Introduction to ASP.NET

Introduction

Hi. This is Dan Wahlin with Pluralsight. I'd like to welcome you to the ASP. NET Web Form Development Course. Throughout this course, I'll introduce you to several key technologies you need to know as a. NET Developer to build successful Web Applications. We'll, start off by introducing the. NET Framework and different tools you can use to create ASP. NET Projects. From there, we'll move into ASP. NET server controls, data access techniques, including Object Relational Modeling tools or ORMs. And, we'll even talk about AJAX, enabling your Web Applications. From there, at the end of the course, we'll talk about, what does it take to deploy an application? There's several different techniques that'll be covered there. Now, in this first module, I'm going to introduce you to what is the. NET Framework. For those that, haven't dealt with it very much before. I'll also talk about a really nice free tool from Microsoft called the Web Platform Installer. And we're going to use that to install, Visual Studio Web Express. Once we do that, I'll introduce you to what Visual Studio has to offer as far as project management. And then we'll jump in to, how do you create ASP. NET Projects, and what are the different types of projects that are available. Now, as you pick different projects, there's different compilation models that are use. So, we'll briefly touch on that, and demonstrate how they work in each type of project. Finally, at the end of this module, I'll build a simple Web Forms Applications for you. And we're going to use some fundamental core server controls. We'll build this application, add a little bit of code, and help you jump start the process of building your applications. From there, as mentioned, we'll move on to more sophisticated techniques that allow you to build dynamic, Web Applications, that can integrate data from different types of Data Sources. So let's go ahead and get started.

Introduction to the .NET Framework

Before we dive into, ASP. NET Web Forms and the different tools that are available for building Web Forms Applications. Let's take a brief look at what the. NET Framework has to offer us, as developers. Now, if you're already working with the. NET Framework and familiar with it, then you can certainly move on to the next section on the Web Platform Installer. But if you're new to the. NET Framework, I'll provide an overview of what it is and what it offers us, as developers. So to get started, designer framework really is just a managed computing platform or environment. It provides a way to write code and execute that code. Now, that statement alone makes it sound pretty simplistic. If I take it up a notch, really what the. NET Framework is, is a library of software. Along with a secure runtime environment that also handles memory management for us. And that's the term managed. It handles our memory for us and manages us. Now the. NET framework uses something called the Common Language Runtime or CLR to actually, execute code, and this also applies to ASP. NET Web Forms. So, if you write something in say, C#, which is a. NET Framework language, then you'll need to have the. NET Framework itself installed. On your Web Server if you want to run that code in the, within the ASP. NET Runtime Framework. Now, there's a lot of different languages available. C# is just one of them, Visual Basic's also very popular, and there's many others. In fact, there's over 25 different languages that are actually, out there in the wild that you can use. As you're using these different languages, you can build a lot of different types of applications. So, you can build of course ASP. NET Web Forms, or desktop applications, or even distributed applications, where we might integrate, or build Web Services. Now, regardless of the type of application you build, or the language. Everything in. NET is Object-Oriented. So, we'll be talking throughout this video series about classes and we'll apply Object-Oriented principles as we're doing it. Now, if we break down the. NET Framework into a little more fine-grain detail, you'll see at the bottom of this diagram we have the Common Language Runtime, I mentioned earlier. This is what has to be there to run your ASP. NET Web Forms applications. You have to install this on your web server. Now, the Common Language Runtime, provides this secure environment I talked about, provides a way to execute and run code. Manages memory and it also allows languages to interoperate with each other. And each language is really treated as a first class citizen. Whether you're doing C# or VB or F# or J#. Now, all languages share a Common Type System, across them. So, we don't have rounding issues for instance with decimals. You might have a team within your company that's doing Visual Basic, and another team doing C#. And those applications might integrate and exchange messages. Well, because every. NET Language uses the same type system. We don't run into, rounding issues and other issues that typically come up, when you exchange data between applications. So, it's very smooth regardless of the language that you're using. NET. Now, the. NET Framework, provides a library of software. And the core of that is Foundation Class library FCl or you might also hear Base Class library or BCl. This is the core set of classes that allow you to do things such as. Read and write the files or streams, integrate with different security mechanisms, perform cryptography, do globalization of your application for different language support, and much, much more. Now, sitting on top of that, we have a very robust, Data Access Framework. So, we can integrate with everything from Oracle to SQL to lots of other databases out there, that are ODBC, OLE DB or other driver type of compliant. We can also, work with XML and parse it using fast forward only readers or we can load it up into memory as well, and even edit it. Now, as mentioned you can build ASP. NET Web Forms, Desktop, Windows Phone 7, Web Services and many types of applications using one of the languages listed above and there's many more of them mess. But these are some of the popular ones available, with C# and VB, being the most popular. All languages are based upon a common language specification. So, when it comes to ASP. NET Web forms, you can pick one of these languages and rest assured that even if you use a component from another language that's a. NET Compatible Language. They'll still inter operate just fine. And again, you won't have these weird issues in talking between languages. And that's because they're all based on these Common Language Specifications or CLS. Now, the thing that really makes. NET, super compelling. At least for me. And over the years of using it, I think this adds greatly to your productivity as a developer. And that thing is Visual Studio. The Integrated Development Environment, definitely my preference for editing code. Now you can certainly use, other editors out there, whether it's VI or something simple like Notepad to build your. NET Applications. The Visual Studio, provides not only a simple way to write code, but a visual way to design Interfaces. You get excellent code support as you're typing what we call Intelesense and much, much more. I can do a lot with this Interface without leaving. Living at. And it's the glue that ties everything together. So, in the next section of this module we're going to install a version of Visual Studio using something called the Web Platform Installer. So let's go ahead and jump in to that next part of this module.

Web Platform Installer

Microsoft's Web Platform Installer provides a simple way to get started building, ASP. NET Web Applications. It makes it really easy to download the necessary tools that you need to help your productivity. So, in this section we're going to talk about the Web Platform Installer. And then we're going to get Visual Studio 2010 Web Express installed. The Web Platform Installer can be downloaded by simply going to your favorite search engine and typing, Microsoft Web Platform Installer, in the search prompt. And once you find that you'll go to this page here. Now, to download it you can simply hit the green Download It Now button. And it's only about two megabytes, so it's very fast, only takes a matter of seconds to get this installed, and up and running. The benefits of this tool are that, now that you don't have to go out to Microsoft. com or www. ASP. NET or even other properties to download the tools, that you need to get going. Plus, this tool automatically links in dependencies. So for instance, to use Visual Studio 2010. You need the. NET Framework 4. 0. Where normally, you'd have to get the. NET Framework installed and do all that type of stuff. Well, with the Web Platform Installer, all the dependencies that you would need to get going quickly and easily are taken care of for you. The other thing is does is, if you have any open source projects that are out there that you want to link in. For instance, there might be a content management system that you'd like to use that's based on ASP. NET Web Forms. Well, the Web Platform Installer makes it a piece a cake to find all the popular open source projects that are out there. And get those installed and running so let me go ahead and pull up a search engine and we'll go ahead and search for the Web Platform Installer and get it going. I'm now at bing. com and we're going to go ahead and search for the Web Platform Installer and download that. So, I've already done the search to save a little typing. But this is what you'd search for, Microsoft Web Platform Installer. You can see it shows right at the top of the search results. So, we go ahead and click on that. And now we're back to the screen that I showed just a little bit earlier. Now from here, all we need to do is click on download it now, but before doing that, let me show you what's available, so if we go to the Windows Web Application gallery. This is some of the more popular Web Applications that are features. So,. NetNuke, we can scroll down and there's Joomla and several others. We can also come over and get the latest editions that are available for download. So, you can see several in here. Get the highest rated applications. And all this makes it very easy to go to one place. Well, we could do it from here but the Web Platform Installer makes it even easier. So let me go ahead and go back. And we're going to go ahead and download and install it. All right. So, this'll start the installation process. Now, as it mentions right below this dialogue, it's only about two megabytes, so it's a very fast install process. And there's not a lot of work involved on our part. In fact, we can kind of just kick back. And it's going to do its thing right here as it loads the latest Web Platform products. And there we have it. So, that is a live example of how fast it is to get it going. So it's very simple. From here, you could see we're on the Spotlight screen. So, some of the more popular downloads are available. And we're going to be downloading the Visual Web Developer Express 2010 SP1 and that's a free version of Visual Studio we can use to build ASP. NET Web Applications. But before we do that, let me show you some of the other things available. So, we have several open source projects you can see here Content Management Systems and others. But, we could also go to products and download several different things here let me scroll down a little bit and let's say you were doing Smooth Streaming Media. And you want to work with IIS Media Services 4. Well, you can download that very easily directly right here. And there's many, many different tools that are available, we can work with Windows Azure. Which is the Cloud Service Microsoft provides. We can go to different servers that are available. Get things like IIS 7. 5 Express. Go to databases and even get some of the tools, such as Microsoft's new WebMatrix tool. Well, we're going to go back. I'm just going to go to Spotlight and we're going to do, the Web Developer Express. Now, if that didn't show up for some reason as you tried it, feel free to go up to the search box here. And we can actually, just type express and this will go out and filter all the different things that have that word in it and you can see we can get to. Web Developer Express here as well. But all we have to do is, once we find it, I can click the Add button, and that's it, from here we can hit install. Now let me show you what's going to be done for us behind the scenes though, there's quite a bit. So I'm going to click on Items To Be Installed. Now, in order to run Visual Web Developer Express 2010 with Service Pack 1. There are several dependencies we have, including the tool, but we also have to install the. NET Framework, if we're going to work with databases locally. We can work with SQL Server Express 2008 R2. And you'll see several others down here. Even the latest version, of a file based database is going to be installed for us. Which is the new SQL Server Compact 4. Makes it really easy to work locally with databases. So, a lot of great stuff. You can see the total download size. And everything is there. Now, from here, all we need to do is hit install. It's a piece of cake. going to have to accept the agreement. And it's going to take us to a screen to configure our SQL Server Express Database. Now, it takes us to mixed mode authentication. If you want to do a SQL Server Password, you can enter that here. I'm going to go ahead and do Windows integrated authentication to just use my Windows credentials. To ultimately Log in once it installs this. Now, from here it's going to go ahead and download all those dependencies that you saw and so this will take a little bit and once those are downloaded it'll install. So, I'm going to go ahead and let it finish that and then we'll come back and we will then jump into the tools that installed. Once the Microsoft Web Platform Installer finishes the installation process, you'll see that Visual Web Developer Express is now showing as installed. Now from here, we can certainly install some of the other frameworks that are available but, at this point, if your goal is to develop. ASP. NET Web Forms Apps, you're ready to go. So we can run off to start programs. And from here we can get to our Microsoft Visual Studio 2010 Express. So we'll go ahead and open that. Now in the next section of this module I'm going to show you some different features that. Visual Web Developer provides you, and explain how you can use it to build your ASP. NET web applications.

Introduction to Visual Studio

Visual Studio is a powerful development tool that can be used to build ASP. NET applications and several others, depending upon which version of Visual Studio you have installed. Throughout this video series, we're going to be using a free version of Visual Studio called Visual Studio Web Developer Express 2010. Now why use Visual Studio? I mean, there's a lot of editors out there that are very popular, and that provide some great features. Well, I can tell you that, having used Visual Studio for many years now, you'll quickly find it's one of those indispensable tools. It's by far the best editor I've ever used for development projects and I've tried several that are out there, and like several that are out there. However, Visual Studio provides everything I need in one kind of nice package and so I have a Code Editor I can use. So I can get IntelliSense, I can get help as I type my code. I have a Solution Explorer, a way to organize my projects, and the different files that I'm working on within my projects. I have a Server Explorer, we can go out and actually look at servers. We can even start and stop services depending on which version of Visual Studio you have. And then, I even can edit properties of objects directly. And this provides a significant productivity boost, because I don't have to type everything in the code and locate it. I can just very easily make changes and edits to control properties. Now, we also have a Visual Toolbox so that as we're developing ASP. NET applications, if you want something like a grid or you want a button or a textbox or whatever it may be. I can visually drag that on, if I'd like, and that will make it very easy to work with. Now from there, probably one of my favorite features is the fact that we have built-in integrated Debugger. Now, nobody likes to write bugs and I certainly try to avoid them but they're one of those things that definitely crop up. And when they do, Visual Studio provides an excellent way to set breakpoints and watch your autos and your locals and your different types of variables to easily see what's going on and more quickly debug the problem that you're having. So, let's take a look at Visual Studio Web Developer Express, and what I'll do is give you a lap around it. Show you some of the different windows if you're brand new to it and explain how it can be used. Now that we have Visual Studio Web Developer Express installed using the Web Platform Installer, we can go open it. So we're going to go to All Programs, come down to Microsoft Visual Studio 2010 Express and we'll select this first item which is Visual Web Developer Express. Now in the next section of this module I'm going to talk about the different ASP. NET projects available. But in order to show the different windows and kind of give you a lap around what Visual Studio offers, I need to create a project. Now we can do that in several different ways, we can come up to File > New and we can do it that way, or you'll see we get a nice little start page here that allows me to create websites and web application projects. Now in addition to that, you'll notice that we have some nice features here. If you want some tutorials, extra videos, extra walk-throughs that are more text-based that we can go get information there. And we can scroll on down, and this changes over time as Microsoft pushes out different information. We can even get information about hosting our ASP. NET web application. We can come over to the Latest News and we can enable an RSS feed from Microsoft. And what this would do is update this news channel which you see here with whatever that latest news is, and load that news in here. Now from here, what I'm going to do is create this project. So, we're going to go to New Project. I'm going to select an ASP. NET web application for C#. I'm just going to hit OK to create a default name project. Okay, so you can see that right off the bat, we have a Code Editor for our HTML plus some of the other ASP. NET type code that we'll learn about later in this video series. So we have a Code Editor that not only allows us to do it by hand, but we can also switch to Design view. Some people like to work in Split view, where you have the Design view but maybe you want to do your tweaks through code, and then we can always go back to Source view down here at the bottom. Pretty nice. Now there's also C# or VB code, depending on what you're creating projects in that we can get to as well. If I right-click, you'll see I can go to View Code. So if I have some dynamic server side code, I also have an editor for that, and what's nice about this editor is we get full IntelliSense, full help as we type. So, I can do using System. web, and you'll see the different name spaces available are shown to me, and so the IntelliSense really speeds things up. Not only does it speed up your typing, but if you can't remember exactly what somethings called, it'll help you out. Now over here to the right of the code windows, we have the really heart and soul of our Visual Studio environment and that's the Solution Explorer. This is where you'll spend a lot of time really, because as you edit different files, this provides you a window into those files that make up your web application project. So, you can see we have a Default. aspx, that's the one I have open here and that would represent our home page. But I can also get to Scripts. We can get to our different CSS styles and other information right in this Solution Explorer. So I don't have to run out to Windows Explorer or some other tool. Now, from here I can use this to add different files. So let's say we want to add a new JavaScript file as an example. Well, I can right-click on Scripts, go to Add and then from here we can go to Add New Item. And when we add a new item, you'll see there's a lot of different options here. We won't go through those now because we just want to create a Script but I do want to show that there's different views of this. So, we can resize it a little bit and click on this view here which is more of Grid view. You can kind of see everything at once if you'd like that view but we'll go on back to this one. And you see we have this JScript file which represents a JavaScript file. So, I'm just going to go ahead and leave the default name for the illustration here and hit Add. Now that adds in the file and now we can do something that we'd like to do in JavaScript. All right? Now what's nice about this is I can come back to Default. aspx and I can actually take this file and if I'd like I can drag it up into my head area and you'll notice that it automatically puts in the script tag for us. So, makes very easy to work with Scripts, I can do the same thing with Styles. And that's why the server or Solution Explorer is so nice to work with. It provides a really easy way to get to things. All right, now in addition to working with this we can also Preview our page right from the Solution Explorer or I can right-click here, so if I'd like to view Default. aspx I can right-click and say View in Browser. Alternatively, I can come over to the page itself and right-click, and say View in Browser. So, it makes it really easy to test your pages and see what you have going at that point based on whatever it is you want to run. So, this'll load up our page and there's our Hello World we just added. We can kind of go from there, that's what this project gives us by default. Okay, going on back to the Editor. Now we've looked at the different options for Design, Split, and Source view, and one of the nice features is the ability to visually design your pages. So, I'm going to hit Enter here and go over to the Toolbox. Now the Toolbox provides us access to not only HTML type controls, and this would be your standard HTML controls here, but also, specialized type of server controls. Later in the video series I'll talk more about what server controls are but to illustrate the Toolbox at this point, I'm going to go ahead and hit the Pin icon. The reason I'm going to do that is right now, it's kind of in the way. So we'll hit that and you'll see it pins it over to the left here. kind of gets it out of the way here. So I'm going to drag on a Label and then I'm going to right-click on this Label and go to Properties because I want to show you the Properties window. Now, the Properties window when you first start out, I've rearranged mine, will actually look more like what you're going to see here in just a second. Let me go ahead and add this back, let me pop this out and drag it back over and normally Visual Studio starts you out in this view. Now I don't like that particular view it's just, I don't know a little bit too small of a scroll area for me. So, as you just saw, mine, I like to make it the full area. So what's nice about all these windows is you can move them around. So I can drag this out and as I move my mouse over the different areas, you'll notice it highlights things to let me know where it's going to go. Now, I'm going to go ahead and we want to put it right on top of this guy. So you'll notice it put it as a tab right next to my Solution Explorer and then you can move those tabs around if you'd like. So, I like to customize my environment. In fact I like my Solution Explorer actually on the left side and my Toolbox on the right, but we'll leave that as the default. All right, so in addition to that, up top we have the standard Cut, Paste, Copy, Save, all those types of things. As I highlight text, you'll notice I can change to different types of headings and paragraphs. I can change my fonts and all that type of stuff very easily, pick colors. Pretty standard type of stuff you'll find in any HTML editor. In addition to that, we have up on the menu, different views of, we can get to Rulers and maybe we want to show a Ruler or we want to show a Grid on our page to line things up exactly. Or maybe I want the Ruler, let me go back to that and that allows us to get this Ruler here to kind of absolute position things if we'd like. So from here, what we can also do is go in and not only run the page, but we can also go in and Debug. And this is a really nice feature of Visual Studio because it allows us to actually set what we call breakpoints and step into our code and if we have any issues, we can resolve those. So let me show a quick example of that. I'm going to right-click on this screen and say View Code. We're going to come back to this page load here. In this, I'm just going to add a really quick, kind of bogus variable just so I can set a break point. We'll do a string and we'll just say Hello World And now what we want to do is I'd like to maybe debug a problem we'll kind of simulate. So, I'm going to set a break point by clicking in this gutter to the left side. I can either hit F5 or I can hit this icon up here that says Start Debugging and if I hit that, what it'll do is run my project and run this Default. aspx that you saw earlier. Because I set a break point and I ran it in Debug mode, you'll notice it set a break point here and hit the break point, I should say. And now, down here on the bottom, I have access to a really rich set of variables, and I can actually drill down into these to see what's going on in my application. Makes it very easy and much faster to identify problems as they occur. What we can do now is, up here on the top, I can either use Hotkeys such as F11 or F10 to step over. We're going to do a Step Out and now, it executes my page and I can run it. So, the integrated Debugger is a really, really nice feature that allows us to easily find out what's going on with our web application. Now, in addition to that if you want to customize things in Visual Studio we can come up to Tools > Options. And this will get us into things like maybe I want to format how tags are displayed. So maybe the anchor tag, I want to highlight it with a specific color as an example. Or I want to add a new tag that's not currently showing here and we can do a lot of different things with our fonts. If I want the default font size to go up or down for my Code Editor window we can do that plus there's a lot of other settings that you can customize for XML files, brackets, all kinds of fun stuff there. So that's a run-through of some of the different key windows in Visual Studio and specifically in this case, Web Developer Express.

