Using the Spry Framework

Acknowledgments

Project Management: Charles Nadeau

Writing: Jon Michael Varese

Production Management: Adam Barnett

Media Design and Production: Adam Barnett, Masayo Noda

First Edition: January 2006

Adobe Systems Incorporated
601 Townsend St.
San Francisco, CA 94103

Contents

Using the Spry framework

This help, which outlines how to use the Spry framework, contains the following topics:

About AJAX and the Spry framework

This section contains the following topics:

- "About AJAX" on page 5
- "About the Spry framework" on page 6

About AJAX

Asynchronous JavaScript and XML, or AJAX, is a concept for how Web developers can use various techniques to update web pages without requiring visible refreshes and without the need for browser add-on technologies like Flash, Java, or ActiveX. AJAX is not a product, company, or trademark.

Implementing AJAX features often requires knowledge of JavaScript, XML, and the Document Object Model (DOM). Spry provides a light-weight, HTML-centric framework that makes the task simpler.

For more information about AJAX, see http://en.wikipedia.org/wiki/AJAX.

About the Spry framework

The Spry framework is a JavaScript library that provides web designers with the ability to build web pages that offer richer experiences to their users. It is a light-weight framework designed to bring AJAX to the web-design community. The first release of the Spry framework is a preview of the data capabilities that enable designers to incorporate XML data into their HTML documents using HTML, CSS, and a minimal amount of JavaScript, without the need for refreshing the entire page. The Spry framework is HTML-centric, and easy to implement for users with basic knowledge of HTML, CSS and JavaScript. The framework was designed such that the markup is simple and the JavaScript is minimal.

The Spry framework is meant primarily for users who are web design professionals or advanced non-professional web designers. It is not meant to be a full web application framework for enterprise-level web development (though it can be used in conjunction with other enterprise-level pages).

How Spry pages work

This section contains the following topics:

- "Anatomy of the Spry data set" on page 6
- "Anatomy of the Spry dynamic region" on page 9
- "Anatomy of the Spry basic master/detail dynamic region" on page 11
- "Anatomy of master/detail dynamic regions that depend on more than one data set" on page 13

Anatomy of the Spry data set

A Spry data set is a JavaScript object. With a few snippets of code in your web page, you can create this object and load data from an XML source into it when the user opens the page in a browser. The data set results in a flattened array of XML data that can be visualized as a standard table containing rows and columns.

For example, suppose you have an XML source file, cafetownsend.xml, that contains the following information:

```
<?xml version="1.0" encoding="UTF-8"?>
<specials>
  <menu_item id="1">
    <item>Summer Salad</item>
    <description>organic butter lettuce with apples, blood oranges,
  gorgonzola, and raspberry vinaigrette.</description>
    <price>7</price>
  </menu_item>
  <menu_item id="2">
    <item>Thai Noodle Salad</item>
    <description>lightly sauteed in sesame oil with baby bok choi,
  portobello mushrooms, and scallions.</description>
    <price>8</price>
  </menu_item>
  <menu_item id="3">
    <item>Grilled Pacific Salmon</item>
    <description>served with new potatoes, diced beets, Italian parlsey,
  and lemon zest.</description>
    <price>16</price>
  </menu_item>
</specials>
```

The data set flattens the XML data into an array of objects (rows) and properties (columns) that can be visualized as the following table:

@id	item	description	price
1	Summer salad	organic butter lettuce with apples, blood oranges, gorgonzola, and raspberry vinaigrette.	7
2	Thai Noodle Salad	lightly sauteed in sesame oil with baby bok choi, portobello mushrooms, and scallions.	8
3	Grilled Pacific Salmon	served with new potatoes, diced beets, Italian parlsey, and lemon zest.	16

The data set contains a row for each menu item and the following columns: @id, item, description, and price. The columns represent the child nodes of the specials/menu_item node in the XML, plus any attributes contained in the menu_item tag, or in any of the child tags of the menu_item tag.

The data set also contains a built-in data reference called `ds_RowID` (not shown) that can be useful later when you display your data.

You create a Spry data set object by using the `Spry.Data.XMLDataSet` constructor. The default structure (or schema) of the data set is defined by the XML data source. For example, if the data source is a repeating XML node that contains three child nodes, the data set will have a row for each repeating node, and a column for each of the three child nodes. (If any of the repeating nodes or child nodes contain attributes, the data set also creates a column for each attribute.)

Once created, the data set object lets you easily display and manage the data. For example, you can create a simple table that displays the XML data, and then use simple methods and properties to reload data, sort and filter data, or page through data.

