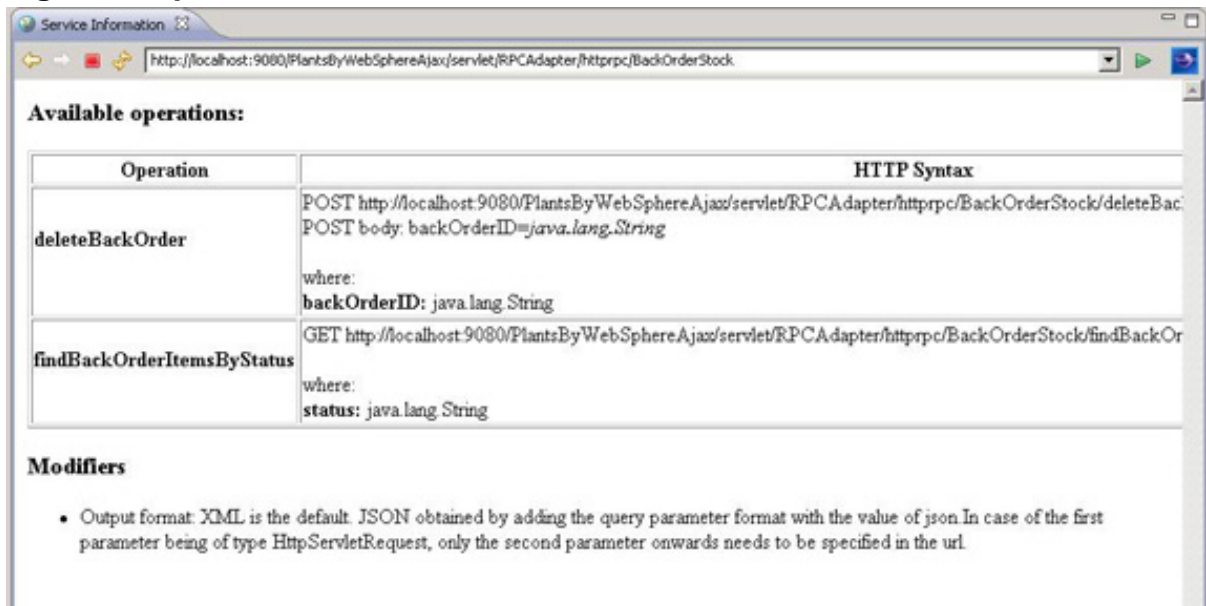
6. The Web browser should show two services in the directory of deployed services, as Figure 4 shows.

Figure 4. Directory of deployed services for PlantsByWebSphereAjax



6. Select the **Base URL** link that appears for the **BackOrderStock** service.
7. The Web browser should now show the operations that are available for that service, as shown in Figure 5. These operations map to the methods that were exposed to the service.

Figure 5. Operations defined for the BackOrderStock service



8. To test the findBackOrderItemsByStatus operation, perform these steps:
 - A. Copy and paste the http URL that appears for this operation into the URL field in the Web browser.
 - B. Replace =java.lang.String with Order Stock.

C. Press the Enter key.

10. Figure 6 shows the results that you should see in the Web browser.

Note:

If you do not see this, replace Order Stock with Order%20Stock to escape the space between the two words in the status parameter value).

The results are returned in XML format because the default data format specified in the RpcAdapterConfig.xml file for PlantsByWebSphere_WEB is set to XML.

Figure 6. XML results from the findBackOrderItemsByStatus operation



10. For this exercise, the Dojo Web page will request a JSON data format when calling this operation. To view the results in a JSON data format, append the URL with `&format=json` and press the Enter key.
11. When a File Download prompt appears, save the results to a file on your local file system. View the contents of the file to confirm that the file shows the same BACKORDER entry shown in Step 9. Notice the format of this file ("attribute name": "attribute value") as shown in Listing 1.

Listing 1. JSON result from the RPC Adapter service

```
{ "result": [ { "status": "Order Stock",
"lowDate": 1140033730659, "backOrderID": "3", "inventoryQuantity": 0,
"inventoryID": "V0007", "quantity": 350, "name": null, "orderDate": 0, "supplierOrderID": null } ] }
```

Testing services from the directory of

registered services

Keep in mind that your database may get updated in the process of testing any operation (method) that is defined in the RPCAdapterConfig.xml with a value of POST for the HTTP Method field.

Developing the Dojo-based client

Now, create the Web page that you will implement, using an Ajax-styled architecture, a subset of the functionality that exists in backorderadmin.jsp.