Creating ASP.NET Projects

ASP. NET Web Forms supports several different types of projects that you can use when creating your web applications. So in this section I'm going to walk-through some of those projects with you and explain how they work and demonstrate how you can create them, using Web Developer Express 2010. Now, the projects themselves can be broken down into two main categories. You can do Web Site Projects, or you can do Web Application Projects. Now, a Web Site Project, allows you to work directly with folder. So you can literally point Visual Studio to a folder and open it up, almost like you would in Windows Explorer. And any files in that folder will display in the Solution Explorer, and then you can edit those. So it provides a more simplistic way to work with different files. The Web Application Project actually builds a solution with a Web Project that you can work with. Now the ways these different projects work and compile are quite a bit different. I'll talk about that in the next section of this module. But for now we're going to focus on how do you create these types of projects and what other types of projects are available. So let's go ahead and take a look at that. You have two different ways to create projects when you're building ASP. NET Web Forms applications. And these can be created directly in Visual Web Developer 2010 Express. They're very easy to get started with, so we're going to talk through those. And then, in the next section in this module, I'll also walk-through the compilation model whether you're using C# or VB or even another language and explain how that works. So that you have a complete understanding of solutions and projects and how they compile and work. So, in this section what I'm going to show is how we can create these two different types of projects and show where you go to do that. So, to get started the easiest way to do this is to go to your Start page, and either click on New Website or New Project. Now, that does create different types of application structures in the Solution Explorer over here to the right. Now, if for some reason, you don't see the Start page, you can always come up to View. And click on start page and this'll take you right back to it. But that's not the only way to create these, there's actually this, actually two other ways. alternatively, I can go up to file and do new project, or new website, and I'll show you both those, or I can go to the menu, click here and do new project, or new website. So, there's a lot of different options to get you started here. Now, so let me create a website first and explain what the difference is between that, and in just web application projects. because they are quite a bit different in the structure, but not so much the files. So I'm going to click on New Website. And you'll see that I have a couple options here. So, on the left I can choose either Visual Basic or Visual C Sharp. And then, from here, I can click on an ASP. NET website. And that will give me a nice starter website that I can work with, that includes logging functionality, registration, and even access to script libraries like jQuery. I can also come in and do an ASP. NET website with a special type of syntax called Razor and this relates to ASP. NET MVC, and there are a set of videos on that as well if you're interested. I don't want to get into that. An ASP. NET empty website does what it says. You basically get an empty website and you'll need to add in all the different files. Now, this is nice if you want to start from scratch, and you want to build up the website file by file. Now, if you've ever had to create an application that has to do maintenance type screens against data base tables, so maybe you have some look up tables, or customer tables. And instead of allowing people within your department or even out, throughout the entire organization, to actually go in to the tables directly, you want to have them do that through a website, then you'll love the ASP. NET dynamic data features. Now, there's two styles here. You can do something called entity framework or link to sequel. And I'm not going to show that now, although we will later in this video series, but this provides a great way to build admin screens. And you could do much more than that, but that's one of the nice features I really like. The final template we have is a WCF service. And a WCF service is simply a way to build a web service that's interoperable. So, I can exchange messages in a cross-platform, kind of language neutral way, with other frameworks out there. Now, for this particular web site demonstration, I'm going to select ASP. NET Web Site, and then you'll notice the path here by default saves into documents/visualstudio2010/websites, and it's going to call it website one. Now, we could name that, FirstWebsite, and that's going to save it to the file system. Now, you do have some other alternatives. You can actually save it if you have some other features related to HTTP going such as WebDAV, then you can use HTTP, or you can even do FTP. And this makes it really easy to work with a FTP server if you'd like. So, we're going to go ahead and select file system, and I'll hit okay to create our website. Now, you can see it creates it really quickly and you can, you know right off the bat if it's a website, because you'll see C drive or D drive or, wherever you said to save that website, that's going to show up here with our project type. Now this literally is just a folder that has these files in it. And why some people like websites over web projects, which I'll show you next, because you can literally just open a folder and start working with files. There's no projects involved. It's all folder based. So, if I close this, and I'm going to go into the location of where it saves these, which is the documents Visual Studio 2010, that I just showed you earlier, we can go back into websites and go to first website. You'll see it's just a bunch of files, nothing fancy at all. So, I can go back now, and we can say file open website, and literally just pick a folder, any folder, as long as it has those websites files that you want inside of it. And we can go to those, so we'll come down, and we can go to, documents, visual studio, websites, and click on first website. We can open that back up, and you'll see we're back to where we started. And then from here, like any project in Visual Studio, I can hit F5 or I can right-click on my designer, or on a specific file, and select View in Browser. And that'll pop up the application and display it in the browser for us. Now if I have a, a nice little thing to point out here is, if I have other browsers, I want to test, which is certainly very common, I can even do Browse With. Now, on this system we just have the basics installed, so I just have Internet Explorer right now. But let's say you want to test with Chrome and Fire Fox and Safari, and you have all those installed. You can actually right click and say Browse With, pick that browser, and then browse to it, and I can even set a different browser as the default, and even change the resolution here. Makes it really easy to test things and work with different browsers. So, you can see a website project is, is folder based. It's just folders and files, very, very easy to work with. So, let me go ahead and close this one. And we're going to go back to the start page. And now let's do, New Project. Now this is going to be quite a bit different than a website. So, I'm going to click on New Project, you'll see right off the bat I have a lot of different templates, I can do either Web Forms applications, or ASP. NET MVC applications. And as mentioned, if you're interested in MVC, there's also a video series available for that too. But we still have the Dynamic Data, we have our Web Forms at the top. I can even do some AJAX controls, or build custom ASP. NET server controls. This is useful if, for instance, you want to build a custom grid that outputs rows and columns, while this will get you started building a custom ASP. NET server control that you can use to generate HTML. So, we're going to come back up to the top and create the same thing I did before. But you'll notice I have a lot of different options here. Under Windows, I can create a class library. That's useful for things like business rules, that I might use in my application or even data access classes, something that I want to break out into a separate project from my web project, if needed. We have all the web projects. I can come into cloud, and we can work with Azure, which is Microsoft's cloud service. I can build Silverlight projects and do WCF Services. So, we'll come back, we'll do the ASP. NET Web Forms application. Again, it's going to save it to the document, Visual Studio 2010 projects by default. If I want to change that I simply hit browse, navigate to wherever you want to move to, and simply give it a name. So, let's call this FirstWebApplication. And we'll hit OK. And you can see, it looks at this point, identical to what we just saw with the website. However, it's quite a bit different behind the scenes. First off, you'll notice there's no C drive or D drive up here, although the files look identical. Now to show you the difference, if we go into Windows Explorer. Let's go back up to Visual Studio 2010. WebSites is where we created that first website which was the folder and simply the files, very basic. Projects is where this particular one is saved. And here is FirstWebApplication. Now, inside of that, we can't see the file extension, so let me go ahead and change that real quick. We're going to go and say, Show Hidden Files, Folders and Drives and turn off a couple of things here. All right. Now, you can see that we have an SLN file. It's called a Solution File. Now a Solution is used to store one or more projects, so think of it as a container that allows you to store not only your web project, but any other projects you might have, such as a business rules project, a data access project, and others that you might create. Now, the project in this case, or the solution, I should say in this case, only points to one project, which is this guy. And inside of it, in addition to the files, you'll notice that we also have this thing called FirstWebApplication. csproj. Now, this is a project file, you don't need to open it or work with it, but just to show you briefly what it does. Let me open this in Notepad. And this links to all the files in your application. So, as opposed to a website type of project that we created earlier, where there really is nothing like this. It just opens the folder and whatever you put in the folder is what opens. This is an actual project with a project file that tracks everything. Now, if you're wondering which one's better it really depends on your preference. My personal preference is a web application project. You, you have a lot more flexibility with the different types of projects that can be created and used. But there's still people that they prefer to just open a folder and just start working with it. So, that's something you'll need to choose. All right, so moving on here, let me show you another feature related to solutions that you can do with website projects and with web applications projects. Let's say that we do have some business rules. And I would like to integrate those into my web application, but I want to write them in a reusable way. So, I don't want to put them right in this FirstWebApplication, because then they don't be useful there. I want to make them in a separate project, so that I can reuse that across different web applications. So, I want to maximize my reuse, simplify maintenance, make it easy for team members to find all that good stuff. But the way we can do that is if we come up to File, when I click on this project here, I can say Add New project or Existing Project. Now, if I had an existing, I can link that in. What we're going to do, I am going to come up to Windows and we're going to do a class library. Now, let's just name this BusinessRules, and you'll notice that it's actually going to save it in this FirstWebApplication. And you can certainly change that, though. We're going to hit OK. And now you'll notice that I have two things and we actually see a solution. Now, you don't see the solution by, by default earlier, because there's only one project, so that you know, they kind of take out the clutter make it easy to work with. Well now, I can go ahead and drill in write some custom business rules here and be ready to go. Now, what if I want to link these two projects together? Well, all I have to do, it's very simple, is if I right click on FirstWebApplication and it needs, let's say, to reference my BusinessRules, then I can right click. We can say add reference. And I want to add a reference to a project. So, I'm going to click on the project's tab, select BusinessRules, and then hit OK. And what that just did is, you'll see BusinessRules shows up in the References. So, whatever I write up here will now be accessible to use down in this project. But that's all I'll go into there. But that's a very common thing that you need to know about solutions and projects in case, in cases and situations where you want to break things out. Now, from here, I'll leave it up to you as to which you want to choose. So, we now know though that we can create website type of projects that are just file and folder based, and we can do full-on web application projects. So, that's an example of how to create those two different project structures in Visual Web Developer Express.