The following example illustrates how you would create a Spry data set called dsSpecials, and load data from an XML file called cafetownsend.xml:

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /
  >
  <title>Spry Example</title>
  <!--Link the Spry libraries-->
  <script type="text/javascript" src="includes/xpath.js"></script>
  <script type="text/javascript" src="includes/SpryData.js"></script>
  <!--Create a data set object-->
  <script type="text/javascript">
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
  </script>
</head>

<body>
</body>
```

NOTE: The examples in this document are for reading purposes only and not intended for execution. For working samples, see the demos folder in the Spry_P1_1_06-08 folder. For more information, see "To prepare your files" on page 18.

In the example, the first `<script>` tag links an open-source XPath library to the page where you'll eventually display XML data. The XPath library allows for the specification of more complex XPath when you create a data set:

```
<script type="text/javascript" src="includes/xpath.js"></script>
```

The second `<script>` block links the Spry data library, SpryData.js, which is stored in a folder called includes on the server:

```
<script type="text/javascript" src="includes/SpryData.js"></script>
```

The Spry data library is dependent on the XPath library, so it's important that you always link the XPath library first.

The third `<script>` block contains the statement that creates the data set called dsSpecials. The XML source file, cafetownsend.xml, is stored in a folder called data on the server:

```
var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
```

In JavaScript the operator `new` is used to create objects. The `Spry.Data.XMLDataSet` method is a constructor in the Spry data library that creates new Spry data set objects. The constructor takes two parameters: the source of the data (`"data/cafetownsend.xml"`) and an XPath expression that specifies the node or nodes in the XML to supply the data (`"specials/menu_item"`).

You can also specify a URL as the source of the XML data, as follows:

```
var dsSpecials = new Spry.Data.XMLDataSet("http://www.somesite.com/
  somefolder/cafetownsend.xml", "specials/menu_item");
```

NOTE

The URL you decide to use (whether absolute or relative) is subject to the browser's security model, which means that you can only load data from an XML source that is on the same server domain as the HTML page you're linking from. You can get around this limitation by providing a cross-domain service script. For more information, consult your server administrator.

In the example, the constructor creates a new Spry data set object called dsSpecials. The data set obtains data from the specified specials/menu_item node in the XML file cafetownsend.xml and converts the data to a flattened array of objects and properties, similar to the rows and columns of a table. (See the beginning of this section for a visualization of the table.)

Each data set maintains the notion of a "current row." By default, the current row is set to the first row in the data set. Later, you can change the current row programatically by calling the `setCurrentRow()` method on the data set object. For more information, see "Anatomy of the Spry basic master/detail dynamic region" on page 11.

Anatomy of the Spry dynamic region

Once you've created a Spry data set, you can display the data in a Spry dynamic region. A Spry dynamic region is an area on a web page that's bound to a data set. The region displays the XML data from the data set and automatically updates the data display whenever the data set is modified. You declare a Spry dynamic region in a container tag using the `spry:region` attribute. Most HTML elements can act as dynamic region containers, however, the following tags *cannot* be used:

- `col`
- `colgroup`
- `frameset`
- `html`
- `iframe`
- `style`
- `table`
- `tbody`
- `tfoot`
- `thead`

- `title`
- `tr`

While you cannot use any of the above HTML elements as Spry dynamic region containers, you are allowed to use them *inside* Spry dynamic region containers.

NOTE Dynamic regions are limited to regions within the `body` tag. You can't add the `spry:region` attribute to any tag that is outside the `body` tag.

In the following example, we've created a container for a dynamic region called Specials_DIV using a `div` tag that includes a standard HTML table. Tables are typical HTML elements used for dynamic regions because the first row of the table can contain headings, and the second row can contain repeated XML data.

```
<!--Create the Spry dynamic region-->
<div id="Specials_DIV" spry:region="dsSpecials">
  <!--Display the data in a table-->
  <table id="Specials_Table">
    <tr>
      <th>Item</th>
      <th>Description</th>
      <th>Price</th>
    </tr>
    <tr spry:repeat="dsSpecials">
      <td>{item}</td>
      <td>{description}</td>
      <td>{price}</td>
    </tr>
  </table>
</div>
```

In the example, the `div` tag that creates the container for the dynamic region needs only two attributes: a `spry:region` attribute that declares the dynamic region and specifies the data set to use in it, and an `id` attribute that names the region:

```
<div id="Specials_DIV" spry:region="dsSpecials">
```

The new region is an "observer" or "listener" of the dsSpecials data set. Any time the dsSpecials data set changes, the new dynamic region regenerates itself with the updated data.

An HTML table displays the data:

```
<table id="Specials_Table">
    <tr>
      <th>Item</th>
      <th>Description</th>
      <th>Price</th>
    </tr>
    <tr spry:repeat="dsSpecials">
      <td>{item}</td>
```

```
        <td>{description}</td>
        <td>{price}</td>
      </tr>
    </table>
```

The values in curly braces in the second row of the table specify the columns in the data set. The curly braces bind the table cells to the data in specific columns of the data set. Because XML data often includes repeating nodes, the example also declares a `spry:repeat` attribute in the second table row tag. This causes all of the rows in the data set to appear when the user loads the page (instead of just the data set's current row).

Anatomy of the Spry basic master/detail dynamic region

When working with Spry data sets, you can create master/detail dynamic regions to display more detail about your data. One region on the page (the master), drives the display of the data in another region on the page (the detail). Typically, the master region displays an abbreviated form of a set of records from the data set, and the detail region displays more information about a selected record. Because the detail region is dependent on the master region, any time the data in the master region changes, the data in the detail region changes as well.

This section covers basic master/detail relationships where both regions are dependent on the same data set. For information on master/detail regions that use more than one data set, see "Anatomy of master/detail dynamic regions that depend on more than one data set" on page 13.

In the following example, a master dynamic region displays data from the data set dsSpecials, and a detail dynamic region displays more detail about the row of data that's been selected in the master region:

