```
btnSave_Cick()
{
    Response.Redirect("~/MyPage.aspx");
}
```

Here, on the postback (button click), we are redirecting the user to another resource (mypage.aspx), and this approach goes against the REST principle as we are delegating the responsibility to load a resource to another page based controller's postback event. This is not REST-like behavior. Now, we will see how MVC compliments the REST approach.

MVC and REST

MVC is radically different from the default page controller based design in the ASP.NET framework as it implements a front controller based design. In our normal applications, we use a lot of postbacks and make use of ViewState, and the development is centered around web forms. For each functional aspect, we may have a single webform; for example, for adding customers, we might create something like AddCustomer.aspx, and for showing a list of customers, we might use CustomerList.aspx.

But in an MVC architecture, webforms lose their importance. We don't create webforms in the same way that we do in standard ASP.NET applications. In the MVC framework, we use URL routing, which means that all URLs have some specific format, and the URLs are used based on the settings in a config file. In a standard ASP.NET application, the URL is linked to a specific ASPX file, say http://localhost/CustomerList.aspx. In MVC, the URL routes are defined in a REST-like fashion: http://localhost/customer/list/.

So in MVC, ASPX pages are reduced to simply showing the view; they will not have any code in their code-behind classes. What needs to be shown on an ASPX page will be handled by the Controller classes. ASPX will just be a kind of view engine and nothing else. ASPX will not have control-level event handlers or any kind of logic in the code-behind. In the next section, we will see how the ASP.NET MVC framework makes our life easier in adopting an MVC based approach in our projects.

ASP.NET MVC Framework

The ASP.NET MVC framework was released by Microsoft as an alternative approach to web forms when creating ASP.NET based web applications. The ASP.NET MVC framework is not a replacement or upgrade of web forms, but merely another way of programming your web applications so that we can get the benefits of an MVC design with much less effort.

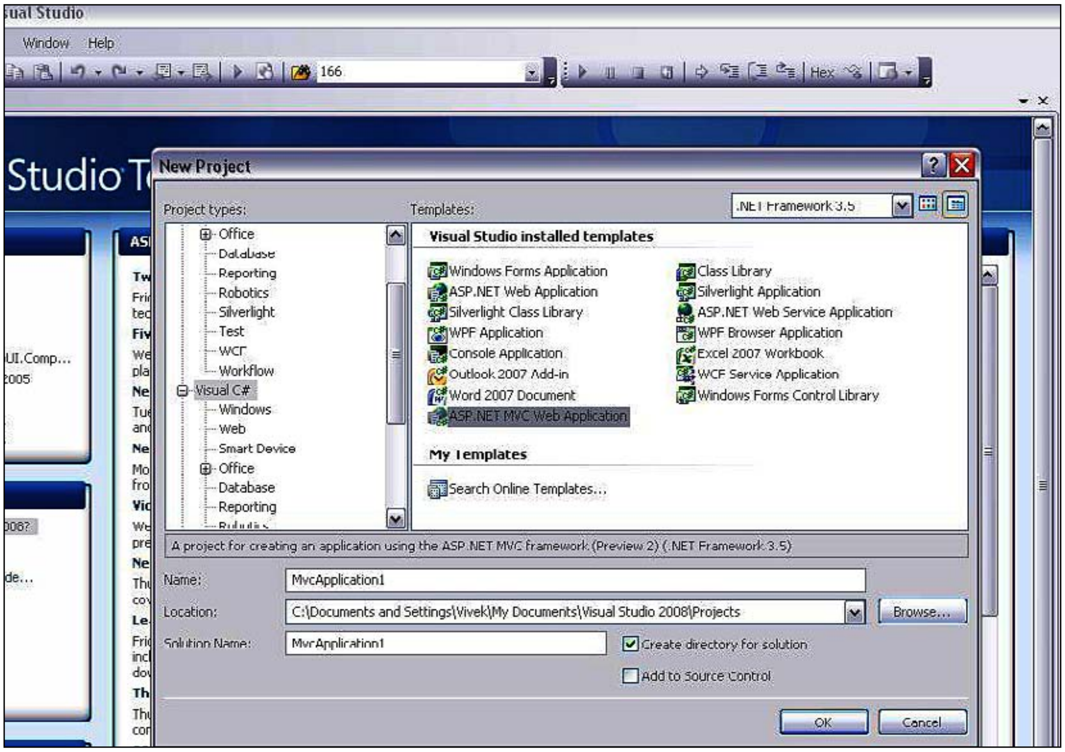
As of now, the ASP.NET MVC framework is still in CTP (Community Technology Preview, which is similar to an advanced pre-stage), and there is no certain date when it will be released. But even with the CTP 5, we can see how it will help MVC applications follow a stricter architecture.

We will quickly see how to use the ASP.NET MVC framework through a small example.

Sample Project

First, download the ASP.NET MVC framework from the Microsoft website and install it. This installation will create an MVC project template in VS 2008.

Start VS 2008, select the **File | New Project** menu item and then choose the **ASP.NET MVC Web Application** template to create a new web application using this template.



A project for creating an application using the ASP.NET MVC framework (Preview 2) (.NET Framework 3.5)

Name: MvcApplication1

Location: C:\Documents and Settings\Vivek\My Documents\Visual Studio 2008\Projects

Solution Name: MvcApplication1

☑ Create directory for solution

□ Add to Source Control