Compiling ASP.NET Applications

Both C# and Visual Basic are compiled languages, and that means if you build an ASP. NET website or web application project, ultimately the code you write for it does have to get compiled, so that it connects it here. So, let's talk about compilation in the. NET framework and how it works. So, you'll of course start out with your source code. And you could write source code in any editor out there. But, Web Developer Express is definitely the easiest way to do it, because it has all this built-in functionality. Now, you'll save your files with a. cs or. vb extension. Now, when you're creating ASP. NET web forms, those will be created for you automatically behind the scenes when you add new web forms pages into your projects or folders. Now, in order for those pages and the C# or vb code to be useful, you have to run them through a compiler. Now, for C#, the compiler is actually a command line tool called csc. exe, or for Visual Basic, it's vbc. exe. Now, in Visual Studio, when you go open build, that will actually invoke the appropriate compiler based upon your project, language type. And what'll happen is, it'll compile all your code in the project, and you're going to get something called Microsoft intermediate language, or we just call it MSIL. Now, MSIL has several things associated with it. Number one, it has some metadata, and the metadata will say what's the version of it, what dependencies does it have, as well as other things. And then of course, you'll have the code. Now, the code's not going to be your C# or VB. It actually is a specific language called intermediate language. Now, the very first time you run a website, you'll see a very slight delay, depending on how big or small your code is there. And, what's happening with that is the first time it runs, the common language runtime of. NET takes your missile code and actually runs it through a just-in-time compiler. Now the just-in-time compiler, takes that code and compiles it down to Native Code. So, it's going to be very, very fast. Now, if you don't ever touch your code again from there, every time the website runs it just comes up instantly as it's served off the server. And that's because the Native Code is already available and ready. Now, if you recompile your website and recompile your C# or VB, you're going to get some new MISL, and so that will again run through the JIT process, which will regenerate the Native Code. Now, that happens automatically behind the scenes. The only thing you have to worry about, is actually doing the compilation and that even depends on which type of project you're selecting. So, let's talk about the different types of projects again, and how those work. So, the ASP. NET web application project, the one we talked about in the previous section, that actually has solution files and project files. That is something that you actually have to build, and you get a dynamic link collaborator, DLL as a result of building now the. DLL is your missile. And the first time that runs, again, it'll JIT it and then you'll get native code. So when you select a web application project. If you hit F5 or you hit the little play icon in the toolbar to run that we application, then it is actually compiling it and do a DLL. And then from there, you can work with that and load the website. Now, if you do the ASP. NET website project option. The one where you have a folder and it just has the files inside of it. In that particular case, you actually don't have to build. You can literally change a C# or VB file and just run a page. And what will happen is it'll automatically compile in the background for you. Now of course, if you get an error, it's going to display an error page in that compilation process. But assuming it all works, it'll run through the JIT process. And so it still follows the same compilation model. But you don't have to do an explicit built. Now, one of the big things to take into account is when you move these types of projects to production. So if you're selecting an ASP. NET web application project. You have to do the build and you get the DLL. And you need to copy that DLL over to your production server. If you're selecting the ASP. NET website option, then you're not going to have a DLL. And what you can do then is literally just copy the folder and all of the code files over. Now, in some organizations it may be against the policies, security wise or something else, where you're not allowed to put the source code files on the production server. If that's the case, you'll definitely want to go with the web application model. If you're going with the website model, then by default, you'll copy those files over. And so you're C# or VB files will actually sit there on the server. Now they're safe and secure. You can't hit those directly. It's all blocked. And that's been proven over. You know, ten years now, nine or ten years, I should say of. NET being out. Now there is a way with the website model that you can deploy a DLL as well. But that's an extra step you would have to take. But that's really the differences in the compilation model. So let's pop up a Visual Web Developer Express. And let's take a look at these two types of projects again and see what happens behind the scenes with compilation. Let's take a look at the different compilation models available with Web Site and Web Application projects in Visual Studio. So I'm going to open up first website. Now, this is a website project. Meaning that we have folders and files and that's it. No solutions, no projects at this point anyway. So, I could come in to default axps. cs. And we could add some code. We'll say hello = Hello World. And maybe we even write that out to the page. We're just going to keep it simple. Now, I can save that and instantly go in and right click on Default. aspx and View in Browser. Now, this will pull it up and it's going to do the compilation to missile and then down to native code, the JIT process for me. So I don't have to do an explicit build. In fact, I won't see a DLL. That would be normally be generated when you build. Now, behind the scenes, that's all happening, of course. But it's doing some magic for us so we don't have to worry about it. Now, if you're coming from a more scripted application type background versus compiled, this type of process and workflow will probably be more appealing to you. It's more along the lines of probably what you've done. So that's what we have to do to work with a website project. Now, if I move this to test, and then ultimately to product, I would need to move the code files by default. So I'd have to move the ASPX files and the code files. Now, there is an alternative there with deployment, where you can compile this all up into a DLL. But out of that box, that's not happens. You would simply copy this entire folder over to your target server. Let's close this project. Go back to the start page. And open the web application project. Now, this one is quite a bit different. You'll notice I already have the string now. And let's say I decided I don't want it. We're going to take that out. Now, I can't just run the web project. Now, if I hit the Play icon to start debugging, or just hit F5 directly, it'll actually build first and then run the application. So it is doing a build in that case. And what I'll get is a DLL, behind the scenes. Now, to show you how to build explicitly though, on doing it yourself, we can right click on first web application and say build. Now, you'll notice one succeeded, zero failed, zero skipped. So that's good. And we get this little item down here. Now, as we're building, if I had an error, let's go ahead and leave that incorrect. We'll do a build. You'll notice that I get a nice error. I can now double click on it. It takes me right to the source of that. And now we can fix it. I'm just going to go ahead and take it out. We're going to do a build again. And we're ready to go here with our application. Now what happened behind the scenes? Well, as I showed you just a bit earlier, we actually get a DLL when you're working with a web application project. So I'm going to hit Show All Files. This is not available in a website project because it's just folders and files. But it is when you're doing a web application project. So we'll hit that. We'll come down into bin, and you'll see, FirstWebApplication. dll, dynamic link library. So that has all the code from all my code files that you see in this project here. Now when I'm ready to move things off to production or test or whatever you're doing with it. I do need to copy the bin folder, as well as the ASPX vfiles. Now, I don't have to copy the. cs or. vb files though. Because they've already been compiled into the bin. Into this DLL that you see. So, the difference there, of course, is with the website project, you don't see the DLL. It kind of does it behind the scenes. And you do need to copy by default the source code files onto your target server. Now, if you work in an organization where you're not allowed to have source code files due to a security principle or rule that you have. Then, you'll probably want to go with the web application project. Because then you have the DLL and everything's compiled into that DLL. Now, rest assured, if you do select a website project and move this source code over to production, you can't hit. cs or. vb files directly. And that's been proven out over since. NET's been released many years ago. It's very secure and safe. So that's what happens. Now, another way you can build is, as I mentioned, you can hit F5 or you can do start debugging. But if we come up and right-click, I can also add a build item to the tool bar up here. So, I'm just going to right click. And you'll see there's a lot of stuff I can add here to add to the toolbar and make it easy for me to work with this. I'm going to select built. And you'll notice this guy right here, which I'm going to move over. Let's add it right about there. This has a couple icons on it. So I'm going to select FirstWebApplication. You'll notice that I can build first web application. Now, right now, this is in a solution which is one project. If I had multiple projects, I can hit build solution, or Control + Shift + B. Now, you can also build, if you're a hotkey person, by just doing Control + Shift + B. So, that'll just build everything in the solution. That's your hotkey. So we can do a build this way as well. Now we're ready to go. I've built. I get the DLL. And now we can run this. Be a very slight delay the first time As it jits. Gets down to native code. And now we're off and running. And you can see that we have our project. So that's the difference in compilation models between website and web application projects.

Using Web Forms

The final topic that we're going to cover in this module is how to get started using ASP. NET Web Forms. Up to this point, you've seen how to get Visual Studio Web Developer Express installed. How to create projects and how they compile. But we really haven't built much of anything to this point. Now, we haven't learned about server controls or data access technologies at this point. So I'm going to keep it very, very basic. But it'll help you get started. So let's go ahead and jump in. So we've already seen how to create projects. So I'm going to come in and create a new project. And we'll select web ASP. NET web application. Now, what I'm going to do though is, I don't want to start with the default website that you've already seen. I'm going to come down and do an Empty Web Application. And add a page by default and kind of work with that page. So we'll come down, and let's go ahead and call this FirstDemoWebsite. All right. So that creates a very, very simple and pretty much empty web application project. In fact, all we have is a configuration file that allows us to put some of the configuration for our ASP. NET web application. And we're not going to need that in this particular demo. What I am going to do though, is right-click on my project. And we're going to do Add > New Item. Now, the item I'm going to add is a Web Form. So let's go ahead and call it Default. aspx, since that's the standard for your homepage in the ASP. NET websites. Now you can name it whatever you want though. That's what I like to do. Now let's build a really simple form from here. So I'm going to switch over to design view. And in design view, I'm going to leverage our toolbox to drag on some of the different controls that you see here. Now, these are server controls. I'll introduce server controls officially and talk about several of these later in a future module in this video series. But for now, we're going to build something very, very basic to capture some form data, have a way for the user to input it, click a button and then work with it from there. So I'm going to go in and drag on, first of all, a Text Box, then a Button and then a Label. So let's say we'd like to capture the user's name. So we'll type name. I'll then come in and drag on a Text Box. Now the Text Box is currently text box one. Let's go ahead and right click and go to Properties. And we'll go up to the top. And change the ID to something a little bit better. I'm just going to call it Name Text Box. And then I'll come in, go ahead and hit Enter to give us a little space. going to hit that twice. Scroll on up. And let's drag on a button control. We'll also go to Properties. And we'll change the Text property to Submit. And that'll update it. Now, keep in mind that everything I doing here, although it's visual, if your someone who really doesn't like to do it visually, everything can be done in source mode if you'd like. You can type in these controls directly. Or I can even do split view mode, where I can type it in. Plus see what I'm doing as I type. And you'll notice as I highlight the different controls, it actually highlights those in the designer. So, it makes it really easy and nice to work with. Now coming on back over, let's say that once they submit this, we'd like to write out what they typed for the name. And we want to go ahead and write out to somewhere down here below the button, to a label. So, I'm going to drag on a label. We'll right click go to properties. We're already there. And I'm going to take out the text. We don't want to show anything when this first loads. We only want to show something after they've submitted something. So we'll come up to the ID. Let's give this a name of Output Label. That'll be the name or ID for it. And you can see, right now, it shows that. But because you see the square brackets around it, it's not going to render it at runtime. So you won't actually see Output Label. That's just showing us what the name is. All right. So this probably the world's most simple form that we can build. But I'm going to show you how easy it is in ASP. NET Web Forms, especially if you're brand new to this framework. To get started and make it so the user can type in something, we can click the button and automatically update this label with just a very minimal amount of code. Now to do that and to start this process, I'm going to double click on my button. Now what that makes is an event handler. And this event handler called Button1\_Click is what'll get called when they click the button. So what'll happen is inside of the source, you'll notice that we have this form tag. And of course, as with standard web technologies, when you click an input type equals button or an input type equals submit, we can actually go in and submit the form. It's going to post back. Now, some of you may be used to actually having an action up here. And that action may be a different page. Well in ASP. NET, by default, whenever you post back, it's going to post back to the same page. And you don't have to worry about much there. You see that the plumbing is very, very simple on your end as a developer. In fact, it's going to do all the heavy lifting for us. But it's going to make it more like a desktop development model. Meaning that we can double click on the button, we get an event handler. Behind the scenes that's using standard web technologies to redirect and detect that the button was clicked, the form was posted back. And then. NET will take care of invoking this button one click event handler method. Now inside of here, we can grab the text box text, and assign it to the labels text. So we'll come in and we'll say, output label. text =, and now we can go to NameTextBox. text. And that will simply go in, grab the value of the text box and assign it to the text of the output label. Now, you notice how simple that is. I didn't have to go into the request variables. I didn't have to check the form posted back data. You know, update the query, or grab the query string value or anything like that. I simply go to the control. Grab that. Go to the property of the control I want to get and then assign it to the other control. Now what'll happen here is these controls will now render the HTML back. And we'll see the posted back screen. So it's all going to stay on default. aspx in this case, because that's how it works by default. All right. So we're pretty much ready to go. So let's go back to the solution explorer. Let's make sure that we build okay. So I'll right click on it. Looks like we get one succeeded. So that's a good step. Now we can go ahead and right click on default aspx. We can save you in browser. And this will pop open in browser and render our form. Now remember, let's go back before I go any further that I drug on a text box control. You'll see this has a little bit of a different thing than you might be used to if you're doing pure HTML. This is called a server control, if you see asp:. We also have the button I drug on, and the label. Now, remember what you see there. Because when I go look in this in the browser, let's view source and see what's rendered. So let's refresh here. Here's my text box. Here's my button. And what we'll do is right click and we'll go to view source. Now if we scroll on down, you'll notice that I don't have that ASP text box or the ASP button or the label. It now rendered standards compliant HTML. So the control that I drug on is just a friendly way to work with text boxes and buttons and many, many other things that ASP. NET supports out of the box. You'll notice it actually made the ID and the name the same as what I gave it on the server side. Same for the button. We didn't rename it, so it called it Button1. And then for the label, it actually output a span tag. But the span tag doesn't have anything inside of it, because I took out that text earlier. So that's one of the nice things about ASP. NET web forms is that there's some very sophisticated controls, like the grid view, or the details view that we'll talk about in later sections of this video course. They take care of rendering the HTML for you, so that you can just focus on your application, the business logic, and get that UI working properly. So, now that you've seen what's generated here, let's go ahead and run it. So, we'll just enter my name. Do Dan Waleen, hit submit. And you can see it echoes that back to our label. Very, very simple. Now there is a post back operation occurring. It's going to be very fast, but it does happen behind the scenes. And if we go now look at the source, what happened is, on the server side, we updated the label. And let me just go back to the source here. So we had that label. When it was posted back, we took the value of the text box and updated the text of the label. Well, coming back in to our source, you'll see the span now has the name inside of it. So ASP. NET automatically detected the post back, handle that, and then went in and updated the label control which then outputs this spam tag. So very, very simple to get started with. So that's an example of building a very basic kind of starter ASP. NET Web Forms application. And as we move forward in this video series, we'll be doing more and more sophisticated things. We'll talk about data access. How to get data from databases into our apps. And show many of the other server controls that are available in the framework.

Summary

In this module, we've talked about what is the. NET Framework, and what does it mean to you as a software developer? Specifically, what does it mean to you as an ASP. NET web forms developer? From there, I introduced the Web Platform Installer and showed you how it makes it really easy to install, not only Visual Studio Web Developer Express, which will help you get going. But also other ASP. NET web forms projects that are out there, including Open Source projects. It's a really nice one click type way to get something installed without having to go find it out on the web somewhere. Once we got Visual Studio installed, I walked you through the interface. And then we walked into how do you create different types of projects and discussed ASP. NET web forms website projects versus web application projects. And from there, I explain the compilation models. As review, website projects don't create a DLL. You have to copy the source code files by default over to your target server. Whereas with the web application projects, in that particular case, we do get a DLL and you need to do an explicit build. That DLL along with ASPX and other supporting files, such as Script and CSS files, need to be copied over to your target server. Once we discuss that, we walk through a basic getting starter type of ASP. NET web forms application. Showed a little bit about the post back model and how to handle basic events. So that's a wrap up for this first module to help get you started and introduce you to ASP. NET web forms development.

ASP.NET Web Forms Features

Introduction to ASP.NET Web Forms Features

Hi, this is Dan Wahlin with Pluralsight. In this module, we're going to talk about ASP. NET Web Forms Features and we're going to focus on some of the key features available. And kind of break down how ASP. NET Web Forms are organized and how they work behind the scenes. So, we'll start off with a higher level look at what features are available. Now, some of these features, we'll drill down into in later modules. Once we go through that, we'll talk about the concept of code separation. If you've ever worked with a web frame work where your code and your HTML is kind of interspersed and gets a little bit messy and spaghetti codeish. Then, you'll appreciate code separation in ASP. NET Web Forms. This allows our html, and our programming logic, to be kept separate and very clean. From there, we'll move in to the page directive, and talk about different attributes. So, I'll first explain what is the page directive. What does it do? And then, we'll talk about some key attributes that can help not only make your life easier, but even keep your users happier. From there, we'll move in to one of the key parts of ASP. NET web forms and that is Web Controls. If you're going to be building successful ASP. NET web applications then you're going to be using Web Controls. And web controls, in a nutshell, simply generate html with a little bit of effort on your part. So they can do a lot of work to help quickly and productively build ASP. NET applications. Now part of the Web Controls, options are the ability to validate user inputs. We'll talk about those, and explain how you can do things such as make sure required fields are entered. And even validate against different data types as users fill out your forms and other types of data input screens. To wrap up this section, we'll talk about how we can set defaults, and this includes setting up the default. Focus to controls, a very good thing for convenience to end users, as well as which button. If you have multiple on the screen, is the default button if the user hits Enter. And I'll show you how easy it is to set that. Finally, we'll talk about what user controls are, and explain how you can use them. Now, user controls are used for things like headers, footers, menus, anything that's going to be shared across multiple web pages. So, I'll explain how to create those and how to use them in your pages. So let's get started by talking about some of the different features available with ASP. NET Web Forms.