```
<head>
. . .
<script type="text/javascript" src="../includes/xpath.js"></script>
<script type="text/javascript" src="../includes/SpryData.js"></script>
<script type="text/javascript">
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");</script>
</head>
. . .
<body>
  <!--Create a master dynamic region-->
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table">
      <tr>
        <th>Item</th>
```

```
         <th>Description</th>
         <th>Price</th>
       </tr>
     <!--User clicks to reset the current row in the data set-->
     <tr spry:repeat="dsSpecials"
   onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
         <td>{item}</td>
         <td>{description}</td>
         <td>{price}</td>
       </tr>
     </table>
   </div>
   <!--Create the detail dynamic region-->
   <div id="Specials_Detail_DIV" spry:detailregion="dsSpecials">
     <table id="Specials_Detail_Table">
     <tr>
       <th>Ingredients</th>
       <th>Calories</th>
     </tr>
     <tr>
       <td>{ingredients}</td>
       <td>{calories}</td>
     </tr>
   </table>
</div>
. . .
</body>
```

NOTE The example XML file in "Anatomy of the Spry data set" on page 6 does not contain nodes for ingredients or calories. We've added these nodes to the data set for purposes of this example.

In the example, the first `div` tag contains the `id` and `spry:region` attributes that create a container for the master dynamic region:

```
<div id="Specials_DIV" spry:region="dsSpecials">
```

The first table row tag of the master region contains an onclick event handler that resets the value of the current row in the data set.

```
<tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
```

The second `div` tag contains the attributes that create a container for the detail dynamic region:

```
<div id="Specials_Detail_DIV" spry:detailregion="dsSpecials">
```

Every Spry data set maintains the notion of a "current row." By default, the current row is set to the first row in the data set.

The binding expressions in the detail region (`{ingredients}` and `{calories}`) display data from the data set's current row when the page loads in a browser. When a user clicks a row in the master region table, however, the onclick handler changes the current row in the data set to the row the user selected.

The `{ds_RowID}` data reference is a built-in part of the Spry framework that points to an automatically-generated unique ID for each row in the data set. (See "Anatomy of the Spry data set" on page 6.) When the user selects a row in the master region table, the onclick event handler supplies the unique ID to the `setCurrentRow` method, which causes the resetting of the current row in the data set.

Whenever the data set is modified, all dynamic regions bound to that data set regenerate themselves and display the updated data. Since the detail region, like the master region, is an observer of the dsSpecials data set, it also changes as a result of the modification, and displays data related to the row the user selected (the new current row).

The difference between a `spry:region` and a `spry:detailregion` is that the `spry:detailregion` specifically listens for CurrentRowChange notifications (in addition to DataChanged notifications) from the data set, and updates itself when it receives one. Normal `spry:regions`, on the other hand, ignore the CurrentRowChange notification, and only update when they receive a DataChanged notification from the data set.

Anatomy of master/detail dynamic regions that depend on more than one data set

In some cases, you might want to create master/detail relationships that involve more than one data set. For example, you might have a list of menu items that has a great deal of detail information associated with it. (This section uses a list of ingredients to illustrate the point.) Fetching all of the information associated with every menu item in a single query might be an inefficient use of bandwidth not to mention unnecessary, given that many users might not even be interested in the details of everything on the menu. Instead, it would be more efficient to download only the detail data that the user is interested in when the user requests it, thus improving performance and reducing bandwidth. Limiting the amount of data exchange in this way is a common technique used to improve performance in AJAX applications.

Following is the XML source code for a sample file called cafetownsend.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<specials>
  <menu_item id="1">
    <item>Summer Salad</item>
    <description>organic butter lettuce with apples, blood oranges,
  gorgonzola, and raspberry vinaigrette.</description>
    <price>7</price>
    <url>summersalad.xml</url>
  </menu_item>
  <menu_item id="2">
    <item>Thai Noodle Salad</item>
    <description>lightly sauteed in sesame oil with baby bok choi,
  portobello mushrooms, and scallions.</description>
    <price>8</price>
    <url>thainoodles.xml</url>
  </menu_item>
  <menu_item id="3">
    <item>Grilled Pacific Salmon</item>
    <description>served with new potatoes, diced beets, Italian parlsey,
  and lemon zest.</description>
    <price>16</price>
    <url>salmon.xml</url>
  </menu_item>
</specials>
```

NOTE This XML sample code is different from the code used in "Anatomy of the Spry data set" on page 6.

The cafetownsend.xml file supplies the data for the master data set. The url node of the cafetownsend.xml file points to a unique XML file (or URL) for each menu item. These unique XML files contain a list of ingredients for the corresponding menu items. The summersalad.xml file, for example, might look as follows:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<item>
  <item_name>Summer salad</item_name>
  <ingredients>
    <ingredient>
      <name>butter lettuce</name>
    </ingredient>
    <ingredient>
      <name>Macintosh apples</name>
    </ingredient>
    <ingredient>
      <name>Blood oranges</name>
    </ingredient>
    <ingredient>
      <name>Gorgonzola cheese</name>
    </ingredient>
    <ingredient>
      <name>raspberries</name>
    </ingredient>
    <ingredient>
      <name>Extra virgin olive oil</name>
    </ingredient>
    <ingredient>
```

