

Walkthrough: Databinding with IronPython for ASP.NET

Introduction

IronPython code is dynamically compiled, which allows simplified data binding syntax and increased flexibility. This walkthrough shows you how to add IronPython code to tailor the appearance of data retrieved from a Microsoft Access database (.mdb file).

Note Access databases do not have the same capacity and are not as scalable as other types of databases, such as Microsoft SQL Server. This walkthrough uses an Access database because the Northwind.mdb sample database is widely known and available. Generally, if you are creating a Web site that will support only light traffic or a limited number of users, you should use a SQL Server Express database. However, if the Web site will support more throughput or a larger number of users, you should consider using SQL Server or another database that is suited for production Web sites.

Prerequisites

In order to complete this walkthrough, you will need the following:

- Microsoft Visual Studio 2005 or Microsoft Visual Web Developer
- IronPython for ASP.NET Community Technology Preview, including Visual Studio Integration or Visual Web Developer Integration, as appropriate
- The Northwind.mdb file that contains the Access version of the sample Northwind database. Alternatively, you can use another database and adjust the steps in the walkthrough to match the database and tables that you are using.

This walkthrough assumes that you have a general understanding of working with ASP.NET in Visual Studio, and a general knowledge of data binding in ASP.NET. For an introduction to ASP.NET in Visual Studio, see [Walkthrough: Creating a Basic Page in Visual Web Developer](#). For an introduction to data binding in ASP.NET, see "Data-Driven Web Pages".

Creating the Web Site and Page

If you have already created a Web site in Visual Studio (for example, by working with the companion walkthrough "Creating a Basic Web Page with IronPython"), you can use that Web site and go to the next part of the walkthrough, where you add an Access database to the project. Otherwise, create a new Web site and page by following these steps.

To create a file system Web site

1. Open Visual Studio or Visual Web Developer.
2. On the **File** menu, click **New Web Site**.
The **New Web Site** dialog box appears.
3. In the **Language** list, click **IronPython** to make IronPython the default language for the Web site.
Note You can use statically compiled languages in the same Web application by creating pages and components in different programming languages.
4. Under **Visual Studio installed templates**, click **ASP.NET Web Site**.
5. In the **Location** box, select the **File System** box, and then enter the name of the folder where you want to keep the pages of your Web site.
For example, type the folder name **C:\DatabaseWebSite**.
6. Click **OK**. Visual Studio creates the folder and a new page named Default.aspx.

Next you will add the Northwind.mdb Access database or your own database to the Web application project.

To add an Access database to the project

1. If the App_Data folder does not exist in Solution Explorer, right-click the name of your Web site, click **Add ASP.NET Folder**, and then click **App_Data**.
2. In Visual Studio, in **Solution Explorer**, right-click the App_Data folder, and then click **Add Existing Item**.
3. Use the **Add Existing Item** dialog to locate the Northwind.mdb file (or other .mdb or .mdf file) that you want to use for this walkthrough, and then click **Add** to add it to your application.

This walkthrough does not configure permissions for the Access database. For information on configuring permissions for an Access database, see the MSDN Online Library topic "Walkthrough: Creating a Web Page to Display Access Database Data".

Using Access Data on an ASP.NET Web Page

You can now use the database in a Web page. This part of the walkthrough uses an **AccessDataSource** control and a **DataList** control.

To add AccessDataSource and DataList controls to the page

1. Open the Default.aspx page (or another page that you want to use) and switch to Design view.
2. From the **Data** group in the Toolbox, drag an **AccessDataSource** control onto the page.
Note If the **Access Data Source Tasks** shortcut menu does not appear, right-click the control and then click **Show Smart Tag**.
3. On the **Access Data Source Tasks** shortcut menu, click **Configure Data Source**.
The Configure Data Source wizard appears.
4. On the **Choose a database** page, in the **Microsoft Access Data file** box, type `~/App_Data/Northwind.mdb` or use the **Browse** button to select the .mdb file.
5. Click **Next** to open the **Configure Select Statement** page, and click **Specify columns from a table or view**.
6. In the Name list, click **Categories**.
7. Select the **CategoryName** and **Description** check boxes and click **Next**.
8. Optionally, click **Test Query** to test your query.
9. Click **Finish**.
10. From the **Data** group in the **Toolbox**, drag a **DataList** control onto the page.
11. On the **DataList Tasks** menu, in the **Choose Data Source** box, click **AccessDataSource1**.
12. Click Ctrl+F5 to run the page with the default layout.
13. Close the browser.