ASP.NET Web Forms Features

ASP. NET Web Forms provides a robust set of features that are available to us as developers that we can really leverage to not only be more productive but really write a lot less code. And hopefully less bugs in the process. So if I had to sum up what an ASP. NET Web Forms page is, it's really a programmable web page. So, as opposed to having just a regular HTML page the overall goal of an ASP. NET Web Forms page is to serve up HTML but dynamically. So, we've been able to do that for years. We've had PHP and we can use a JSP we could use classic asp. But, what really sets the ASP. NET Web Forms model apart in my mind is that number one its very very easy to do a lot of work or get a lot a lot of work done without having to write a lot of code. And so, we'll talk about and touch on web controls later in this module. So, the overall goal is to serve up HTML dynamically and as we do it what's nice is everything is object oriented. So, if you're doing C# or VB, you're going to be writing classes with properties, methods, events, those types of things. But every page that you write will ultimately derive from the ASP. NET Page class. This gives us a lot of functionality. We can do things like set focus using it. We can go in and check if things are valid. We can check if there's been a post back and much more actually. So, by using the Page class and web controls, it's really easy to display data, we can collect data, and as I just mentioned, we can validate our data as well. Now, if I were to break down the features of ASP. NET Web Forms, we can kind of break 'em down into several different groupings. So, the first is ASP. NET controls. So, when you get ASP. NET and install either Visual Studio or the Web Developer Express that we'll be using, you can use Security controls. That's just a built in out the box and we'll actually have a whole module talking about some that later in this video series. Now, Security controls make it easy to log in and log out. We can do registration of users and even, what if they forget their password? Well, we can help with that, too. Data controls are, at least for me, really where it's at though. Most of the applications I work on are line of business applications. For different sized companies, large, small, medium, it really depends. But they generally display and collect a lot of data. Well, what really sets ASP. NET Web Forms apart in my mind from some of the different options out there is how easy it is to integrate data. Into your applications. And so, we'll have a couple modules on data access techniques and controls. We'll talk about things like the grid view control that makes it super easy to output columnar data, and there's many others. Now, another feature of web controls that's built in to ASP. NET Web Forms. Forms is Navigation. If you'd like to do bread crumb trails so the user can basically track where they are. If you'd like to do a treeview control or a hierarchical menu. All that stuff is built in and you can even create a specialized XML file called web sight map and that makes it easy to tying the breadcrumb trails. Menus and things like that. Now the final kind of general category is Web Parts. Now web parts are actually what share point is built upon, uses ASP. NET to do most of its core integration, and you can even use web parts outside of share point, and that's also another big thing. Now a few others to note here that I personally use from time to time are there's also AJAX controls that we can use. So if we want to perform partial page post backs where we don't re-load the whole page, that's also available. And we'll look at some of that. Later. Now some of the key features we do have full support for you can say laying out our page. What we call Master Pages. Now a Master Page is really just a template. Nowadays it's very common that. As you navigate, navigate to different pages that they all have the similar look and feel. Now, that was quite a bit different than when I started in the mid 90s with this where almost every page looked different. Well, now using Master Pages it's very, very easy to add consistency throughout all the pages in your site. And really simplify your maintenance. Now another nice feature for people that need to scan or theme sites is the ability to create skin files, and custom themes with CSS. That's also a feature built into ASP. NET and it allows you to really leverage CSFs to the max if you need it. Vocalization is also available, this provides built in support for multi-lingual type apps. So you might have a English. You might have German, Spanish, whatever it may be. And then the web controls that we're going to be talking about are adaptive. They can generate HTML to browsers based upon the browsers capabilities and features. Now there's also a bunch of services that are available. One of the biggest things that is generally. Kind of a pain with general web applications is Security. So we have two types of things we can do that's really nice when, when it comes to Security. And that includes a Membership provider which allows us to track user membership. And log users in and out of a system. We can use Windows authentication or we can even use custom databases and Microsoft actually has one that works out of the box, very, very nice and easy to use. We can also manage roles. And authenticated applications often times it's not enough to just know the user. We want to actually go in and show or hide different parts of the page based upon specific roles they may be in. Now, moving along to some of the others we have Personalization features, so if you want to track personal settings for users that's also available that can even tie into that database I mentioned. I've already talked a little bit about Navigation controls but there is a navigation provider so you can even customize that to say store your navigation in say a database. Really robust support for caching now. I really can't underestimate how. Important that is for high skill ability sites and high availability sites if you have a site and the data doesn't have to change every second then you can actually cache entire pages parts of a page or even go down to the individual data objects and then finally we have some management APIs to allow us to. Take a look at what our website's doing and some of the different things that are going in with it. So, what's in a Web Form actually? Well we've talked about the Web Forms inherent from the page class and there's several things though that you'll find inside of. The ASPX part of a Web form. And that's the extension we're going to be seeing, what we've seen up to this point actually. I'm going to run through these real quick. Some of these we'll talk about quite a bit as we move along in this video series. Others are maybe a little more rare to use, but you may come across them. So the first is Directives. Now, one of the most common Directives is the page directive. So, you'll notice this % @ symbol on the left side. And the page directive in this particular case, we know it's a directive because of that % @, it defines things like the language, the code file, C# or VB, that might be associated with the page. And those types of things. So again, all these things we're talking about will go in the ASPX page. The server side script blocks allow us to integrate C# or VB code right into our ASPX page. Now I personally don't use this feature very much. I prefer to keep it totally separate. And we'll actually be talking about that next. You might also see Render Blocks. Now these are pretty common across multiple frameworks. Including classic ASP. The predecessor to ASP. NET. This allows in this example the user details variable. Whatever the value is of that. To be written out from the server side and injected into the HTML content. So that percent equals simply means write out the value of that variable. Now those aren't used quite as commonly as they used to be but there's occasions where you might use those. That brings us to really, as I mentioned earlier, kind of the heart and soul of web forms and that is web server controls. Now you'll always know a Microsoft web server control, the default ones anyway because they'll start with this ASP colon and they'll have the runat Server on the tag. Now these are controls that run at the servers side and simply generate HTML. So in the case of the label, this simply generates span text and there's many more we're going to be talking about in this video series. Now, there's a couple others you might come across, depending on what you're doing, and that is user control, which we'll talk about at the end of this module. Again, that's for headers, footers. Things that repeat across pages and then you also may see ASP. NET expressions. Now expressions are a little bit different because unlike directives that use percent at. Expressions use percent dollar and so this particular expression would allow us to read a connection string. Without having to write any C# or VB code to do it. So this would go into ASPX page and pull out the north wing connection string from a file called web. config. Very useful as you're using data source controls, which we'll be covering. And then finally, we'll be talking about Data Binding, one of my favorite. Features of ASP. NET and something you'll do frequently. Each time we see a %# that is a data by an expression you can do what it shown here and you can use the Eval function to write out a field name. So, you might have a result set that comes back from the database. And you want to ride out a specific field in the web page. So, we'll be talking about these in conjunction with items control such as the Grid View. So, there's a lotta great stuff that's available. And as we move along in this module and others, we'll be introducing these features with more detail.

Code Separation and the Page Class

One of the nice features in ASP. NET Web Forms is the ability to separate your ACML from your programming line. So you ACML, CSS and Java Script can go in one. Page and your C# and VB code can go in a separate one. Makes it very clean and very easy to work with especially if you do things such as production support. So the default option with ASP. NET and Visual Studio is to create two separate files. When you add a new web form into a project. So in this example, we have form. aspx. This would have all the HTML, the server controls, the javascripts, the CSS that might be linked in. Things like that in this page. All of my programming code would go here. In form. aspx. cs, or if I was doing Visual Basic, it would be. vb. Provides a very clean approach to developing web applications. And having come from a classic ASP back in the late 90s scenario, which is what I did a lot of back then. This is much better, much cleaner. I can get rid of that spaghetti code. Now there may be times you don't want that though, you want just one page. You can do that too if you'd like. It's not the default but it is possible. You can use the script blocks. Put run at equal server on it. Give it the language, that's the C# or VB. And then put that script block either at the top of the page, or at the bottom of the page. Then you would just put your HTML, your server controls, and the other things, in as normal. Visual Web, Wed Developer Express, doesn't provide a way to, out of the box, create this single file. But if you do have the paid versions of Visual Studio, you'll see a check box down at the bottom, when you create a web form. And this will allow you to either do the code separation, which is the default. Or do the single file. Now, as I've already mentioned, I definitely prefer the code separation and would recommend, especially if you're new to this, that you go with that approach out of the box. I think it's a good way to go and it keeps it clean. So, let's take a look at the fundamentals of doing this in Web Developer Express. In order to demonstrate code separation, I have Visual Web Developer Express already open. And I have a web application project created. I used C# for the language. So, out of the box, if you create a web application project you'll get two files About. spx and Default. spx that are web form files. So you'll notice that if I open these, if I hit the little icon here that there's also two code files under those. Now the first code file is your code behind or sometimes you'll hear it called code beside file and really it's just your language files. And that's where you can go in and dynamically drive the HTML and controls that might be in the ASPX page. Now, this designer. cs or designer. vb if you're doing a visual basic file, this is something Visual Studio creates. And as you drag and drop controls onto a design surface, it'll update this file, so this is something you definitely don't want to touch, don't want to modify it. And it kind of gives you a hint to that right here, because it will regenerate this file as changes are made. But this first one, About. aspx. cx and default. aspx. cs. These are your files. These are the files you can use. So, to do a real simple example of this. In About. aspx, you'll notice that I have some basic HTML here and I have some server controls. We know that because of the asp colon. And we see the runat server. So what I'm going to do is dynamically set the value of the current time to this label server control and the label server control as a review generates span tags, but I want to dynamically drive this. So the way I can do that is I can either a, go into here and double click on it to get to here. Or B, I can right click and say view code. And that's the two easiest ways. So we'll come in and we'll say OutputLabel. Text equals, I'm going to grab the current date and time, and we'll do two short time string and we'll go ahead and build that. Looks like it builds and now to run it we can simply right click on it and say view in browser. And once this runs we should simply see the time and there we have it right there. Alright, now that's the default behavior that you get out of the box. Now what happens if you would like to just create a separate file. So I'm going to add a new webform so we'll right click and say Add New Item. We'll select webform from the top. And then I'm going to go with the default name here, we'll hit Add. So in this case it just created a basic HTML type page and you'll notice that I do have the code behind though for it. Well, in this example I don't really want the code behind, we're going to say, for whatever reason. So, I'm going to delete these two files. And then I'm going to come up to the top here and delete the reference to those files. So there's two things I'll remove here. The code behind and the inherits and we'll save it. Now let me go ahead and we'll add a label. We'll do runat server because it's a server control and now I can right click and modify this. So let me cut and paste that into my clipboard and we'll come in, you'll notice when I right click it takes me to the same page. Well that's because I don't have a code behind of course, I don't have code separation so, now what I can do is use a service side script lock. Now I can put that at the top, I can put it anywhere I want but I'm going to put it down here. So we're going to say, script, we could do a type if we want. We'll say C# in this case. We can even do language, but I'm going to go with that. And then we'll say, and this is really a key, you have to put runat server in this case. I'd like to update this label so what I'm going to do is just borrow from the code that we had in about. aspx. I'm going to grab this page load. Now, page load fires when the page is ready and the, the children in the page are ready to accept instructions such as this label. So, we'll go ahead and do the same thing. Now I don't have two files. So, I can right click, View in Browser. And we should see up here on the left the time. So that's an example of doing a single file. Now, as mentioned earlier, I really don't ever do the single file approach but there certainly may be scenarios that, you know, for whatever reason, that's what you'd like to do and that's how you can do it. Now, with other versions of Visual Studio up from Visual Web Developer Express, when you add a new item, you will get this check box right in this area. And so if you did want to create a single file, you can uncheck that check box. So that's an example of how code separation works in ASP. NET Web Forms, and it really is a nice feature to help simplify your development, keep things clean and make maintenance easier into the future.