```
        <name>balsamic vinegar</name>
      </ingredient>
      <ingredient>
        <name>sugar</name>
      </ingredient>
      <ingredient>
        <name>salt</name>
      </ingredient>
      <ingredient>
        <name>pepper</name>
      </ingredient>
      <ingredient>
        <name>parsley</name>
      </ingredient>
      <ingredient>
        <name>basil</name>
      </ingredient>
  </ingredients>
</item>
```

Once you are familiar with the structure of your XML, you can create two data sets that you'll use to display data in master/detail dynamic regions. In the following example, a master dynamic region displays data from the data set dsSpecials, and a detail dynamic region displays data from the data set dsIngredients:

```
<head>
. . .
<script type="text/javascript" src="../includes/xpath.js"></script>
<script type="text/javascript" src="../includes/SpryData.js"></script>
<script type="text/javascript">
<!--Create two separate data sets-->
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
    var dsIngredients = new Spry.Data.XMLDataSet("data/{dsSpecials::url}",
  "item/ingredients/ingredient");
</script>
</head>
. . .
<body>
  <!--Create a master dynamic region-->
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table">
      <tr>
        <th>Item</th>
        <th>Description</th>
        <th>Price</th>
      </tr>
    <!--User clicks to reset the current row in the data set-->
    <tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
```

```
          <td>{item}</td>
          <td>{description}</td>
          <td>{price}</td>
        </tr>
      </table>
    </div>
    <!--Create the detail dynamic region-->
    <div id="Specials_Detail_DIV" spry:region="dsIngredients">
      <table id="Specials_Detail_Table">
      <tr>
        <th>Ingredients</th>
      </tr>
      <tr spry:repeat="dsIngredients">
        <td>{name}</td>
      </tr>
    </table>
</div>
. . .
</body>
```

In the example, the third `<script>` block contains the statement that creates two data sets, one called dsSpecials and one called dsIngredients:

```
var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
var dsIngredients = new Spry.Data.XMLDataSet("data/{dsSpecials::url}",
  "item/ingredients/ingredient");
```

The url for the second data set, dsIngredients, contains a data reference (`{dsSpecials::url}`) to the first data set, dsSpecials. More specifically, it contains a data reference to the url column in the dsSpecials data set. When the url or XPath argument in the constructor that creates a data set contains a reference to another data set, that data set (the data set being created) automatically becomes an observer of the data set it's referencing. As such, the new data set is dependent on the original data set, and reloads its data or reapplies its XPath whenever the data or current row changes in the original data set.

In the example, the act of changing the current row in the dsSpecials data set sends a notification to the dsIngredients data set that it also needs to change. Because each row of the dsSpecials data set contains a distinct URL in the url column, the dsIngredients data set must update to include the correct URL for the selected row.

By default, the dsIngredients data set (whose data is displayed in the detail region) is created using the data it obtains from the URL specified in the constructor—in this case a reference to the data in the url column of the dsSpecials data set. The default current row in the dsSpecials data set (the first row) contains a unique path to the summersalad.xml file, and thus the detail region displays the information from that file when the page loads in a browser. When the current row of the dsSpecials data set changes, however, the URL also changes—to salmon.xml for example—and the dsIngredients data set (and by association, the detail dynamic region) updates accordingly.

This process is functionally equivalent to the one illustrated in "Anatomy of the Spry basic master/detail dynamic region" on page 11, the technical difference being that in this case the second (or detail) data set is listening for data and row changes in the master data set, whereas in the basic example, the *detail region* is listening for data and row changes in the master data set.

It should also be noted that in the example code, `spry:region` is used for the detail region instead of `spry:detailregion`. As outlined in the previous section, the difference between a `spry:region` and a `spry:detailregion` is that the `spry:detailregion` specifically listens for CurrentRowChange notifications (in addition to DataChanged notifications) from the data set, and updates itself when it receives one. Because the current row of the dsIngredients data set never changes (it's the current row of the dsSpecials data set that changes), there is no need to use a `spry:detailregion` attribute. In this case, the `spry:region` attribute, which defines a region that only listens for DataChanged notifications, suffices.

Building Spry pages

This section contains the following topics:

- "To prepare your files" on page 18
- "To create a Spry data set" on page 19
- "To display data from a Spry data set" on page 21
- "Sample code: Spry data set and dynamic region" on page 23
- "To sort a Spry data set" on page 24
- "To create basic master/detail dynamic regions" on page 26
- "To create master/detail dynamic regions that rely on more than one data set" on page 28

To prepare your files

Before you begin creating Spry data sets, you'll need to obtain the necessary files (xpath.js and SpryData.js). The xpath.js file allows you to specify complex XPath expressions when creating your data set; the SpryData.js file contains the Spry data library.

You'll also need to link both files to whatever HTML page you're creating.

To obtain the necessary files

1. Locate the Spry_P1_1_06-08.zip file on the Labs website.
2. Download and unzip the Spry_P1_1_06-08.zip file to your hard drive.
3. Open the unzipped Spry_P1_1_06-08 folder and locate the includes folder. This folder contains the xpath.js and SpryData.js files necessary for running the Spry framework.
4. Copy the includes folder and either paste or drag a copy of it to the root directory of your web site.

   NOTE If you drag the original includes folder out of the unzipped Spry_P1_1_06-08 folder, the demos in the Spry_P1_1_06-08 folder won't work properly.

5. In Code view (View > Code) link the Spry data library files to your web page by inserting the following `script` tags within the page's `head` tag:

   ```
   <script type="text/javascript" src="includes/xpath.js"></script>
   <script type="text/javascript" src="includes/SpryData.js"></script>
   ```

   The SpryData.js file is dependent on the xpath.js file, so it's important that the xpath.js file come first in your code.

   Once you've linked the Spry data library, you are ready to create a Spry data set. For instructions, see "To create a Spry data set" on page 19.

6. Add the Spry name space declaration to the HTML tag so that the HTML tag looks as follows:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" xmlns:spry="http://
     ns.adobe.com/spry/">
   ```