Creating a Web page

To create this Web page using Rational Application Developer, follow these steps:

1. Select the **PlantsByWebSphere_WEB** project in the Enterprise Explorer view and then select **New > Web Page** from the drop-down menu.
2. In the New Web Page wizard, specify a File Name of `backorderadminAjax.jsp` and then click **Finish**

Adding Dojo components to the Web page

If you open Properties on the PlantsByWebSphere_WEB project and select Project Facets, you will see that these two facets are enabled:

- Web 2.0 > Dojo Toolkit (The version of this facet does not reflect the version of the Dojo Toolkit being used. To see the version being used or to identify which one to use, select **Properties** from the Web project's drop-down menu and then select **Dojo Toolkit**).
- Web 2.0 > Server-side technologies

The first facet enables you to use Dojo widgets within the new Web page. The second facet enables you to use the RPC Adapter to invoke a method on the BackOrderStock session bean on the server. The data that is returned in JSON data format will then be rendered in the new Web page.

Here are the steps for developing this new Web page in Rational Application Developer:

1. If you closed the backorderadminAjax.jsp when it appeared earlier within an editor, reopen it.

2. Select the **Split** tab at the bottom of the editor.
3. Select the **Preview + Source** editor toolbar button at the top of the editor. This toolbar button is shown circled in red in Figure 7. This split view is recommended for Dojo developers who are using Rational Application Developer V7.5.

Figure 7. Preview + Source editor toolbar button circled in red



4. In your Web perspective, open the **Palette** view. Notice the drawers that appear for Dojo development.
5. Begin by developing the layout of the Web page:
 - A. Drop **BorderContainer** from the **dojo – Layout** palette drawer in between the `<body>` and `</body>` tags in the Source pane. BorderContainer is a new widget as of Version 1.1 of the Dojo Toolkit and is recommended over LayoutContainer, which is deprecated. (The Source pane provides more precise placement of the Dojo tags).
 - B. Place the cursor within the div tag for the BorderContainer component. Then, in the **Properties** view, select the button to the right of the **Style: Properties** field. Select **Layout** and then specify a width of 600 pixels and a height of 400 pixels.
 - C. Drop three **Content Panes** from the **dojo – Layout** palette drawer within the content of the border container (between its `<div>` and `</div>` tags).

Use Content Assist (Ctrl+Spacebar) to add a region attribute within the `<div>` tag of each content pane. Specify values of "top," "center," and "bottom," respectively.

Listing 2. Defined Web page layout

```
<div dojotype="dijit.layout.BorderContainer" style="height: 400px; width: 600px">
<div preload="true" dojotype="dijit.layout.ContentPane" href="" region="top"></div>
<div preload="true" dojotype="dijit.layout.ContentPane" href="" region="center"></div>
```

6. Listing 3 shows the code developed for the top content pane, which will contain a heading and instructions.

Listing 3. Content of the top content pane

```
<div preload="true" dojotype="dijit.layout.ContentPane" href="" region="top">
<h1><b style="color: green">Backorder Administration</b></h1>
<br>
<h3>Back Order Items</h3>>
<br>
The <b>Back Order Items</b> list shows the inventory items that may be ordered from a
supplier. Select one or more ordered items and click the <b>Order Stock</b> button to
send an order to the supplier. The <b>QUANTITY TO ORDER</b> may be changed before
the order is submitted. <br><br>
```

7. The next content pane will use a Dojo grid to display the list of backorder items that may be ordered from a supplier. To develop this, follow these steps:
 - A. Drop Grid from the dojo – Other palette drawer within the content of the second content pane (between its <div> and </div> tags).
 - B. Update the attribute values for the grid control as indicated in Table 4. Later in this article, you will see JavaScript code that references these attribute values.

Table 4. Attributes to define for the grid control

Attribute	Attribute value
id	backOrder_grid
model	backOrder_grid_model
structure	backOrder_grid_layout

8. The last content pane will contain Dojo buttons that enable a user to act on a selected item in the grid.
 - A. Drop two **Buttons** from the **dojo – Form** palette drawer within the content of the second content pane (between its <div> and </div> tags).
 - B. Add an **ID attribute** and replace the content of each button component as shown in Listing 4.

