

Handling Parent-Child Tables

By: Roddy P. Carbonell

URL: <http://www.cttemplates.com>

E-Mail: rpcarnell@hotmail.com

Sometimes you will encounter the need to output the contents of two tables in a sequential manner (usually after one row from the main table is outputted, at least one row from the child table will follow, and so on). In ASP.NET, doing this is easier than you think. For this tutorial, we'll choose two tables: groups and forums, with **groups** being the **parent table**, and **forums** being the **child table**. The structure of these tables is as follows:

For table groups:

```
groid int 4
gname varchar 50
grdesc varchar 500
```

And for table forums:

```
fid int 4
groid int 4
formme varchar 50
fordesc varchar 500
```

For forums and groups to be related, you need at least one variable to connect them both. The integer called **groid** is that variable. The output will be something like this:

Science

Everything you want to know about science is here. Biology Want to study life? Start here Physics Chemistry We know more than the periodic table

Engineering

Design, science, technology. We have it all Electrical We have all the circuits you need Mechanical Yes. We know more than NASA does. Industrial It is almost business in disguise, we still like it.

Complains

You have something to say about our forum, or its users? This is the place.

The words Science, Engineering and Complains belong to table groups, and the rest belongs to forums. Now let's examine the C-Sharp script used to develop the relationship between tables:

```

<%@ Page Language="C#" debug="TRUE" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<script runat=server>

void Page_Load(Object sender , EventArgs e)
{
    DataSet dasetForCateg;
    SqlConnection conForCateg;
    SqlDataAdapter AdapForCateg;

    // Grab the Categories and Products table
    dasetForCateg = new DataSet();
    conForCateg = new SqlConnection( "DATABASE=name-of-your-
database;SERVER=name-of-your-server;UID=your-id;PWD=your-password" );
    AdapForCateg = new SqlDataAdapter( "Select * From groups", conForCateg );
    conForCateg.Open();
    AdapForCateg.Fill( dasetForCateg, "groups1" );

```

If you are familiar with ADO.NET, or if you have read the previous tutorials, the script as it is so far will seem easy to you. Here we have **conForCateg** as a **SqlConnection** to the database. **AdapForCateg** is a **SqlDataAdapter** that is taking all the values available in a table called groups. **conForCateg** is opened, and then the values taken by **AdapForCateg** are filled into the **DataSet** (called **dasetForCateg**) and stored in a table called groups1.

```

    AdapForCateg.SelectCommand = new SqlCommand( "Select * From forums",
conForCateg );
    AdapForCateg.Fill( dasetForCateg, "forums1" );
    conForCateg.Close();

```

Now it gets a little more complicated. We return to the **SqlDataAdapter** (**AdapForCateg**), and we employ the property **SqlCommand** to select all the values from a second table: forums. Those values are also sent to the **DataSet** called **dasetForCateg**, but they are stored in a table called forums1.

```

    // Add Parent/Child Relationship
    dasetForCateg.Relations.Add( "gr_for_out",
dasetForCateg.Tables["groups1"].Columns["groid"],
dasetForCateg.Tables["forums1"].Columns["groid"] );

```

This line creates a Parent-Child relationship between groups1 and forums1. The **DataRelationCollection Class** is used to handle data relations inside a dataset. This class is not declared directly but by using the **ParentRelations** property of a **DataTable**, or the **Relations** property of a **DataSet**, which is what we are doing in this script. Our **Relations** property is creating a connection between two tables, which allows us to treat **dasetForCateg.Relations** altogether as part of **DataRelationCollection**. This class has **Add** as one of its methods, and **go_for_out** is the

name of the connection between the DataSet tables groups1 and forums1. Notice that both Columns have groid as an ID, the same ID both tables have in common. And notice that forums1 is the second table, and this makes it the child table.

```
// Display each Category and Child Products
foreach (DataRow drowParent in dasetForCateg.Tables["groups1"].Rows)
{
    lblOutput.Text += "\n<tr>\n<td class='group' colspan='2'>\n<h2>" +
drowParent["grname"] + "</h2>\n" + drowParent["grdesc"] + "</td>\n</tr>\n";
    foreach (DataRow drowChild in drowParent.GetChildRows( "gr_for_out" ))
    {
        lblOutput.Text += "\n<tr>\n<td class='forum'><b>\n" + drowChild["forname"]
+ "</b>\n</td>\n<td>\n" + drowChild["fordesc"] + "</td>\n</tr>\n";
    }
}
</Script>
```

The **Tables** property of the **DataSet** is followed by another property: **Rows**, which belongs to the **DataTable** class. The Tables property can get all the tables inside the DataSet, and the Rows property can get all the rows inside the tables. Our first foreach loop seizes the parent values and stores them in **lblOutput** as one big string.

Notice that we have a second foreach. This one will fetch the childrows contained by the previous DataRow by using the **GetChildRows** method, where **gr_for_out** is the name we gave to the data-relation.

What follows is just basic HTML-ASP.NET. At this point, you should have a good idea of how to create a Parent-Child relationship between two different data-tables.

```
<html>
<head><title>DataRelation.aspx</title>
<style type="text/css" >
h2
{
    font-size: 20px;
    font-weight: bold;
    padding: 2px;
}
.group
{
    background: #def;
    font-size: 15px;

    padding: 10px;
}
.forum
{
    background: #fff;
```

```
padding: 8px;
}
</style>
</head>
<body>
<table>
<asp:Label
  ID="lblOutput"
  Runat="Server" />
</table>
</body>
</html>
```