   The Spry name space declaration is necessary if you want to validate your code.

NOTE Spry features will work locally as long as the Spry data library files are linked to your HTML page. When you want to publish the HTML page to a live server, however, you'll need to upload the xpath.js and SpryData.js files as dependent files.

Related Topics

"Anatomy of the Spry data set" on page 6

To create a Spry data set

1. Open a new or existing HTML page.
2. Make sure that you've linked the Spry data library files to the page. For more information, see "To prepare your files" on page 18.
3. Locate the XML source for the data set.

   For example, you might want to use an XML file called cafetownsend.xml located in a folder called data in the site's root folder:

   data/cafetownsend.xml

   You could also specify a URL to an XML file, as follows:

   http://www.somesite.com/somefolder/cafetownsend.xml

   NOTE The URL you decide to use (whether absolute or relative) is subject to the browser's security model, which means that you can only load data from an XML source that is on the same server domain as the HTML page you're linking from. You can get around this limitation by providing a cross-domain service script. For more information, consult your server administrator.

4. Because you'll need to specify the repeating XML node that supplies data to the data set, make sure you understand the structure of the XML before you create the data set.

   In the following example, the cafetownsend.xml file consists of a parent node called specials that contains a repeating child node called menu_item.

```
<?xml version="1.0" encoding="UTF-8"?>
<specials>
  <menu_item id="1">
    <item>Summer Salad</item>
    <description>organic butter lettuce with apples, blood oranges,
  gorgonzola, and raspberry vinaigrette.</description>
    <price>7</price>
  </menu_item>
  <menu_item id="2">
    <item>Thai Noodle Salad</item>
    <description>lightly sauteed in sesame oil with baby bok choi,
  portobello mushrooms, and scallions.</description>
    <price>8</price>
  </menu_item>
  <menu_item id="3">
    <item>Grilled Pacific Salmon</item>
    <description>served with new potatoes, diced beets, Italian parlsey,
  and lemon zest.</description>
    <price>16</price>
  </menu_item>
</specials>
```

5. Create the data set by inserting the following `script` block after the `script` tags importing the library:

```
<script type="text/javascript">
  var datasetName = new Spry.Data.XMLDataSet("XMLsource",
  "XPathToRepeatingChildNode");
</script>
```

In the Cafe Townsend example, you would create a data set with the following statement:

```
var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
```

The statement creates a new data set called dsSpecials that retrieves data from the specials/menu_item node in the specified XML file. The data set will have a row for each menu item and the following columns: @id, item, description, and price. The table can be visualized as follows:

@id	**item**	**description**	**price**
1	Summer salad	organic butter lettuce with apples, blood oranges, gorgonzola, and raspberry vinaigrette.	7
2	Thai Noodle Salad	lightly sauteed in sesame oil with baby bok choi, portobello mushrooms, and scallions.	8
3	Grilled Pacific Salmon	served with new potatoes, diced beets, Italian parlsey, and lemon zest.	16

You can also specify a URL as the source of the XML data, as follows:

```
var dsSpecials = new Spry.Data.XMLDataSet("http://www.somesite.com/
  somefolder/cafetownsend.xml", "specials/menu_item");
```

NOTE

The URL you decide to use (whether absolute or relative) is subject to the browser's security model, which means that you can only load data from an XML source that is on the same server domain as the HTML page you're linking from. You can get around this limitation by providing a cross-domain service script. For more information, consult your server administrator.

The completed example code could look as follows:

```
<head>
...
<script type="text/javascript" src="includes/xpath.js"></script>
<script type="text/javascript" src="includes/SpryData.js"></script>
<script type="text/javascript">
  var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
</script>
```

```
...
</head>
```

6. (Optional) If the values in your data set include numbers (as in this example) you'll want to reset the column types for the columns that contain those numerical values. This becomes important later if you want to sort data.

   You set column types by adding a data set method called `setColumnType` to the head tag of your document, after you've created the data set, as follows (in bold):

   ```
   <script type="text/javascript">
       var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
     "specials/menu_item");
       dsSpecials.setColumnType("price", "number");
   </script>
   ```

   In the example, the expression calls the `setColumnType` method on the data set object called dsSpecials, which you've already defined. The `setColumnType` method takes two parameters: the name of the data set column to re-type (`"price"`) and the desired data type (`"number"`).

   For more information, see "To sort a Spry data set" on page 24.

Once you've created the data set, your next step is to create a dynamic region so that you can display the data. For instructions, see "To display data from a Spry data set" on page 21.