Listing 4. Button definitions for the last content pane

```
<div preload="true" dojotype="dijit.layout.ContentPane" href="" region="bottom">
<div id="orderButton" dojotype="dijit.form.Button">Order Stock</div>
<div id="cancelButton" dojotype="dijit.form.Button">Cancel</div>
```

9. Close backorderadminAjax.jsp, saving your changes.

Adding JavaScript code to further define the Dojo grid

Listing 5 shows some variables that were added within the <script> tag to initialize the grid (for example, to define its columns, grid layout, and model). If you examine the code further, you will see that a NumberTextBox is identified for the Quantity to Order column so that its value can be edited and that a format is specified for the Low Inventory Date column to display its value in this format: MM/dd/yyyy hh:mm:ss.

Content Assist feature

Content Assist is available for any components in the Dojo Toolkit (dojo, dojox, or dijit, for example) that are used within the <script> tag. You can activate the Content Assist tool simply by pressing Ctrl+Spacebar.

Listing 5. JavaScript code to further define the Dojo grid

```
var clickedOrder = null;
var selectedItemIndex = -1;

// a grid view is a group of columns
// return (new Date(value)).toString();},
var backOrder_grid_view = { cells: [[
  {name: 'BACK ORDER #'},
  {name: 'ITEM #'},
  {name: 'ITEM DESCRIPTION', width: "200px" },
  {name: 'QUANTITY TO ORDER',
    editor: dojox.grid.editors.Dijit,
    editorClass: "dijit.form.NumberTextBox" },
  {name: 'CURRENT INVENTORY QUANTITY' },
  {name: 'LOW INVENTORY DATE',
    formatter: function (value) {
      var invDate = new Date(value);
      //format the date value so it appears as "MM/dd/yyyy
hh:mm:ss"
      //adjust the zero-based month value
      var month = convertSingleCharacter(invDate.getMonth() +
1);
      var day = convertSingleCharacter(invDate.getDate());
      var invDateString = month + "/" + day + "/" +
invDate.getFullYear();
      var hours = convertSingleCharacter(invDate.getHours());
      var minutes =
convertSingleCharacter(invDate.getMinutes());
      var seconds =
convertSingleCharacter(invDate.getSeconds());
      var invTimeString = hours + ":" + minutes + ":" +
seconds;
      var invFullString = invDateString + " " + invTimeString;
      return invFullString; },
      width: "100px" }
  ] ]};

// a grid layout is an array of views
var backOrder_grid_layout = [ backOrder_grid_view ];

data = [];
backOrder_grid_model = new dojox.grid.data.Table(null, data);
```

```
//disable multiple selection
dojox.grid.selection.multiSelect = false;
```

Adding JavaScript code to populate the Dojo grid and identify event handlers

Listing 6 shows JavaScript code that populates the Dojo grid and adds event handlers. The `init()` function is called when the Web page is loaded. This function in turn calls the `populateTable()` function and `addEventHandlers()` function. The `init()` function calls the EJB RPC Adapter service, `BackOrderStock`, through its URL and indicates to return the results in JSON data format. The `populateTable()` function identifies what function to call when the user clicks the Cancel button or a row in the Dojo grid (Note: Support of the Order Stock button is left as an exercise).

Listing 6. JavaScript code for the grid table and event handlers

```
dojo.addOnLoad(init);

function init() {
    populateTable();
    addEventHandlers();
}

function populateTable() {
    //fire async call to prime the table with the JSON results from the
    //HTTP GET request
    dojo.xhrGet({
        url:
        "servlet/RPCAdapter/httprpc/BackOrderStock/findBackOrderItemsByStatus?status
        =Order%20Stock&format=json",
        load: backOrdersByStatusCallBack,
        handleAs: "json",
        error: function(type, error) {
            console.debug("Error in populateTable!" + error.message)}
    });
}

function addEventHandlers() {
```

Listing 7 shows the code for the function that gets invoked when the `findBackOrderItemsByStatus` method in the `BackOrderStock` EJB session bean asynchronously completes. This callback code pushes the JSON results returned from calling the exposed EJB session bean method on the server to the data grid.

Listing 7. Callback code to process the JSON results for the exposed `findBackOrderItemsByStatus` method

```
function backOrdersByStatusCallBack(response, ioArgs){
    var backorderItems = response.result;

    for (i=0; i<backorderItems.length; i++)
```

```

    {
        data.push([backorderItems[i].backOrderID,
                  backorderItems[i].inventoryID,
                  backorderItems[i].name,
                  backorderItems[i].quantity,
                  backorderItems[i].inventoryQuantity,
                  backorderItems[i].lowDate]);
    }
    backOrder_grid_model.setData(data);

```

Adding JavaScript code for the event handlers

The remaining code to process events on the grid (`onRowClick`) and the Cancel button (`onClick`) is included in Listing 8, which also shows the supporting function: `convertSingleCharacter`