Working with IronPython Code in the DataList Control

In this part of the walkthrough you will use IronPython code to perform data binding, rather than using the **Eval** method that is generated by default for declarative data binding.

To use IronPython code in the DataList

1. Switch to Source view, and remove the **Eval** methods from `DataList1`. When you are finished, the **DataList** control markup will look like the following:

```
<asp:DataList ID="DataList1" runat="server"
    DataSourceID="AccessDataSource1">
    <ItemTemplate>
        CategoryName:
        <asp:Label ID="CategoryNameLabel" runat="server"
            Text='<# CategoryName %>'>
        </asp:Label><br />
        Description:
        <asp:Label ID="DescriptionLabel" runat="server"
            Text='<# Description %>'>
        </asp:Label><br />
    <br />
    </ItemTemplate>
</asp:DataList>
```

The data bindings are now simply IronPython code. Because IronPython is a dynamic language, the field names can be resolved using late binding.

2. Click Ctrl+F5 to run the page, and verify that the page looks the same.
3. Close the browser.

Because the bindings are IronPython code, you can use them to change the way the field values are displayed. In this part of the walkthrough, you will change the case of the `CategoryName` field, and change the background color of the `Description` field depending on the length of the `CategoryName` field.

To use IronPython code to change the appearance of fields

1. Switch to the code for the page, and add the following import statement:

```
from System.Drawing import Color
```

2. Add the following function to return a color based on the size of a string:

```
def ColorPicker(input):  
    input = str(input)  
    if len(input) > 10:  
        return Color.Yellow  
    else:  
        return Color.White
```

3. Return to the Web page markup, replace the text "CategoryName:" with the text "The < %# CategoryName.upper() %> category includes:", and remove the `CategoryNameLabel` label from the item template.

When you are finished, the item template will look like the following.

```
<ItemTemplate>  
    The < %# CategoryName.upper() %> category includes:  
    Description: <asp:Label ID="DescriptionLabel" runat="server"  
        Text='< %# Description %>'>  
    </asp:Label><br /><br />  
</ItemTemplate>
```

IronPython will resolve the field name and apply the **upper()** method to the field. Be sure to include the parentheses; if you omit them, IronPython returns an object representing the method itself, which will not display anything.

Note You can use either the Python **upper** method or the .NET Framework **ToUpper** method.

4. Remove the text "Description:" above the `DescriptionLabel` label, and add a **BackColor** attribute to the `DescriptionLabel` label.

When you are finished, the item template will look like the following.

```
<ItemTemplate>  
    The < %# CategoryName.upper() %> category includes:  
    <asp:Label ID="DescriptionLabel" runat="server"  
        Text='< %# Description %>'>  
        BackColor='< %# ColorPicker(CategoryName) %>' >  
    </asp:Label><br /><br />  
</ItemTemplate>
```

IronPython will evaluate the expression and call the `ColorPicker` method, changing the background color of descriptions for category names longer than ten characters.

5. Press Ctrl+F5 to run the page and verify that the category name appears in upper case, and that the description has a yellow background color whenever the category name is longer than ten characters.

Next Steps

This walkthrough describes the basics of using IronPython code in data binding. You can use IronPython with all types of data binding in ASP.NET. You can explore other ASP.NET data features in the MSDN Online Library topic "Data-Driven Web Pages".

See Also

[Walkthrough: Creating a Basic Page in Visual Web Developer](#)