Page Directive and Attributes

Directives are an important part of ASP. NET Web Forms development. It's something you'll definitely see in the ASPX pages. So I'm going to talk about the page directive in this section and introduce some of the attributes you can use to do some pretty cool things once you know about them. So the page directive, you've already seen it. In fact we briefly touched on this in the beginning of the module, but anytime you see %@, that is a directive. So in this case we have a %@, and then the name of the directive which is Page. And then anything after that is just an attribute of the directive so very similar to an HTML tag that has attributes on it. Now as we see in this case we are identifying the language and that is one of the key things that the Page Directive does. But some other things you can do with it as well include maintain scrollbar positions. This is a pretty big deal when you have grids and maybe the user can click on the grid to select a row, and then as they do that though, it reloads the whole page based on the action they performed. Well, then they have to scroll back down and find that selected row. Can be a little bit of a pain. By you could certainly mitigate that by using some JavaScript, but using a page directive that we'll talk about, or page directive attribute I should say, we can just with, in literally a matter of five seconds take care of that problem. Page directives are also used in Visual Web Developer Express and Visual Studio in general to identify the code separation rules or the files that are used, so code file paths are identified. We can turn on some logging features, referred to as tracing. If you want to see session variables. If you want to see request variables, server variables, all those types of things, you can simply add trace equals true to the top of the page directive and you'll magically get all this information spit out into the page. Now you can also turn themes at the page level on and off. So if you'd like to apply a very specific theme which a theme really is just a kind of way the skin thinks, different CSS files typically and you want to do that for a page then you could theme equals and give it the name of the theme that you've created. Master pages in ASP. NET which is something we'll talk about later in this video course. Are also identified using attributes of the page directive. So there's something called master page file that will point a page to the overall structure of how that page should look once it's rendered at run time. So that you have the consistent header, footer, menu, things like that. Now you can even go in and identify error pages so if an error occurs in your ASPX page, you can go in and say where the processing should go from there. Now here's just a brief list of some of the different attributes. So these attributes can be substituted in where it says attribute equals. And then of course, you'd give it the appropriate value. So, you can form some asynchronous operations using ASP. NET and you could say that if this is page is on async or not. The code file, I've already mentioned. You can identify that code separation. cs or. vb file. Enable theming. Do you want theming to be turned on or off and you can even list a specific theme. It's what you'll see down here at the bottom of the list. Language we've talked about, that one's pretty simple. Specify C Sharp or VB. Tracing as I mentioned is the login. Now one of my favorites though that is just one of those super, super simple tricks that once you know about it you'll use it fairly frequently, is well, not the most short name attribute but it's called maintain scroll back position on post back. You definitely know what it does by the name of that one. As the user as I mentioned earlier clicks on a grid. If they're, they've scrolled down say half way in the page. It's a long grid. Lot of customers for instance. And let's say they select the row, then by default if that posts back, it's going to put 'em right back at the top of the page. Well, if you simply add at the top here, on the attribute equals, if you add MaintiainScrollBackPositionOnPostBack=true, that will automatically inject a little bit of JavaScript to track where that scroll bar is. And it does that automatically for you out of the box. So that's a great feature you can leverage very easily. Now, as I've already mentioned, and in the previous section demonstrated, we have this code separation concept. And the page directive simply uses this CodeFile attribute to specify what is the code file for the aspx. And that way it will point over to it and we can do some pretty cool things there to dynamically drive from code our HTML content. Very, very nice. So that's a few of the things the Page Directive does. So let me show you a little bit of a demo here on what we can do with some of these attributes. Here is an example of the page directive in a page called About. aspx. And we'll recognize it because of percent at there is the page directive and then you will notice we have several attributes that have been applied. So I am going to space these just a little bit better. So first off one of the more simple ones I didn't mention earlier is the title. We can set the title of what should be rendered up in the browser's title bar there, and that's very easy to set using the page directive. Of course, we've already talked about you can set the language, you can set the master page that determines the overall structure of how the page will render. To keep things consistent, we'll have a whole module just on that topic later in the course. This handles events and the way that page load is called, as an example. This points to our code behind, so we have our About. aspx. cs that it points to. And in that file we have about which derives from page. Well, our ASPX ultimately inherits from about, which in turn inherits from page, and that's defined by using the inherits. And you can see about there. Now, there's many others that you can use. In fact, if I just hit space right here. We'll get IntelliSense for all the different options, and there's quite a few that you can set and use. Now, one of the more interesting ones deals with the maintaining the scroll bar's position as you post back, so let me show you a demonstration of that. So I have a page here that uses the page directive as normal, nothing new than what we just saw. And it has a server control called a GridView. Now the GridView simply is going to render some customer data from a database. So if I right click and view in browser, you'll see that the customers show and then I can select a customer. Now, if I select a customer up top here, it does to a post back and you'll notice it highlights the row. So no big deal. You can't even really tell there's the post back, if you watch up here on the tab you can see it flicker just slightly though. Okay where this is kind of a hassle is let's say the customer scrolled down, this one doesn't leverage paging, so we'll select down here the big cheese. When I click on select it does highlight it, but you'll notice we're taken all the way to the top of the page. But, if I scroll down it is highlighted. It's just that it did a post back and the scroll bar position was lost in the process. Well a really nice end user feature to make your end users, kind of keep them happy is the maintain scrollbar position one. So, we're going to come up and I'm going to say maintain scrollbar position on postback equals true. Now just by adding that single line. Watch what happens here. So if we do it up top, you won't notice much because that already worked pretty well, but if we scroll all the way to the bottom. And let's go find The Big Cheese again. Select it. Look at that. It almost looks like the page didn't even reload. Like it didn't do a post back. But it did. It did do a post back. What's happening is by setting the maintain scroll back position on post back to true, the web form's infrastructures, specifically the page class, automatically outputs some little bit of Javascript. To track the x and y, with the x coordinate. If you're going this way, if you had a horizontal. Or the y coordinate if you have a vertical scroll bar. And it automatically tracks that. And so if we go into the source code here. And come on down you'll notice there's some Javascript that was injected. Here's all our grid data. And you can see that we have our vertical scroll bar position here. So, scroll position Y was 1817. Didn't have an X because I didn't have a horizontal scroll bar, but that's what was added for us, and then there's a script. One of these scripts here was used to then render that and run the JavaScript very nice. Alright now another nice feature, I'm going to close this guy and we're going to go to about. Now this doesn't have much in it but let's say I would like to go in and see some logging information about the page. I want to see the request variables. Maybe I have something called a session variable or cookies, server variables, those types of things. Well, I can go in to the page directive and add another attribute called trace and just simply say, trace equals true. Now if I View in Browser. A lot of code will be generated. So here's my regular page, but at the bottom of it you'll see that this tracing attribute that we set to true puts some information into the page. So we have some timing of how long did it take during the page life cycle. So do the different stages of the page object. What controls are in this page, shows all that information, we can get to sessions application. Cookies. Headers. All of that type of information I easily accessible through here and you can see we have server variables as well. So, this makes it very easy to get information that you know. Otherwise, in the old days, I used to have to loop through that type of stuff and write it out. But with a Trace Directive, it's a piece of cake. Now, there's certainly some others that we could add. I've talked about, you can do Fee Mean. Master Pages I've already shown and there's even more than that. But that's an introduction to what the Page Directive is. And some of the attributes that you can use in your ASP. NET websites.