Listing 8. Remaining JavaScript functions

```

function resetItemSelection() {
    clickedOrder = null;
    selectedItemIndex = -1;
}

function setItemSelection(event) {
    selectedItemIndex = event.rowIndex;
    var backOrder = dijit.byId('backOrder_grid').model.getRow(selectedItemIndex);
    clickedOrder = backOrder[0]; // get the backOrderId of the selected row
}

function cancelOrder() {
    if (clickedOrder == null)
        alert ('Error: you must select a BackOrder you wish to cancel');
    else {
        //else get backorderId for that row
        var selectedContent = {};
        selectedContent['backOrderID'] = clickedOrder;
        resetItemSelection();
        //call the xhrPost
        dojo.xhrPost({
            url: "servlet/RPCAdapter/httprpc/BackOrderStock/deleteBackOrder&format=json",
            content: selectedContent,
            load: cancelOrderCallBack,
            handleAs: "json",
            error: function(type, error) {
                console.debug("Error in cancelOrder!" + error.message)
            }
        });
    }
}

function cancelOrderCallBack() {
    alert("The cancellation request has been submitted and processed.");
}

//appends value with a "0" if it's a single numeric character

```

Final touches

In addition to the JavaScript code shown previously, the following modifications were made:

- Updated the **djConfig** attribute value in the `<script>` tag to suppress a warning message that appears at runtime. This warning suggests using `mimetype:text/json-comment-filtered` to avoid potential security issues. Because you are using a service from a known source (your own Web page), it is
- The enhancement that was made is shown in bold text in Listing 9.

Listing 9. Update made to the **djConfig** attribute in the `<script>` tag

```
<script type="text/javascript" src="dojo/dojo.js"
      djConfig="isDebug: true, parseOnLoad: true, usePlainJson: true">
```

- Changed the class attribute value of the `<body>` tag to `tundra`.
- Added two `dojo.require` statements within the content of the `<script>` tag, as shown in Listing 10.

Listing 10. Additional `requires` statements that were added manually

```
//the following requires statements were manually added
dojo.require("dijit.form.NumberTextBox");
dojo.require("dojox.grid._data.model");
```

- Updated the `<style>` tag to enhance the grid's look and feel, as shown in Listing 11.

Listing 11. Additions made to the `<style>` tag to make the grid look better

```
@import "dojox/grid/_grid/tundraGrid.css";
#grid {
    border: 1px solid #333;
    padding: 1px;
}
```

That completes the development of `backorderadminAjax` JavaServer page.

Testing the Dojo-based Web page

To test the `backorderadminAjax.jsp` file on WebSphere Application Server V7 from within Rational Application Developer, follow these steps:

1. Select **backorderadminAjax.jsp** in the Enterprise Explorer and then select **Run As > Run** on Server from the drop-down menu.
2. In the Run on Server dialog, select **WebSphere Application Server v7.0** and then click **Finish**.

Figure 8. backorderadminAjax.jsp at run time

Backorder Administration

Back Order Items

The **Back Order Items** list shows the inventory items that may be ordered from a supplier. Select one or more ordered items and click the **Order Stock** button to send an order to the supplier. The **QUANTITY TO ORDER** may be changed before the order is submitted.

BACK ORDER #	ITEM #	ITEM DESCRIPTION	QUANTITY TO ORDER	CURRENT INVENTORY QUANTITY	LOW INVENTORY DATE
3	V0007	Watermelon	<input type="text" value="350"/>	50	02/15/2006 12:02:10

Order Stock Cancel

Notice that the grid is populated with the same backorder that appeared when you tested the `findBackOrderItemsByStatus` operation for the `BackOrderStock` RPC Adapter service. You can also run `backorderadmin.jsp` and compare the runtime results. For example, notice how much more interactive the grid is in `backorderadminAjax.jsp`. You can resize its columns, and if you enter more backorders in the database with a status of "Order Stock," you can also alter sorting of data in the grid by clicking a column.

3. Without clicking any row in the grid, click **Cancel**. An alert should appear informing you to first select an item from the grid.
4. If you double-click within the **Quantity to Order** column, a `dijit.form.NumberTextBox` will appear to allow you to edit the value. Entering a non-numeric value should cause an error notification.
5. Select the backorder that is listed in the grid and then click **Cancel**. An

alert should appear when the backorder has been canceled.

Note:

Showing an alert is not a recommended practice if you are developing a Web page for production use. But because the focus of this article is on defining, calling, and displaying the results of an EJB 2 session bean method, we kept this Cancel callback behavior simple.