Related topics

- "Sample code: Spry data set and dynamic region" on page 23

To display data from a Spry data set

Once you have created a Spry data set (see "To create a Spry data set" on page 19), you bind a Spry dynamic region to the data set. A Spry dynamic region is an area on the page that displays the data and automatically updates the data display whenever the data set is modified.

1. In Code view, create a Spry dynamic region by adding the `spry:region` attribute to the tag that will contain the region. The attribute uses the syntax `spry:region="`*`datasetName`*`"`.

   NOTE Most, but not all, HTML elements can act as containers for dynamic regions. For more information, see "Anatomy of the Spry dynamic region" on page 9.

   For example, if you are going to use a `div` tag as the container for the dynamic region displaying data from the dsSpecials data set, you would add the `spry:region` attribute to the tag as follows:

```
<div id="Specials_DIV" spry:region="dsSpecials">
</div>
```

NOTE: Dynamic regions can depend on more than one data set. You can add more data sets to the region by listing them as additional values of the `spry:region` attribute, separated by a space. For example, you could create a dynamic region using `spry:region="dsSpecials dsSpecials2 dsSpecials3"`.

2. Within the tag containing the dynamic region (this example uses a `div` tag), insert an HTML element to display the first row of the data set. You can use any HTML element to display data. One of the most typical elements used for this purpose, however, is a two-row HTML table, where the first row contains static column headings and the second row contains the data:

```
<table id="Specials_Table">
    <tr>
      <th>Item</th>
      <th>Description</th>
      <th>Price</th>
    </tr>
    <tr spry:repeat="dsSpecials">
      <td>{item}</td>
      <td>{description}</td>
      <td>{price}</td>
    </tr>
</table>
```

The values in curly braces in the second row specify columns in the data set. The curly braces bind the table cells to data in specific columns of the data set.

NOTE: If the Spry region depends on more than one data set, you must specify the data set to which you're binding the dynamic region. The full syntax takes the form of `{`*`datasetName`*`::`*`columnName`*`}`. For example, if you wanted to bind the dynamic region to two or three different data sets, you would need to enter the data in the above example as follows: `{dsSpecials::item}`, `{dsSpecials::description}`, and so forth.

3. Make the HTML element repeat automatically to display all the rows of the data set by adding the `spry:repeat` attribute and value to the HTML element tag in the form of:

```
spry:repeat="datasetName"
```

In the example, you would add the `spry:repeat` attribute to the table row tag as follows (in bold):

```
<tr spry:repeat="dsSpecials">
  <td>{item}</td>
  <td>{description}</td>
  <td>{price}</td>
</tr>
```

In the example, the completed code binding the dynamic region to the data set would look as follows:

```
<div id="Specials_DIV" spry:region="dsSpecials">
     <table id="Specials_Table">
     <tr>
       <th>Item</th>
       <th>Description</th>
       <th>Price</th>
     </tr>
     <tr spry:repeat="dsSpecials">
       <td>{item}</td>
       <td>{description}</td>
       <td>{price}</td>
     </tr>
   </table>
</div>
```

4. If you want, you can make the dynamic region more interactive by defining click events that allow users to sort data. For instructions, see "To sort a Spry data set" on page 24.

Related topics

- "Sample code: Spry data set and dynamic region" on page 23

Sample code: Spry data set and dynamic region

The following sample code creates a Spry data set and dynamic region to display a list of menu specials in an HTML table.

```
<head>
...
<script type="text/javascript" src="includes/xpath.js"></script>
<script type="text/javascript" src="includes/SpryData.js"></script>
<script type="text/javascript">
  var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
</script>
...
</head>

<body>
...
<div id="Specials_DIV" spry:region="dsSpecials">
  <table id="Specials_Table">
    <tr>
      <th>Item</th>
      <th>Description</th>
      <th>Price</th>
    </tr>
    <tr spry:repeat="dsSpecials">
      <td>{item}</td>
      <td>{description}</td>
      <td>{price}</td>
    </tr>
  </table>
</div>
...
</body>
```

Related topics

- "To create a Spry data set" on page 19
- "To display data from a Spry data set" on page 21

To sort a Spry data set

You can add click events to your dynamic region that allow users to interact with the data. You add click events by adding onclick handlers within the appropriate HTML tags. This section uses the table code from "To display data from a Spry data set" on page 21 as an example.

1. Locate the place in the code where you want to add the onclick handlers. In this example, we'll be adding the onclick handlers to two column headers in a table that displays the XML data.
2. Add an onclick handler with a sort method to the appropriate column header tags in the form of

   `onclick="`*`datasetName`*`.sort('`*`columnName`*`');"`

   The value defined in the sort method tells the data set which column to use when sorting the data.

   For example, adding the following onclick handlers (in bold) to column header tags causes the dynamic region to sort the data according to the specified value whenever the user clicks a column header on the page.

   ```
   <div id="Specials_DIV" spry:region="dsSpecials">
     <table id="Specials_Table" class="main">
       <tr>
         <th onclick="dsSpecials.sort('item');">Item</th>
         <th onclick="dsSpecials.sort('description');">Description</th>
         <th>Price</th>
       </tr>
       <tr spry:repeat="dsSpecials">
         <td>{item}</td>
         <td>{description}</td>
         <td>{price}</td>
       </tr>
   ```

```
    </table>
  </div>
```

Clicking “Item” on the page sorts the data alphabetically according to the menu item name, and clicking “Description” on the page sorts the data alphabetically according to the menu item’s description.