Introduction to Web Controls

ASP. NET web server controls are really where it's at when it comes to ASP. NET web development. The reason for that is server controls really, really boost your productivity. If you've ever had to output a table with a bunch of TRs and TDs and put the data in the proper spots, then you'll know that it takes a little bit of effort. At a minimum, you'd have to write the loops and write out the HTML to generate those appropriate tags. Well, in this section, I'm going to introduce the concept of a Web Control. And you'll hear me say Web Control, server control, those types of things. They're all very similar. But the whole go on life with these controls is to generate HTML. Now, you can use these to collect data. You can validate data and of course display data. Really, all they are though, is classes. They have properties, and events. So they're just standard. net framework classes that you can use in your ASPX pages. Now, you can define these statically in the ASPX pages or you can even create them dynamically in your code behind or code beside pages, your code separation. Now they can generate XHTML compliant code, and they're adaptive meaning that they can target the browser based on what the browser supports. And they're very, very easy to use, once you know the general process for how they work. Now, there's really four main categories you can break down these web controls into. And all of these generate some type of HTML, and some actually output some JavaScript as well. So, the first category is the more commonly used ones, and that's called a Web Server Control. And you'll hear me just say the term server control quite a bit throughout this course. Now, this are strongly typed objects, so as I mentioned earlier this are actually classes. But they have properties that are very, very, specific to the purpose of that class. So we'll be seeing those in just a moment. Now, you can also have something called HTML Server Controls. Now, these are more rare. These are simply HTML tags such as input, type equals text for a text box. But then you'll throw the runat=server attribute on them. So Microsoft, behind the scenes, has some classes that match up with the core ACML tags. And these ACML server controls can be used to generate the ACML for those. Now, why would you want to do that over just going with the standard ACML tag? And the answer is by adding runat=server to it, and giving it an id, you can more in easily manipulated from the server side. So, it's much easier to work with. Now ASP. NET also has support for validation controls and we can do things such as. Maybe in a form, you have a text box and that text box says required. Well, there's a required field validator you can use. Maybe you have a date that has to be entered. Well, you can use a compare validator for that and set the type of it to a date comparison. You can even use the compare validator to compare two password textboxes. Oftentimes, when you register for a site, you have to enter your password twice to make sure you, you know, did it right. So Validation Controls are built in out of the box. They provide a really easy way to make sure that users input the data you want. And we'll be talking about those next, actually in this module. Now, the final type of controls that you can have are user controls. Now user controls are custom, so they're not actually something that's built in, you have to write these. So at the end of this module, I'll also have a section on how to write custom user controls and. If I had to sum it up really what they are is a way to create a reusable control that you can share across multiple pages. The most common example given is a header or a footer that you want across multiple pages. Now when you go into Visual Studio. And whether it's the full version or Web Developer Express, you're going to have the toolbox and we've seen this up to this point several times actually. And you can drag and drop the server controls onto the design surface or even do it into the HTML, the ASPX page, if you're in source view. There's a lot of different server controls available and I'm not going to read all these to you but let's look at some of the general categories here. So these first two boxes that you see in the left have some of the more common, what I'll call, core controls. Now, these particular controls that you see here, such as Image or Calendar. These are Web Server Controls, not the HTML controls, where you simply add run. net equals server. In fact all of these are the Web Server Controls. So, we have things like a Calendar. Well a Calendar, if you had to write that from scratch, which I've actually had to do that back in the day. And it's actually pretty involved to do a Calendar. The calendar that's built into ASP. NET, you can simply drag it on, it's very customizable. So you can have a small calendar, or we could even have a full page calendar where each day has different events that are listed. And that's all possible with that one control drop down lists, very common. And something that simply outputs Select and Option tags for you. That's another example of a Web Server Control. If you want to do file uploads, well, you know, on the client side it's just a pure HTML. You can always do that yourself with the input but there is a file upload control that makes it easier. kind of simplifies your life there. Now moving over to the right side, we have very robust support for data binding and grabbing data, getting into your page and outputting them. So we have something called data source control, such as the SqlDataSource control, you can see here. We also have EntityDataSource, which can be used against things like entity framework, which is one of one of Microsoft's data access technologies for getting data out of a database. We have charting capabilities so if you want to generate pie charts or bar charts you can actually do that right in ASP. NET by using one of these Web Server Controls. And then, one of my favorite controls that I use you know a fair amount is GridView control. I showed an example of this earlier when I showed maintaining the scroll back position earlier in the module. Now addition to data source controls, we have Validation controls. I'll go into those more in the next section. Moving along, we have some site Navigation controls so we can do breadcrumb trails. We can do Menu and TreeView controls. So if you want to do a breadcrumb trail, there's a site map path that allows the user to see where they are within the overall site. We also have security controls. So if I want to create a Login page, I don't even have to drag on a Login or I should say a text box for say the user name and password. I can literally just drag on a Login control and it does that for me. Or maybe I want to create a registration screen. There's a create user wizard control you can drag on. We'll also be talking about these later in the course. And the final one I'm showing here, some of the AJAX controls. So if you want it very easily without having to know any JavaScript, add the ability to do what's called partial. Page postbacks. That's the ability to have a part of the page update, not have to, during the postback, and not have to reload the entire page. Makes it very easy to work with. And here's an example of some of the syntax of these controls I've mentioned. So, as an example, a textbox, very common, use it probably almost every app I do. You'll notice we start with asp: That right there is a giveaway this is a Web Server Control. And then we always have runat=server. So whether you're doing a grid view, or a multi view, or a file upload, or whatever it may be from those ones I just showed, those will always start with asp: then the name of the control. Now the big thing with the Web Server Controls, in fact all of these controls listed here, you have to put the attribute runat=server. That is required and something that you'll need to remember to put on there. Now the HTML Server Controls. These aren't ones I use as frequently. But an example would be a hidden and you just want to easily get to the idea of that. You don't need anything fancy, there is an HTML hidden server control, webserver control you can use as well, but let's say we just want to use the standard HTML tag, but I want to, want to modify it or get to it. Easily on the server site maybe when a form is posted back as an example. Well, everything on here is pretty standard, input, type is, type is hidden, give it an ID, give it a name, but then, by adding runat server, it exposes the ID to the server site so ASP. NET the framework itself,. Will let you get to this simply by saying, hidVal., and I can get to, for instance, the value of this. So I don't have to go into any posted back variables. I can just get to it by the id, very easy to work with. The next one is Validation Controls. Now, a Validation Control really is just a Web Server Control but it's obviously specific to validation. So the Validation Controls you see here is called the required field validator. Make sure than in users have to enter some type of data, for a, for instance a text box. So we know it's a Web Server Control, has the asp:. It has the runat server, but then it has some other properties such as ControlToValidate. This would validate against the control called txtName, which is probably a text box. Now the final one is custom. These are not built into the framework. These are ones that you'll have to build. So in this case, we'd have a file. And the file would actually be called header. ascx not aspx but ascx. And then, you have to define the name space in the assembly and some other stuff or where that file is. If it's in the separate one. Well, user controls normally are just basic files that you can add in to your aspx pages or your master pages to do things like creating consistent headers. So, if you just want to define the header in one spot, there's a couple ways to do that. But, one of the ways is to create a user control or maybe you have a section of a form that's repeated across multiple pages. And you don't want to keep typing that over and over and cutting and pasting. So if you're familiar with include files, this is very analogous to what an include file is. Now that we've talked through some of the key aspects of server controls, let's take a little bit closer look. So as I mentioned earlier, every web server or HTML server control will always have runat=server, that is a requirement. Now, if it's a web server control, these are the more robust classes that Microsoft wrote that generate html then you'll always have asp: as the namespace prefix. And really, all this does is simplifies having to type having to type or prevents having to type out a longer name space. System. web. ui dot then you get into web controls. You can give these IDs so we can programmatically get to them or even on the client side, we can get to these. And then in this case, because it's a label control, we have a text property. Now, that is a strongly type property. It's a string. Where you can say what is the text of the label. Now oftentimes, if you know the text and it doesn't change, you don't need to use the label, just type it into your aspx page like you would an HTML page. So normally when I use label controls is when I need to dynamically output some text to a page. I'll throw in a label. And then, I'll give it an ID. Then, in my code behind or code beside page, my code, I'll write some code to get to, in this case, lblHello, label hello, dot text, and I can assign the value dynamically. Another one is the grid view that I mentioned, so you'll see asp:GridView, we know it's a web server control has an id, has runat server. That's kind of a broken record, something you'll always see with server controls. Now, the case of a grid view, it's a very robust control. It has a lot of properties. A lot of things you can do with it including templates, for the data that you're going to bind into it. So you'll see, we can set things like the border color, cell padding, cell spacing. This outputs a table by default and we can use these properties to control what's being written out in that table. So very very easy to control, easy control to work with, and something that, I use fairly frequently. Now, the final thing we'll talk about with web server controls is the ability to hook up events. This is really really key in why, other than the fact that you have web controls, which output HTML for you. Why ASP. NET is so popular. When you work with these Web Server Controls, many of them have a rich set of events you can tie into. Now some might only have one or two. But others like the grid view control have quite a few events that can fire, if you're new to events. An event is simply a notification that something happened. So, the most simple one is when a button is clicked. How do you know a user clicked a button in the browser on the server side. Well, what happens is ASP. NET uses standard web technology. So we have a form tag. Now, were going to throw in a web server control which is a button. So in this example, you'll see we have a button here, give it an ID run a server, very similar to what we've covered. Now the text to the buttons going to be submit, but when this button is clicked, it's going to cause the form to be posted back. What ASP. NET will do is look at that ID of the button. And if it's wired up on the server side to an event handler, then that event handler will be called. And that's all done automatically, you don't have to check for the presence of a button in the request. form or on the query string or whatever you're doing, it just automatically handles this for you. So, in this case you'll see that we're linking the event handler which is called btnSubmit\_Click, to the actual button through this OnClick=, and then we give it the name. Now when that button is clicked, the form posts back. ASP. NET will then automatically invoke btnSubmit\_Click. It'll pass in the sender, which would be the object, and there won't be any event args in the case of a button. And then in this case we're going to update some label called lblMessage. Text to you clicked the btnSubmit button. So events are extremely powerful because it's almost like doing desktop programming. You can literally in Visual Studio double-click on a button, handle that click event in the event handler and then write the code that you want to run when that event handler's invoked. So, let's take a look at how we can use the different types of controls that are available in ASP. NET. Take a look at a few of those and how we can handle events as well. Let me demonstrate how you can get started using web server controls in your ASP. NET applications. So, I have Visual Web Developer Express open and we're going to create a new project. Now, I'm going to create an empty web application in this example, and let's go ahead and call this UsingWebControls. And we'll go ahead and save that. Now that'll generate our project. And because I did an empty one, we don't have anything. So I'm going to right-click, and we'll do Add, New Item. We'll do a Web Form, and I'll rename this to Default. aspx. Okay so you can see we get our standard code. And we're ready to go here, so we have a form tag first off. You'll notice this has runat=server. Now that is an HTML server control, and you do need to leave the runat=server there, that's an important part of the overall framework. So, let's say that we would like to design a simple form to capture, maybe a user's first name, last name and the state they're from. So to do that I'm going to switch over to Design view to speed up some of the process here. Let's do a table real quick. And, we'll do a couple rows. So first thing we do is we need first name, last name and state, and then we also need an area for the button, to be able to click it. So, we'll do four rows, and we'll just do the two columns here. All right, so that'll give me a real quick table to start working with. And I'm going to make this a little bit smaller. So I'm just going to type in First Name. We'll do Last Name, State. And then we'll have a place for the button down in here somewhere. Let me move that over a little. Okay, so now we'll come to the Toolbox. And I'm just going to go to the standard controls that you see here on the left. And I'm going to drag in a text box for here. I'll drag in another one right there. I'm going to put a dropdown list for the state. Let's come down to the drop, up to the drop down list, actually. And, we'll then add a button. And then when this button is clicked, I'm just going to hit Enter a few times, and I'm going to throw a label in here. We'll write out what the user actually typed. We'll echo it back to the page. So, I'm going to come to the Properties of that label. We'll take out the text, but I'm going to give it an ID of L, let's do actually OutputLabel. Let's work through these controls. So this is a Web Server Control. These four controls that you see here, and then label is as well. So I'm going to come up to the text box. The first thing I'm going to do is give it a better name. So let's call this FirstNameTextBox to be pretty obvious about what it does. This will be LastName. For the states we'll call this StatesDropDown And for the button, don't really have to do much with it, but we'll just call it, SubmitButton. And I'm going to change the text of it to Submit. Now everything I've just done is modifying the properties of these controls, and remember these are just classes, it's just that they're specialized type of web classes that can generate HTML for us. So if we go to Source view, you'll see that the aps:TextBox was used for our text boxes. There's that runat=server. Here's the IDs I just did. Here's my drop down list, we're going to come back to that, add a couple states. And then finally we have the button and the label. So the nice thing about this is, if I run it as is, let's go ahead and we'll do a build here. And incidentally, you actually don't have to build, if you only change the aspx, if you're using a web application project. The code in here will automatically be picked up, so I could just right-click View in Browser. I like to build often, though. Make sure I don't have any issues. All right, and that's what we get out of the boss. Now if we look at the source. We can come down, and you'll see that it output the appropriate type of HTML control, tag. So we have an input, type equals text, input type equals text, here's our select. We don't have any options yet, we'll get to that. Here's our button. And here's our label, which is a span tag. And then, wrapping all of that up here at the top, we have a standard form tag that redirects back to the action, will go back to the same page itself. Alright, so let's go back to the drop down list now and I'll do this visually because it's pretty easy, and I'm going to click on the drop down list. And I could do one of two things to add some hard coded options. I could right-click and go to Properties, and then we can come down into the items and click this little ellipsis button, that will let me add in items. Now that's one way to do it. The easiest way to get to it though, is to highlight it. Click the little smart tag, anytime there's a smart tag take a look what that does because, notice we can do Edit Items right here. That takes me to the same screen, so it saves me a few clicks, if I go through the smart text. So I'm going to hit Add. And let's just add some text and some values. So, we'll do Arizona. Let's do, California. And we'll do New York. All right now, maybe above that I want a Select One, to make them select something. But, I don't want a value on that we'll say. I wanted to say select one, but I don't want to value. Then I'm going to hit these little arrows, to move Select One up to the top. And so there we have our hard coded states, and I simply hit OK. Now if we go back to the Source view, you'll see that it added something called a list item. Now a list item actually in this case will translate to an option tag in HTML. But what's nice about it is I didn't have to really type a whole lot except for the value. So, I'll View in Browser and now you'll notice I have my Select One with my options. And if we View Source, as mentioned earlier the whole goal in life of a web server control is to output HTML. So you can see it output the select tag, there's my ID, and my options. You'll even notice that, right now the value of Select One is not quite what I wanted so, we're going to come back and fix that. So, we're going to come in and for the, value here, what you can do is actually come in and say value=. And I'm going to force it to an empty string so that if we do validation, we can check, did they type any or select anything or not. So let's View in Browser, let's View Source. And now you'll notice that it does have an explicit value of empty string here because of what I went in and added. Now, real quick while I'm on that. I could of also come in and saved a little bit, although I'd have to type this manually. And we could also say Text=SelectOne. Then we could do a shortcut close tag on that. Now, you'll notice I don't have runat server on this and that's because the parent is already running at the server, so you don't have to put it on these particular child tags. Now the tag I have here is the same as really one of these. If you just want to put out Arizona, there's no need to put, value=Arizona and text=Arizona. You can actually save yourself a little bit. But in cases where you want to take over control and you're going to be typing it, this is what I'll usually do in this case. Okay so running it again, we can now type into the text boxes. We can select a value. But the button doesn't do anything here. All right, I'm going to hit No for Autocomplete. It just posts back but, nothing's really happening. So to fix that, let's talk about Events. Now, there's a couple of ways to handle Events in ASP. NET. One way is, in the case of a button click, I can simply double-click on it, and that will take me to the code beside or code behind file and give me an event handler out of the box, it's very easy. But another thing that is nice to know about is if I go to the button, you'll notice over here in the Properties, I can come in and change any property of that button I want. But, there's a little lightning bolt, right here at the top. So if I click on that, you'll notice there's a couple different items I can click on here. So, Data Binding, Dispose. Now, for a button you'll generally not use these others. Maybe, but generally not. The one we'd want is Click. So if I want to add a event handler for the click, I can double-click on it here, that's one option, but the easiest way for a button is to simply double-click on it right in the Designer, that makes my event handler. So, the reason I want to point that out though, going back to the Properties window with the lightning bolt here, is while the button only has a click that's very useful, there's going to be other controls, such as the Grid view, Details view, and others that they have a lot of events. So you can't just double-click on them in a Designer. You'll typically go to the Properties, click on the lightning bolt, and then you'll select the event, and just double-click. In fact, if I double-click on it, you'll notice right now it's directing me to that event handler. If I double-click on it here, it'll actually take me right back to it, which is very nice. Okay, so now what we want to do is we want to take this, this and this. And when they click the button, handle that callback when it's posted. And, update the label with those values. So let's go ahead and do that. The way we'll do that is we'll come into here. And now we need to grab the text values. So, I'm going to go ahead, and we'll just create some basic values for, First Name. And we're going to say that First Name is from the text box. And we have FirstNameTextBox. Text. So I can go to the ID of the web server control, and then go to the property to programatically get to the value that was posted back. Again, I don't have to go into the form collection, if you're used to doing it that way. So let me go ahead and type these out. So we'll do lname is LastNameTextBox. Text. And then let's grab the State. Now the way we do that is I need to go to the drop down. So we can go to StatesDropDown.,... and I have a couple selected options here. I can do SelectedItem, we don't want that in this case, but I can also do SelectedValue. Alright, now the value will be, literally, that behind the scenes value associated with a select tag under the options. So we'll go ahead and say SelectedValue. Now we want to write all these out to our label. So, we can simply say OutputLabel. Text =, and I'll just concatenate them and it won't be too pretty, and we'll just add some spaces between them. Alright, so you can see with a very minimal amount of code I am able to capture the data from the form, store it in some variables, and then write those out to our label. Now if this were a database driven app, we'd probably send this along its way and we'd update the database. We'll be talking about data access technologies in this course a little bit later. So let's go ahead and do a build now because we have modified code at this point. Looks like it builds okay. So, now we can right-click on it to view it in the browser. All right, so we'll go ahead and enter Name. Enter the State. Hit Submit, and you'll notice that it automatically writes that out. Now if we analyze, you know, what's been done here depending on what framework you might have done web development in. Or maybe you haven't and you're brand new to this, have you done desktop development. What I just did here is pretty close to desktop development. I can visually design my interface, I can double-click on things to create events. And then I can get to my controls by simply typing the ID of the control, and then get to the control properties. Really, really nice. I know when I first moved from classic ASP into ASP. NET way back in the day, this pretty much blew me away because we couldn't do it this easy. We had to go into the request variables and get these, variable data out. Very, very easy to work with. So, that's an introduction to working with server controls. Working with properties, defining them within the ASPX page and then also handling events so that we can write out the data from the server controls. In addition to the core controls that ASP. NET web forums provides, there's also a set of more rich controls that can output lots of different types of HTML. Now one of those is the Grid View control. A very commonly used control that can output tables. Now it's located in the Data section and we'll be revisiting that particular control as we get to the data access modules in this course. But I want to show how easy it is to get started with the control. And show that you really have to write very very little code that its very productive to use. So to demonstrate it, let me go ahead and right-click on our project and we'll add a new item. And I'm just going to call this grid view demo. And we'll add that in. Now from here I'll switch over to DesignView. And then drag on the GridView, drop it on our page and we get a smart tag that pops up. Now again, a smart tag allows you to modify typically properties or other settings of a control. But what's nice about this one is I can actually go and modify the format of it, how it looks. So there's a lot of different built in styles and you could certainly go beyond this. This is just some starter ones but you can customize it anyway you'd like. So we'll hit OK there. Now just like the text box and button, I can go to Properties. And this control happens to have a pretty rich set of properties, it also has a rich set of events and those are quite a few here. But were going to come into the ID and I'll call it CustomersGridView. All right, now let's say that when this particular page loads. I want to go ahead and grab some names and bind it into this particular item, this GridView. So I'll right-click and go to View Code. And in our Page Load event, once that page loads and that means the children, such as the GridView, are ready to be acted upon to add some dynamicness to them. We can come in and create a collection of names and bind it. So I'm just going to create a very simple one here. We're going to use a list of string,. And I'm going to do a simple for loop. So we'll do the code snippet for that and do for, tab have that pop it in, we'll do 99, and I'm just going to say names. add. And we'll add in John Doe plus i's value, just to give it some differentiation between names. Okay, now that'll create the loop for me, but we obviously need to get that data into our control in the page so it renders. Right now, if we go back to the page, we literally have a GridView and if we look at the source for this. You can see it put out a bit of code for us. So all these different styles are being set based on that earlier style format I picked. But you know, we don't have anything as far as data that this is going to render and if we run it right now you actually won't see anything. So we'll come back into our code behind and now go to Customers, GridView. And we're going to assign to what's called the DataSource property. Now the data source property takes an IM numerable collection. And you know we can iterate through, such as names. And that alone won't be enough though. We need to actually call a method called data bind, for it to actually kick in, and render the HTML back to the browsers. So we'll say CustomersGridView. DataBind. But that's it. It literally takes two lines of code, to assign data into this control then have it do its thing. So, let me go ahead and build this. Looks like we're good there. So, now we can right click, view in browser, and we should just get a one-column grid. Now, if we go to the source for this. You'll see that it allows us to really do a whole bunch of work, with just a minimal amount of effort on our part. So it's generating all the table tags that we need to output this particular table. There's the start of it right there. And you could see, it's also applying some CSS styles here. So that's an example of another control, an items control that we can use in ASP. NET, and how easy it is to get started with it. Now we could just as easily apply these names that we created to other types of controls as well. So let me come into here. And I'll just comment this code out here. And we're going to do a server side comment. All right, so a server side comment in ASP. NET looks like that. You can put the percent followed by two dashes and then dash dash percent on the bottom and that'll stop it. And so I'll switch to design view now and let's look at some other types of controls that we can add. So in this case, I'm going to go on up and we'll go to the bulleted list. And so as the name kind of suggests, this bulleted list simply spits out a list of bullets. And we can hard code those by clicking Edit Items and this would be much like using the drop down list we did earlier. But we can also bind to it. So let's go to Properties, and we'll change this to Customers List, we'll call it. And we'll go back into the code. Comment out this section right here. You'll notice in this case I'm going to come up to the comment area, comment that out and we can also come in and do the same exact thing. Then we'll call data bind. And that's it. It'll do the same exact thing you saw before, but this time it'll be a bolded list. So let's go ahead and run it. And you can see we get our lists. If we go back into our source here, you can see it did a standard, un-ordered list that it added for us. So it does a lot of work for us. I didn't even have to write the looping code to take care of that. So there's many others we could do. There's a control called repeater, there's a ListView and others that are typically used in data access scenarios. So we'll talk about some of those later in this course.