Debugging JavaScript code

If you run into any problems in running a Web page that contains JavaScript, you can use Firebug to debug it. This is a great tool to use for debugging JavaScript, as well as CSS, HTML, and DOM. To enable Firebug within Rational Application Developer:

1. Select **Window > Preferences** from the Rational Application Developer menu bar.
2. Expand **General** and then select **Web Browser**.
3. Select the Use external Web browser radio button.
4. Select the **Firefox with Firebug** check box and then click **OK**.

Recap of what you have learned

By using Rational Application Developer V7.5 functionality, you developed an Ajax Web page. Additionally, you used the RPC Adapter configuration editor provided within Rational Application Developer to define an EJB RPC Adapter service for an EJB session bean (BackOrderStock) for reading from or writing to the Derby database. Handwritten JavaScript was added to access the EJB RPC Adapter service through its URL and display the JSON result in a Dojox grid on the Web page.

Acknowledgements

Many thanks to Jim Zhang, Daniel Lee, Justin Berstler, and Kevin Haverlock for their technical and editorial contributions to this article. Thanks also to Mohan S. Saboji and Jarett Stein for providing references to assist in writing this article and to Jared Jurkiewicz for his assistance in coding the grid support.

Downloads

Description	Name	Size	Download method
Code sample	BackOrderStockBackup	40KB	HTTP

[Information about download methods](#)

Resources

Learn

- For more information about the RPC Adapter, visit [Getting started with RPC Adapter Libraries](#). This information provides a great overview of RPC Adapter and provides some detailed feature information, including information pertinent to defining an EJB 3.0 session bean service.
- The developerWorks article [Create Ajax-style architectures with the IBM Web 2.0 Feature Pack](#), by Kevin Haverlock, shows how the original form of the "Plants by WebSphere" application was enhanced with Ajax-style architecture by using the IBM WebSphere Application Server Feature Pack for Web 2.0.
- Understand more about how Rational Application Developer Version 7.5 provides solutions to the many challenges that software teams face when tasked to deliver Web and service-oriented applications by reading this article [Why Rational Application Developer for WebSphere Software Version 7.5?](#).
- Read [What's new in IBM Rational Application Developer Version 7.5](#) by James Chung (IBM developerWorks, November 2008).
- Read [Web 2.0 Application Development using JPA, AJAX, and Dojo tools in Rational Application Developer Version 7.5](#) by Jarett Stein (IBM developerWorks, September 2008). This article demonstrates how to use the RPC Adapter tools to expose JPA methods as a JSON-RPC Web service.
- Understand more about JavaServer Faces tooling in Rational Application Developer Version 7.5 by reading this article [What's new in JavaServer Faces tooling in Rational Application Developer Version 7.5?](#).
- Read [Developing Web applications with the Java Persistence API and JavaServer Faces](#) by Thomas Mutdosch (IBM developerWorks, August 2008).
- IBM's Education Assistant provides a [Web presentation with audio](#) about the WebSphere Application Server Web 2.0 Feature Pack.
- Visit the [Rational software area on developerWorks](#) for technical resources and best practices for Rational Software Delivery Platform products.
- Explore [Rational computer-based, Web-based, and instructor-led online courses](#). Hone your skills and learn more about Rational tools with these courses, which range from introductory to advanced. The courses on this catalog are available for purchase through computer-based training or Web-based training. Additionally, some "Getting Started" courses are available free of charge.
- Subscribe to the [Rational Edge newsletter](#) for articles on the concepts behind effective software development.

- Subscribe to the [IBM developerWorks newsletter](#), a weekly update on the best of developerWorks tutorials, articles, downloads, community activities, webcasts and events.
- Browse the [technology bookstore](#) for books on these and other technical topics.

Get products and technologies

- Download the [trial version of Rational Application Developer](#).
- Download [trial versions of IBM Rational software](#).
- Download these [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Tivoli®, and WebSphere®.
- [DojoToolkit.org](#) provides documentation, downloads, and community forums for the Dojo OpenSource toolkit written in JavaScript.

Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).
- Join the [Rational Application Developer forum](#) on developerWorks.

About the author

Kathy Endres

Kathy has been testing Web tooling at IBM for the last seven years. As a member of the Rational Application Developer System Verification Test team, she designed and developed test applications to simulate use in a customer-like environment.

Trademarks

IBM and the IBM logo are trademarks of International Business Machines Corporation in the United States, other countries or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.