Numerical sorting

By default all data in the data set (including numbers) is considered text, and sorts alphabetically. To sort numerically (for example, to sort by price of menu item), you can use the data set method called `setColumnType` to change the data type of the price column from text to numbers. The method takes the form:

```
datasetName.setColumnType("columnName", "number");
```

Using the above example, you would add the `setColumnType` method to the head tag of your document, after you’ve created the data set (in bold):

```
<script type="text/javascript">
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
    dsSpecials.setColumnType("price", "number");
  </script>
```

The expression calls the `setColumnType` method on the data set object called `dsSpecials`, which you’ve already defined. The `setColumnType` method takes two parameters: the name of the data set column to re-type (`"price"`) and the desired data type (`"number"`).

You can now add the onclick handler to the price column so that all three columns in the HTML table are sortable when the user clicks any of the table headers:

```
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table" class="main">
      <tr>
        <th onclick="dsSpecials.sort('item');">Item</th>
        <th onclick="dsSpecials.sort('description');">Description</th>
        <th onclick="dsSpecials.sort('price');">Price</th>
      </tr>
      <tr spry:repeat="dsSpecials">
        <td>{item}</td>
        <td>{description}</td>
        <td>{price}</td>
      </tr>
    </table>
  </div>
```

To create basic master/detail dynamic regions

When working with Spry data sets, you can create master/detail dynamic regions to display more detail about your data. One region on the page (the master), drives the display of the data in another region on the page (the detail).

For an overview of how basic master/detail dynamic regions work, see “Anatomy of the Spry basic master/detail dynamic region” on page 11.

1. Create a data set. See “To create a Spry data set” on page 19.
2. Create the master region by adding the `spry:region` attribute to the HTML element that will act as the container tag for the region. See “To display data from a Spry data set” on page 21.

   In the following example, a master dynamic region displays repeated data from the data set dsSpecials:

```
<head>
. . .
<script type="text/javascript" src="../includes/xpath.js"></script>
<script type="text/javascript" src="../includes/SpryData.js"></script>
<script type="text/javascript">
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");</script>
</head>
. . .
<body>
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table">
      <tr>
        <th>Item</th>
        <th>Description</th>
        <th>Price</th>
      </tr>
      <tr spry:repeat="dsSpecials">
        <td>{item}</td>
        <td>{description}</td>
        <td>{price}</td>
      </tr>
    </table>
  </div>
</body>
```

3. Add event handlers that will allow users to change the current row in the data set. In the following example, an onclick event handler (in bold) changes the current row in the data set whenever a user clicks a row in the master region table:

```
<tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
  <td>{item}</td>
```

```
    <td>{description}</td>
    <td>{price}</td>
  </tr>
```

The {ds_RowID} data reference used in the example is a built-in part of the Spry framework that points to an automatically-generated unique ID for each row in the data set. (See "Anatomy of the Spry data set" on page 6.) When the user selects a row in the master region table, the onclick event handler supplies the unique ID to the setCurrentRow method, which causes the resetting of the current row in the data set.

4. Create the detail dynamic region on the page by adding the `spry:detailregion` attribute to the tag that will contain the region. The attribute uses the syntax `spry:detailregion="datasetName"`.

   In the following example, a `div` tag contains the detail dynamic region:

   ```
   <div id="Specials_Detail_DIV" spry:detailregion="dsSpecials">
   </div>
   ```

5. Within the tag containing the detail dynamic region, insert an HTML element to display the detail data from the current row of the data set.

   In the example, an HTML table displays detail data from the `{ingredients}` column and `{calories}` column in the dsSpecials data set.

   ```
   <div id="Specials_Detail_DIV" spry:detailregion="dsSpecials">
     <table id="Specials_Detail_Table">
       <tr>
         <th>Ingredients</th>
         <th>Calories</th>
       </tr>
       <tr>
         <td>{ingredients}</td>
         <td>{calories}</td>
       </tr>
     </table>
   </div>
   ```

The completed example code binding both the master and detail dynamic regions to the dsSpecials data set would look as follows:

```
<div id="Specials_DIV" spry:region="dsSpecials">
  <table id="Specials_Table">
    <tr>
      <th>Item</th>
      <th>Description</th>
      <th>Price</th>
    </tr>
    <tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
      <td>{item}</td>
      <td>{description}</td>
```

```
      <td>{price}</td>
    </tr>
  </table>
</div>
<div id="Specials_Detail_DIV" spry:detailregion="dsSpecials">
  <table id="Specials_Detail_Table">
    <tr>
      <th>Ingredients</th>
      <th>Calories</th>
    </tr>
    <tr>
      <td>{ingredients}</td>
      <td>{calories}</td>
    </tr>
  </table>
</div>
```

To create master/detail dynamic regions that rely on more than one data set

You can create master/detail relationships that involve more than one data set. For an overview of how such relationships work, see "Anatomy of master/detail dynamic regions that depend on more than one data set" on page 13.