Validating User Input

A common task required in ASP. NET Web Forms applications is the ability to validate user input. But fortunately that's a fairly easy process to go through when you use Web Server controls. So in this section. I'm going to explain how we can do that and what the different controls are. So validation obviously can be done in two ways. We can validate on the client side. But we also need to validate on the server side. There's nothing to stop a hacker from stripping out JavaScript code to bypass the validations, so we need to do it in two places. Fortunately, ASP. NET validation controls, which are just web server controls, make this a very very simple process and they validate in both places. They'll output the necessary JavaScript if you'd like to use. And you can turn that on or off and then they also validate on the server side. So that way you're covered and you don't have to write a lot of custom JavaScript. Now, having worked with these controls for many years, they do cover all of what I'll call the core validation activities. But you'll certainly in more robust applications end up writing some custom code as well. To handle situations where you have to validate maybe many controls against different values. But to get started these are a great way to go. And they'll cover a large percentage of the validation that you typically have to do on an everyday basis. So some examples are shown here. Required fields. You know, that's not something I really personally enjoy writing JavaScript for. Well that can be done in a matter of seconds by using the web server control. We can validate ranges of values very easily. We can even compare control value. So if for instance, you have password one and password two, in the user registration screen, then we can use a compare validator. And the compare validator can also be used to validate data types, such as integers, or that they enter the valid date and those types of things. Pattern matching is available through regular expression. So for instance you had an email address and you want to make sure they entered it and it follows the standard email pattern. Then you could use a regular expression validator. And then if none of that meets your needs, you can even write, as I mentioned earlier custom JavaScript. And use a control to invoke that custom JavaScript to make it easier to run it. Now here's the different controls that are available out of the box. So you have a RequiredFieldValidator, a RangeValidator which can be used to validate a range of values. Compare validator, I already talked about briefly. Now this one is kind of multi-purpose though. As I mentioned it an control validation against multiple controls such as password one password two. And that's where normally you'll see the demonstrations of using it, but it can also be used to validate data types. Integers, decimals and dates and things like that. Regular expression validator can be used for any pattern matching so, maybe you have to enter an IP address into a form text box. Well, as long as you know the regular expression for that pattern, you could certainly do that, or good example be a zip code. That maybe want a zip plus four, and you would like them to enter that appropriately. Custom validaters when you want to call out to custom Java Script. And then finally the validation summary ties it all together, each of these validators can have a custom error message associated with them. So the validation summary allows you to consolidate all of those error messages into one place. So, let's take a look at how we can use these controls to do some validation quick and simple in an ASP. NET Web Forms form. It's quite easy to add validation logic in to your asp. net web forms applications using these web server controls that specialize in validation. So in this demonstration, I'm going to show how we can validate against this form. So I've added one additional field called Birthday, so I want to demonstrate a particular concept here. To what I had done earlier in this module with web server controls and obviously you can type your birthday in here. So if I run the form now and we View in Browser we can fill things in and some we don't have to fill in, hit Submit and it writes out what I wanted. But, that's pretty dangerous, because number one, the user doesn't have to enter certain fields. Let's say all of these are required plus you want to make sure that the Birthday is a valid date. Now normally, you'd have to whip out some JavaScript. And when they hit the button you'd invoke a JavaScript function that would take care of calling your different text boxes and things. Saying if it's valid and then submitting the form. But then you'd have to write the code on the server side to also make sure that it's all valid from that perspective. Because it'd be really easy to as a hacker, strip out the JavaScript, load the form up in your browser and then bypass the checks for validation. So what we're going to do is do it in both places. We'll do client side validation and we're going to do server side validation. It's going to be very simple, you'll see. So fist thing I'm going to do is over here in the toolbox. We have our standard controls, our data, and then under that we have the validation controls, you see, right here. And I'm going to drag on required field validators for all of these. So, bear with me one moment, we'll drag these on. I'm just going to cut and paste these down. And now what I need to do, is I need to hook each of these controls, these validater controls to the appropriate control. And the way you do that is by going into the properties. So if I just right click and go to Properties. We can go in and change, first off, the error message. So let's say, please enter first name and then we can also set the control to validate. So I'm going to select that, and you don't even have to type it, you can see. It's pretty awesome. You can just go in and select it, so. Since these have some pretty descriptive names, I'll select FirstNameTextbox. And the, we'll just repeat this process for all of them. So we'll come in, please enter last name, and then we'll set the ControlToValidate to LastName. Likewise, we'll do birthday,. And you can see this part here is pretty simple. Very repetitive actually. Actually, let's say, please select a state. And you can even use this against dropdowns or selects when the HTML is rendered. Okay so, just by doing that alone, I've now made all those four items required. So let's try it out. We'll View in Browser here. Hit submit and you'll notice it didn't do anything. We didn't even post back the tab up here you can't see anything reloading. And that's because it's not. It's using client side validation. Now the reason that's working, is out of the box, these validation controls. They have a property here called EnableClientScript, right there. And that's set to true by default because in most applications, that's what you'd want. You want to give the user that. Friendly error message, not make them reload the whole page. So that's set by default to make it easy to work with and I didn't even have to do anything fancy there. And what's nice about it is, as I fill in the blanks, I'm going to hit Submit. We'll go ahead and now start to fill it in. As I tab off and lose focus. The JavaScript even removes that particular error message. Then we'll do a, that's when I wish I was born, I'd be a lot younger that means. But so now if I hit Submit can't do anything, but it actually removes the errors as I type and that's really nice. Now having the errors next to the different form elements is nice, but on some of the larger forms, that can get a little bit messy. So oftentimes you'll see a little asterisk appear next to it. So let's go ahead and do that instead. And what I'm going to do is, if there's an error then right above the button, so I'm going to highlight the button. I'm going to hit the left arrow key to get right on the left side of it, and then hit Enter a couple of times, I'm going to drag up a validation summary. Now what a validation summary control does is, allows us to automatically display a nice list of what are the errors? All in one place, makes it really easy for the user to see what's going on. But I don't really want to show the errors here and here. If I run it as is and we hit Submit, you'll see that we kind of duplicate our efforts. We see the errors. The user definitely can't miss it, I'll, I'll say that. But maybe we want an asterisk here. So the way you do that is leave the error message. We want that to display in the validation summary. But what we want to do is come down and change the text to star and that will override it locally. So we'll go ahead and do that for all of these. Okay, so now if we run it the asterisk will show if there's an error and then go away if there's any problems, but it also shows down here the validation summary. Very nice, very easy, you know I've yet to write a line of any code actually. So, now as I type you'll notice that first name, the asterisk will go away. Now it doesn't go away here, until you invoke the button. Now you'll notice it does. And that's just kind of how it works out of the box. And we'll get it all working, and now we can post back. Okay. Now that works and we've done some pretty good validation on the client side. But, we really haven't accounted for the Birthday. So let's go in and do that. What we can use here is what's called a compare validator. Now, compare validator often times when you see this used, it's used to actually compare two types of controls to each other. And these controls would be like password one, password two, maybe you're filling out a registration screen, something like that. But in this case, we can actually use the compare validator to validate against things like currencies, sates, integers, things like that. So I'm going to drag on a compare validator. We'll go through the same process here. We'll go in and we're going to set the error message. Please enter. Enter a valid date. And then if I come on down, we're going to come into the type of the comparison. So right now by default it's always a string comparison, by default. But if I hit the drop down I can compare it to I want it to be an integer. They can't type alphabetic characters. I want it to be a double which is a type of decimal, a currency which is a decimal. Or in this case, obviously we want a date, that's what we want to work on. Now, the type of comparison we want to do as a date but we don't want to see if it's equal to something. That would be for comparing two passwords, we want to come down and do a data type check. Now let's talk through these real quick though. If you're doing integers, you could do greater than, greater than or equal, less than, less than and equal, that type of thing. Or equal if you're comparing two things. That's really nice. But if you're doing a data type check, you need to select that as the operator. Makes it very easy to now validate against a date. And if you've ever validated dates in JavaScript, it's actually a little bit trickier than you might think. Now we'll come up to control the validate. Now there's two options here control the compare and control the validate. Now you don't use both in this scenario, in fact in this scenario we're just going to pick one of them cause that's all we have. But again if you're doing things like passwords comparing two of them you would select both types of the password controls here. Now if I highlight these two properties, you'll see controller compare down here, ID of the controller compare with. Now we're not comparing with in this sample. Controller validate, ID of the controller validate. Well that's what we want. So we're going to select, birthday textbox. Alright, so at this point, we've now gone through and we've made this particular, birthday text box required. But we've also done a second data type check or second validation for is it a valid date? Now, I'm going to do the same thing with the text. And we'll come down and just put a star there. Make sure it's not messed up or anything. All right, so let's go ahead and try to run this now. So, I'll go ahead and just hit Submit. And you could see at this point, because we haven't typed anything, the date validation hasn't kicked in yet. Let's go ahead and fill things in. To the date I wish I was born nowadays. And it looks like it's good. Now let's do something invalid. Let's do 2/29/2011. We'll mouse off and you'll notice that we get the error. Let's hit submit and we also get it in the validation summary. Very, very simple. Super, super easy to use. So let's go ahead and clean this up. Now we're down to one. And let's just go ahead and fix this to 28. Hit Submit and we're in business. So, we've now done all of this client side validation but I've yet to write a single line of JavaScript. Now certainly there's going to be situations where these types of control won't cover all the validation you need. And although I won't show it in this demo, I'll point it out, there is a custom validator, and really what this is used for is to make custom calls out to a JavaScript function that you might write that does some more advanced type of validation. Okay, so we are doing good so far, we have some validation, we didn't have to write much code, very simple maintenance, we've done a validation summary to make it easy for the user to see the errors. But, we got a big problem here. If I turned off client side validation, so, I'm not going to do this in this example, but if I came to these controls. And turned enable client side script to false to all of those, then no JavaScript would be emitted and when they hit submit, it would just post it back. Now, the other problem here is even if we do have enable client side validation it's pretty easy for a hacker to save the HTML locally strip out the JavaScript that does the validation and then submit the form. So they would bypass your validation and potentially you could get some junk in your database or wherever else the data goes. So we also need to be careful to add validation into the server side now that's very simple to do. All we have to do is call page validate so I'm going to say page. validate. And that will then iterate through all of the validation controls, the web server controls that specialize in validation. And say, hey are you valid? Are you valid? Well if one of those returns false then the whole set is not allowed and so what we do is say if page is valid then we want to go ahead and do our thing here. Now if the page isn't valid, and even if client-side script is turned off, those error messages that you saw earlier will be written out back to the page. So it'll do a full post-back. But it'll actually regenerate the form. Leave the data they typed by the way that'll still be maintained and then the error messages the asterisk and the validation summary will be displayed form so it's really really nice super simple. Alright so now if we run it let's go ahead and a just do a build here. We can view in browser and I'll just fill this in really quickly. And you'll see it validates. Now just to prove it's actually working though. Let's come back into here. We'll set a break point but make sure this is set as my start page. So, I'm going to right click on default ASPX and set as start page. And then, we're going to run this in the debugger. Okay, so now we're in debug mode. So let's go ahead and do Dan Waleen. I'll just put in a bogus date here. Enter a value and hit Submit. Now you can see it did call back. We'll F11 to step down to the next item and if we mouse over is valid you can see that it's true. Obviously that's the case. It's a valid date and everything that was required has been entered. So now I can come down and I'll step in and then it just goes from there. And so I'll just go ahead and step out. And it'll render our, our page and we're ready to go. So that's an example of how easy it is to get started using the ASP. NET Web Forms validation controls. Really, I think it's just a huge productivity boost especially if you've been writing a lot of custom JavaScript and then duplicating those efforts on the server side. So, there's other controls we could use if we had an email address. We can come in and let me just show that one to finish up here. So, we'll just simply type email. I'll drag a text box on. Let me go ahead and give that a decent ID. We'll say, EmailTextBox and then we'll do the same type of process. But in this case I'm going to drag a regular expression validator. Now we'll say email's not required, you've already seen how to do that. So, we'll come into the properties, we'll say inner a valid e-mail. We'll come up to the controller validate again. So it's a very repetitive process, you can see here. But then what I need to do is what is my validation expression? So, if you've never done regular expressions before. They're not exactly the most user friendly to type So fortunately, Microsoft, if we go to this ValidationExpression property, we can hit the ellipsis and they have some built in regular expressions that we can select. And so, we can come in and for instance do an internet email address. There's a ValidationExpression to get you started, and of course you can tweak it. Now if you've done regular expressions before then you've probably seen this type of thing, if you haven't don't be too worried, they're not something you have to master in this case. But there's a lot of good websites out there that'll help you out if you do want to create custom regular expressions. So we'll go ahead and hit OK. Now let me do the same thing as before. We'll just simply put an asterisk and we'll save it and run it. All right so we'll fill in the blanks Now I'm going to go ahead and just type test here, hit submit and no go. It's not a valid email address pattern. To be a valid pattern, it's going to have to be test at some domain. Test. com as an example. So I can even put a space there, hit submit, doesn't like it, so we'll go ahead and put the at back in. And you can see now we're valid. And we submitted the form back and updated the data. So, that's an example of yet another control you can use that's very, very powerful. Now the RangeValidator, which I'm not going to show here, can be used to validated a range of, for instance, numbers. The Compare Validator I've already mentioned, can be used to validate multiple passwords, or just any control where they have to type something twice that matches. And then we've seen the others. So that's a wrap up on using custom validation without writing custom code, though. I have custom error messages. I do custom server side validation, but I didn't have to write any of that custom code. It's a beautiful thing, very productive and that's what you can do with validation controls.

Setting Defaults

Let's take a look at how we can set some defaults in ASP. net web forms applications. So the things I'm going to show you here are very, very simple to implement. But they kind of satisfy or help out some pet peeves I have. One of the things that kind of drives me crazy is when you go to, say, a login screen and you have to actually click in the first text box, such as user name to input it. In my opinion, that should always be the default focus and we should be able to set that. Likewise when you hit enter when you go to login. Sometimes that doesn't seem to do anything, it just sits there. So we're going to talk about how we can satisfy both of those very easily in ASP. NET applications and you don't even have to write any fancy C# or VB, you can do all this in the ASPX file. So the first thing up is setting a default button. So if we would like to have a button on a form, a login screen, whatever it may be set as the default. So that when the user hits Enter, that button gets triggered. Then all we have to do is on the form tag, simply add this defaultButton equals, and then give it the ID. Now the ID would come from the ID you define on the button server control. So, not a client side ID. You'd actually put the ID of the control itself. And so just by adding that it automatically will make it so that you when, when you hit Enter it will submit that particular form. Very nice. Very easy. You don't have to handle, you know, key press or anything like that events, using JavaScript. Now, there's a panel control in ASP. NET that can also be used for this. There may be cases where you have different buttons on one page. And, we'll say maybe you have two forms. One is to fill out a newsletter and that's just a very small one off to the right. And then in the center of the page you have a real form, a bigger one that you want to fill out. Well, if the focus is on the newsletter form, which maybe it just has you enter your e-mail address to subscribe to the newsletter. Then on the panel control, and a panel control is just a container control where we can put child controls. Then we can add defaultButton equals and give it that button. Now if that has focus and they hit Enter, it will work just for that form. Versus if the main form has focus such as the one up top here. Then that form would be submitted. So, very easy and that's all you have to do to get started with it. So that's the default button. Now I can get around that. I don't, I don't necessarily mind clicking a button, although I think the default button is so easy to add that we should all be doing it. But setting the focus is one thing that should just happen. When you go to a form, you should be ready to go as far as typing, into that first form element, at least in my opinion. Well, we can go in and do that in a very similar way, by doing defaultFocus equals and give it that text box that we'd like to set focus to. And that's all you have to do. Don't have to write any java script to set the focus yourself. Now, there may be times when you want to programmatically set the focus. You don't know for instance, what the idea of that first control is. Maybe it's dynamically generated. So you need to do this through code. Well that can be done in a couple ways. First off, the page class has a set focus method and if you already have the control in memory. Then you could just give it to control and it will set the focus to it. Second if you don't have the actual control instance but you do you know the control ID. Then you can call an overload of set focus and pass it as the client ID. The final one is if you do have a reference to the control itself you can also call. focus on it. And that will set the cursor blinking once that page loads in the browser. So let's take a look at implementing that in a form. And see how these default focus and default button behaviors kind of work. Earlier in this module, we built a simple form to demonstrate web server controls and also validation controls. So here's the form. And right now, if we run it, there won't be a default focus set when it first loads. Now, because there's only one button in this scenario, we're pretty good on the button itself. But I'm going to go ahead and add a default button as well, just to show how easy it is to implement this. So if we view it right now in the browser, you'll see there's no focus set. So I'd have to grab the mouse, come into here and click. And while that's not a huge deal, it definitely does slow down the workflow as compared to if it was already there I can start typing. So let's go ahead and fill this in. And if I hit Enter right here, because the focus is on the state, you see nothing happens. But if I come up, you can see I can hit Enter and it does submit it. Now, if we had multiple buttons though, that would not necessarily be the behavior. So let's go ahead and account for not only the focus, but also the default button. So we'll come back to the source. I'm going to come up to the form tag and we get intellesense as we type this. So I'm going to type default. And notice I can set default button. So our button is called submit button. And again that is the ID on the actual button control. So that's the ID you would always need to give here. You can also command set default focus. And we'll set the default focus to first name text box. And now when we run it, we can go view in browser, notice the focus is now on first name. Great. I could just tab through as a user. Especially, some users don't use the mouse that much they're pretty keyboard-centric. It's pretty easy to work with. Now, I just hit Enter while I was on the drop down here and because, I set the default focus, this now works, whereas when I didn't have that before, it doesn't. Let me go ahead and take that off real quick and demonstrate it. And now as we go in, I'll have to enter my, the name again, we'll just put some bogus information real quick. We'll do test@test. com, and let me fix that guy first. And then I'll come on to Arizona, now I'm going to hit submit, and you'll notice it doesn't work Okay so I only have one button, if you're in a text box I can get this to submit. But if I'm not it doesn't work in some scenarios. So I tend to just add it always. I think it's just a good thing to do. So we'll go ahead and add this back onto our form here. Now that's how easy it is to add those and I didn't have to write any custom JavaScript or even drag on custom server controls in this example. Now if we wanted to programmatically set say the focus then we can do that as well. Let's say for whatever reason we want last name to have the focus. In this case that probably wouldn't make a lot of sense, but we can come up to our page load. And I could do page. set focus. And this has two overloads, I can give it the control or I can give it the client ID. So, we could do either, I could give it last name text box or in quotes, if I didn't have an instance of that available, because it was dynamically created, but I did know the ID, I could do that as well. Now, in this case, since we have the control, it's generally easier just to call name of the control, focus. And now when we build and run this, it'll actually cause that particular control to be focused. So, we'll set our build here. Run it. And you'll notice that last name now has the focus. So although those are some very simple techniques you can use, they're definitely powerful, and something that I highly recommend everybody should use. Especially on forms where you want to, you know, let the user hit the ground running. Let them type right away. And be able to hit Enter. Since a lot of users are used to being able to hit Enter. So that's how easy it is to set defaults with ASP. NET Web Forms.

