Using "roles" and "writeRoles" Attributes with Fields and Actions

Data Aquarium Framework allows controlling user access to the fields and actions defined by data controller based on user roles. The standard ASP.NET security infrastructure is being utilized to determine the role of the current user. You must enable any form of authentication supported by ASP.NET to be able to use this features.

Element “field” in the “/dataController/fields” section has attributes “roles” and “writeRoles”. The first attribute specifies a space- or comma-separated list of roles allowed to read the field. If “roles” element is not present or defined as a blank string then any user is allowed to see the field presented in the views. If “roles” attribute is not blank then a call

```
HttpContext.Current.User.IsInRole(role)
```

is executed by the framework to see if the field should be visible in the presentation views defined in the data controller XML file.

Attribute “writeRoles” specifies the roles that user must have in order to be able to change the field content.

Element “action” of any “/dataController/actions/actionGroup” section allows specifying user roles that are allowed to execute that action. If the attribute is not present or blank then the action is available to everyone. If a space- or comma-separated list is present then the framework will ask ASP.NET to see if the user does have one of this roles before this action is allowed to be displayed and executed in the views defined in the data controller file.

Notice that the same action may be defined in multiple actions group. For example, the standard generated definition of the data controller will have “Delete” action defined twice in the action group with scope “Grid” and twice in the action group with scope “Form”. You have to make sure that the same list of roles allowed to execute the “Delete” action is duplicated in all four instances.

Let's try the real example. This example assumes that you do have *SQL Server 2005 Express Edition* installed on your development machine.

Generate a *Data Aquarium* project for the *Northwind* database. Open the web site with *Visual Studio 2008* or *Visual Web Developer Express 2008* and add a form named *Login.aspx* to the root of the new web site. Switch this form to design mode and drop *Login* control on the form. Save the form. Here is the snippet of your from text with the *Login* control in it.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Login ID=Login1 runat=server></asp:Login>
    </div>
    </form>
</body>
</html>
```

Select "*Website ASP.NET Configuration*" menu option in *Visual Studio* and enable security for our sample application. Indicate that your users are visiting your site from Internet. Specify that you want to enable roles. Create two roles named *Admin* and *User*. Create two user accounts, one for each role, and make sure to memorize their user names and password. Create an access rule that denies anonymous user access to the root of your web site. The configuration tool will automatically insert the needed settings in to the *web.config* file of the sample application will create *App_Data* project folder with *ASP.NET Membership* database in it.

Start *default.aspx* page. You will be asked to sign in. Make sure that you can sign in with one of the user accounts that you have created. When sign in successfully the default page will display the views defined in the **~/Controllers/Employees.xml** data

controller configuration file. Close the browser and open that data controller file in *Visual Studio*.

Make the following changed in the definition of *LastName*, *FirstName*, and *Title* fields.

```
    <field name="LastName" type="String" allowNulls="false" label="Last Name"
writeRoles="Admin"/>
    <field name="FirstName" type="String" allowNulls="false" label="First
Name" writeRoles="Admin"/>
    <field name="Title" type="String" label="Title"  roles="Admin"/>
```

Run the application a few times while singing in with different user accounts that you have created. Notice that when you sign with account that has *User* role then *Title* field is not visible in any of the presentation views. User with *Admin* can see the *Title* field at all times.

Fields *LastName* and *FirstName* are accessible to the user with role *User* but this field is always read-only whenever you switch to any of the edit or insert views.

Now change the action groups to have roles attribute for command Select, Edit, and Delete as shown below.

```
    <actionGroup scope="Grid">
      <action commandName="Select" commandArgument="editForm1"
roles="Admin,User" />
      <action commandName="Edit"  roles="Admin"/>
      <action commandName="Delete" confirmation="Delete?" roles="Admin"/>
      <action whenLastCommandName="Edit" commandName="Update"
headerText="Save" />
      <action whenLastCommandName="Edit" commandName="Cancel" />
    </actionGroup>
    <actionGroup scope="Form">
      <action commandName="Edit" />
      <action commandName="Delete" confirmation="Delete?"  roles="Admin"/>
      <action commandName="Cancel" headerText="Close" />
```

```
      <action whenLastCommandName="Edit" commandName="Update" headerText="OK"
/>
      <action whenLastCommandName="Edit" commandName="Delete"
confirmation="Delete?" roles="Admin"/>
      <action whenLastCommandName="Edit" commandName="Cancel" />
      <action whenLastCommandName="New" commandName="Insert" headerText="OK"
/>
      <action whenLastCommandName="New" commandName="Cancel" />
    </actionGroup>
```

User with administrative accounts will see no difference in the presentation views. User with role *User* will only *Select* option in the context menu of the data row in the grid view of employees. Delete action will not be allowed to that user in the form views as well.

As you can see there are plenty of declarative options to control security in the AJAX-enabled user interface without actually writing a single line of code. Any web form that is using your data controller will automatically inherit all security settings, which makes application maintenance a snap.