1. Familiarize yourself with the structure of the XML files used in creating the data set. You'll need to understand the structure in order to make one data set depend on another.
2. Create a data set by adding the appropriate code to the head of your document. (See "To create a Spry data set" on page 19.) This will be the master data set.
3. Create a second data set (the detail data set) immediately following the master data set you just created. The URL or XPath in the constructor of the detail data set contains a data reference to one or more of the columns in the master data set. The data reference is in the form `{MasterDatasetName::columnName}`.

   In the following example, the third `<script>` block contains the statement that creates two data sets, one called dsSpecials (the master) and one called dsIngredients (the detail):

   ```
   <head>
   . . .
   <script type="text/javascript" src="../includes/xpath.js"></script>
   <script type="text/javascript" src="../includes/SpryData.js"></script>
   <script type="text/javascript">
       var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
     "specials/menu_item");
       var dsIngredients = new Spry.Data.XMLDataSet("data/
     {dsSpecials::url}", "item/ingredients/ingredient");
   ```

```
</script>
</head>
```

The path to the XML file for the detail data set, dsIngredients, contains a data reference (`{dsSpecials::url}`) to the master data set, dsSpecials. More specifically, it contains a data reference to the url column in the dsSpecials data set. When the url or XPath argument in the constructor that creates a data set contains a reference to another data set, that data set (the data set being created) automatically becomes an observer of the data set it's referencing.

4. Create the master region by adding the `spry:region` attribute to the HTML element that will act as the container tag for the region. See "To display data from a Spry data set" on page 21.

   In the following example, a master dynamic region displays repeated data from the data set dsSpecials:

```
<body>
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table">
      <tr>
        <th>Item</th>
        <th>Description</th>
        <th>Price</th>
      </tr>
      <tr spry:repeat="dsSpecials">
        <td>{item}</td>
        <td>{description}</td>
        <td>{price}</td>
      </tr>
    </table>
  </div>
</body>
```

5. Add event handlers that will allow users to change the current row in the master data set. In the following example, an onclick event handler (in bold) changes the current row in the dsSpecials data set whenever a user clicks a row in the master region table:

```
<tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
  <td>{item}</td>
  <td>{description}</td>
  <td>{price}</td>
</tr>
```

The `{ds_RowID}` data reference used in the example is a built-in part of the Spry framework that points to an automatically-generated unique ID for each row in the data set. (See "Anatomy of the Spry data set" on page 6.) When the user selects a row in the master region table, the onclick event handler supplies the unique ID to the `setCurrentRow` method, which causes the resetting of the current row in the data set.

6. Create the detail dynamic region on the page by adding the `spry:region` attribute to the tag that will contain the region. The attribute uses the syntax `spry:region="`*`datasetName`*`"`.

   NOTE When creating master/detail relationships using two or more data sets, it is not necessary to use the `spry:detailregion` attribute as is outlined in "To create basic master/detail dynamic regions" on page 26. Because the current row of the detail data set never changes (it's the current row of the master data set that changes), the `spry:region` attribute suffices. For more information, see "Anatomy of master/detail dynamic regions that depend on more than one data set" on page 13.

   In the following example, a `div` tag contains the detail dynamic region:

   ```
   <div id="Specials_Detail_DIV" spry:region="dsSpecials">
   </div>
   ```

7. Within the tag containing the detail dynamic region, insert an HTML element to display the detail data from the current row of the master data set.

   In the example, an HTML table displays detail data from the `{name}` column in the dsIngredients data set. The dsIngredients data set creates the `{name}` column based on the information it receives from the dsSpecials data set.

   ```
   <div id="Specials_Detail_DIV" spry:region="dsIngredients">
     <table id="Specials_Detail_Table">
       <tr>
         <th>Ingredients</th>
       </tr>
       <tr spry:repeat="dsIngredients">
         <td>{name}</td>
       </tr>
     </table>
   </div>
   ```

The completed example code binding the master region to the dsSpecials data set, and detail region to the dsIngredients data set, would look as follows:

```
<head>
. . .
<script type="text/javascript" src="../includes/xpath.js"></script>
<script type="text/javascript" src="../includes/SpryData.js"></script>
<script type="text/javascript">
    var dsSpecials = new Spry.Data.XMLDataSet("data/cafetownsend.xml",
  "specials/menu_item");
```

```
    var dsIngredients = new Spry.Data.XMLDataSet("data/{dsSpecials::url}",
  "item/ingredients/ingredient");
</script>
</head>
. . .
<body>
  <div id="Specials_DIV" spry:region="dsSpecials">
    <table id="Specials_Table">
      <tr>
        <th>Item</th>
        <th>Description</th>
        <th>Price</th>
      </tr>
        <tr spry:repeat="dsSpecials"
  onclick="dsSpecials.setCurrentRow('{ds_RowID}')">
        <td>{item}</td>
        <td>{description}</td>
        <td>{price}</td>
      </tr>
    </table>
  </div>
  <div id="Specials_Detail_DIV" spry:region="dsIngredients">
    <table id="Specials_Detail_Table">
      <tr>
        <th>Ingredients</th>
      </tr>
      <tr spry:repeat=”dsIngredients”>
        <td>{name}</td>
      </tr>
    </table>
  </div>
. . .
</body>
```