Creating User Controls

In this final session, we're going to talk about how to create User Controls, and how to define and use those user controls within pages. So a User Control is simply a piece of commonly used user interface, code and the controls in HTML that might go with it. That you might want to reuse and share across multiple pages in a given website. Now, you can register these controls within a page and then, it'll pick those up at runtime. So if you've ever done an include file with other frameworks, where you have a file you want to include dynamically at runtime. That would be very analogous to a User Control. Now, examples of this include login screens that might go across multiple pages. Headers or footers that you just want to define that once, and then that way you have one place to go to update, for instance, something as simple as the copyright. Or maybe you have menus and the menu you would also like to share across multiple pages. Now, the way that these work is we use a new directive that we haven't talked about up to this point. We did talk about the page directive and how that works. Well, with User Controls, it's not a full page, it's a User Control. In fact, it's even going to inherit from a class called User Control. So what we'll do is in our pages, we'll add in this percent at, which is a directive and then say hey, this is going to be a control. And then we'll give it the language. Now, the file extension that you'll create is an ASCX. So up to this point, everything we've done to this point has been ASPX. But with user controls, you can do. ascx. Then of course, you'll have to reference that. ascx file within your. aspx, your web form files. And then work with those. So let's take a look at an example of creating and then defining and importing user controls into our web forms pages. Let's assume that we'd like to create a header or a footer and add it into our existing web pages, but we don't want to cut and paste and duplicate that code across the pages. Well, fortunately, that's very very easy to do with ASP. NET using user controls. So, to create a user control, what I normally like to do is create a separate folder and I'll just call it UserControls. And you certainly don't have to do that, but I think it makes it easier to find these and keep them separate from your full pages, your ASPX pages. So, I'm going to right-click. We'll say Add, New Item. And you'll see that we can do a Web User Control. You'll notice the extension down here is an ASCX for control. So let's call, call this header. ascx. Now from here on out it's very much like designing a regular standard web form. We can drag and drop, we can use server controls, we can do all that magic and then when we're done, we can reference this control in our ASPX pages. So to start off though, let's look at the control directives. So you can see the language is set. Some events are wired up, for instance page load. Here's the code behind, so we do have code separation. Here's our code file, right there. It looks very similar. The main difference being we don't inherit from page, we actually inherit from user control. And then we have the inherits which says the header class should be inherited from. And that can be found right there. So now, whatever we put in this control will be encapsulated and we can now reuse it across this particular web application. So, let's assume that we just want to do a little header. We'll put something like acme corp. And maybe I want to go ahead and wrap that with a div. We'll just add some inline styles to keep it quick. All right, we can go to Design View, everything looks good there. And let me just space over just a little bit. Now, let's say we also want to put the current date over here as well. So, we can drag on a label control. And we'd probably want to align it right or something, but I'm going to go ahead and just add the code. We'll come into here and it's because I didn't give it a name, it'll be something like label one. text equals and we'll do date time. now. To long date string. Now, that will automatically update that label just like we've been doing throughout the different modules up to this point. Just like you would do on a page as well. So we can come in and I'm going to cheat and do it this way instead of doing a float, just to get it to work. Now you'll notice I can't right click in view and browser though. Likewise I can't come over here and right click in view in browser. The reason for that, is it's only part of a page, it's not the whole page. So, as a result, we can't just right click and run it. There isn't a body tag, an HTML tag and all that. Going back to the source is just very basic source, as you can see. What I'm going to do now is, let's go ahead and build. Make sure we're okay there. Looks like we are and now what I'd like to do is add reference to the choosing control into my form, my Default. aspx. And just as a reminder this is the one that has the first and last name and all that type of stuff in it. So how do we do that? Well it's actually very, very simple. What I'm going to do is create a new row right above. So we'll insert row above. And in this row I'm going to go ahead and we can go in and, and modify our cells. I'm going to merge them. So that we just have one cell there. And now all I have to do is grab my file, my header, and we can just drop it in. And you'll notice that it automatically renders. And then from here I can edit the user control to go right to it. So it's actually very handy, very easy to use. Now this just saved us quite a bit of code by dragging and dropping in this case because we would have to another directive called register. To register the path to the user control the name space and then give it what's called a tag name, so let me show you how that works. So, if we go into the source, another directive has been added called register, and this is used with registering user controls. Now, the register has three things you have to specify on it in the case of user controls, and that is what's the source. So, what's the path to this particular control? What's the tag name? Now, this can be anything you want in fact the next two can be anything you want but because we called it header we're going to call the tag for this header it's very much like when you do a label. You do ASP:label. Well, label is, like, the tag name. In this case, we're going to name it the same as the class header. Now tag prefix references the path to it and the namespace that we'll ultimately be using in this path. So it's just a prefix to let us know this is a custom control. That we're going to embed in the page. Now, in this case, I'm going to change it to user controls, since that makes a lot of sense. And we'll come down here and we'll simply change this to user controls, in this example. So, now we have a way to with a combination of the prefix and the tag name reference this particular file which internally has a name space and a class. And again you can name these whatever you want though. It doesn't have to equate or match up with the class or the name space, but I like to usually keep them very similar. It makes more sense. So now we have user controls:header. Very much likely to do with ASP: and then we put label. Now this is not a web server control, because it's just an ASCX file. It's not actually a full on web control that it derives from a class called web control or control. Instead it's a custom user control that simply references the ASCX of course. But you'll notice, we do have to put run and server. This is a server side control. Java Script the client side browser in general would have no idea what to do with the control directive or the label tag. Needs to be rendered at the server side. So now that we've embedded that directly in our page we can go ahead and run it. And now we'll see that our header shows up right at the top. Our dynamic code runs to write out the date in this case and then we can fill out the form, I hit submit, everything else functions as is, because this is just a part of the page. Now likewise, in the grid view demo here. If we want to drag and drop it inside of here then I can come over, I'm just going to hit Enter there, and drag it up in. It renders it. And now when I run it we'll get the same exact thing. And now if I want to change something. Lets say we don't want to do the date want to do the time for some reason. Lets do two short time string. And now when we run it. You'll see that it updates across all our pages, cause of course we're sharing it, much like an include file. So there we go. And we'll come back to the GridView. View it in the browser here. Very nice. Now the last little tip I'd like to share is there may be times when you want to dynamically add the user control into what we call the control collection of a page. Now that can be done as well. All we really need to know is the path to the user control, which we have here. So let me show you an example of doing that. Let's say that we have a new web form, so I'm just going to add, New item. We'll go back up to Web Form. Web Form1 works for me. And let's say that we'd like to, within the form called Form1, we want to dynamically add this user control. Very easy to do that. Now keep in mind, user controls, although I'm doing a header or a footer, we could do a menu, they could be used for anything. It could be grids. It could be a, a grid with filtering controls above it, that you need to share across pages. So, kind of the sky's the limit here. Now what I'm going to do though, is let's say that we have, this is some text and then this is some other text. Let's say we want to add the user code control right between these two. Well, we could do that pretty easy. I'm going to assume we had a, a BR tag here for a line break. And then right inside of this I'm going to add what's called a placeholder control. We're going to give it an ID of, we'll just call this headerplaceholder. And we have to do runat sever of course. Now a placeholder does what it says. It just simply reserves some space in the controls collection. If you don't put anything in it, nothing happens on the client side. This is a server side only tag that allows you to add children into it and if you do, they'll get rendered and added to the html. So now what we'll do is we'll come back into our code. And when the page loads, I can go and find that control and add to it. So we can come in and say, Header Placeholder., and then we can say, Controls. Add, and I'm going to create a control here. So the way I'm going to do that is we're going to say, var ctl equals page. loadcontrol. And load control takes the virtual path to get to our in this case header. So I'm going to give it the path that we have. I'm going to make sure we can find it though, so I'm going to say start at the root of the site, that's what the tilde means. Go down from that into the user controls folder and then find the header. Once we find that load control at returns a control. And now we can go ahead and add that control dynamically into the place holder. So let's go ahead and build and try this out. So we come down web form1. aspx, and you can see it added it just as we expected, right between the text. So, I hope your apps will look a little better than this one but, you get the idea. This could be a dynamic report that gets added. Anything that's reusable that's a user control and so all we have to do is call page. loadcontrol to do that. So to wrap up this demonstration of use controls, we've talked about how we can right click and add a new item and select web user control. And that they have ASCX extensions. Once you filled in the programming logic if any and the user interface code including web controls into that ASCX. Then we can simply drag and drop from the solution explorer into our page where we need that. And that's all you have to do. You're in business at this point. Then the final thing we just covered of course is how do you dynamically load user controls? And so page. loadusercontrol is your kind of gateway to doing that. And that's something I've, on, depending on the application, have leveraged quite a bit when I had some dynamic controls that I want to add into the page based upon for instance, user security. So, that's a wrap-up on how we can use user controls in ASP. NET web forms.

Summary

In this module, we've talked about several different features of ASP. NET Web Forms and introduced you to the overall framework. We started off by talking about some of the directives that are available and covered the page directive and the different attributes that you can add to it. One of the more useful ones that I demonstrated was the maintain scrollback position on post back. Which allows us to, without any code on our part, emit some JavaScript code that can track either horizontal or vertical scroll bar position, so that when a form post's back it's very easy to keep the user where they were in the page. Now later on in this module we also talked about another directive, called the control directive, that's used with user controls. We also discussed web server controls and as I mentioned earlier this is really where it's at with ASP. NET Web Forms programming. By using the User Controls, handling the events, interacting with the properties, you could do a lot of work without a lot of code. And so we talked about how Web Server controls allow HTML to be generated and set back to the browser without doing much on our part in many cases. We also talked about how defaults could be set on the form tag and how we could set the default focus so that the cursor is automatically blinking in a particular text box. And we also discuss default button so that if the user goes in and hits Enter it automatically would submit that form for us. And then finally we talked about user controls and how they allow us to encapsulate frequently used ui into one reusable control that can be shared across multiple pages. And we discussed how we create the ascx file and then how we import that and reference it within our web forms pages. So everything we've talked about here is very, very key to getting started. And moving forward, we're going to build upon this and start talking about data access and other key technologies that you'll use as you build ASP. NET Web Forms applications.

Working with Data Source Controls

Introduction

Hi. This is Dan Wahlin with Pluralsight. In this module, we're going to talk about how we can use Data Source Controls in our ASP. NET web applications. So, we'll start off by talking about what is Data Binding? And why would we want to use it as developers. And in a nutshell, it's just a great productivity tool that allows us to bind our data without having to write a ton of code. We'll also talk about Data Binding Expressions and what those are. From there, we'll jump into Data Source Control. So, we'll talk about Query in the Database directly with the SqlDataSource. Querying a data access object, with the ObjectDataSource, or even querying into an entity model with the EntityDataSource. Then, we'll wrap up by talking about a newer control to ASP. NET, and it's called the QueryExtender. And this allows us to extend existing entity queries that we might be running in our ASP. NET applications. So, let's go ahead and jump in. We'll start talking about data binding.

Data Binding in ASP.NET

Data Binding allows us to take data from a variety of data sources including XML files, databases, custom objects and more. And bind those into web controls in our applications. So in this section, we're going to talk a little bit about Data Binding, what it is, and what Data Source controls offer us. And then, we'll move forward and talk about expressions, and how we can use these different Data Source controls. So first up, Data Binding is something that's built into ASP. NET. So the overall framework has a way to take a result set, from say a database, and bind that into a control such as a grid view which we've seen in previous modules. Now, in the old days when we did that, we'd oftentimes have to write some looping logic to loop through the different result set rows. And then write out the HTML that would be appropriate for those rows. Well with ASP. NET, we can actually leverage data binding features to take our data, get that result set, and then bind it right in to something like a grid view. With just a minim, minimal amount of code on our part. Now, something that can really help with this is Data Source controls. Data Source controls allow us to grab data from databases, XML files, or custom object structures and then bind that data into these different web controls. And you can bind, as mentioned, relational data, hierarchical data, or even custom collection, such as a list of customer, as an example. Now, data aware controls such as grid view, details view, list view, and there's many others, they all have a special property that works with Data Source controls called DataSourceID. And that's going to become really important when we hook up a Data Source control that knows how to get the data to the control, such as a grid view, that actually knows how to render the data. Now, there's several different types of Data Source controls available. And you can see those here. In fact, this is the wizard, when you use Data Source controls, that Visual Studio will pop up. So you do have the option to type these. But, it's definitely much easier to use the built in Visual Studio wizard and I'll be showing that coming up here. So you can see, we have an access database source control. We can do a sequel database. Now, it says sequel but you can actually use it against multiple types of database not just sequel server. We have an entity framework one. And this would be for the entity framework models that you might create. We have a link to SQL data source for a link to SQL diagrams. The object data source I mentioned briefly earlier is used more in end tier. Or sometimes, you'll hear end layer architectures, where you or somebody else on your team might have already written a data access control and you simply need to tie into that control and bind the data that it returns. So we can use an Object data source for that. And then finally, we have two hierarchical controls built in and these both worth, work with XML type structures out of the box. The first is the Site Map. And this is used for more navigation type features. And then finally, we have an XML data source control that can bind to any type of XML feed, such as an RSS feed out there. So those are the basic data source controls that are built in out of the box with ASP. NET. And by using these, and walking through the wizards that I'll be showing you, you can get a lot going into web pages without having to write a lot of code. So now let's switch gears, and we'll talk a little bit about something else you'll see in conjunction with Data Source controls, and that's Data Binding expressions.

Data Binding Expressions

Data Binding Expressions are an important part of the whole data binding process in ASP. NET. So we're going to take a quick look at some of the different expressions you're going to encounter as I do some demos coming up in this module. So that, when we see them it'll make a little more sense. So, expressions provide a really simple in a super compact way to either bind data to controls, or maybe even get data from something. A good example of this if you're going to use a data source control that can hit a database, and that's defined in your page. And if we're trying not to write a lot of C# or VB, then how are you going to get the connection string? Sure, you could hard code it into your page, but probably not a good thing because you might have multiple pages that use the same connection string. So, we'll store that connection string in web. config and then we'll use a data binding expression to go get it. That would be one example. Now, the others that are shown here. Here are used for data binding. So eval is used once you have a result set and let's say we have a row back and that row has a first name property in it. Then, that property needs to be bound somehow. And the way we can do that is we can use the Eval statement. Now I said property in this instance, because it might come from a custom set of objects. But it could just as easily be a field in a result set as well. So property or field. You can use this to actually bind that particular data wherever you'd want it to be written out in the page. Now Eval just writes it out. Bind can actually be used to kind of go both ways. It'll rat it out plus if a forms posted back in say an edit scenario. We can actually take care of that and update the database or whatever our target data source is. And then the final one listed here is the XPath expression and this is used with the XML data source. Now the XPath expresses not something we're going to cover specifically in this module, but I do want to touch on it briefly. And, it's basically used to supply an XPath statement. So, if we loaded XML data up in the memory, how do we query that data and actually get out the target nodes or target data that we want to write out to the page? Well, the XPath expression can be used to do that. We can give it an XPath statement, if you've done XPath before, and it'll go find that for us. So the last one here is an example of using the most common expression that you'll come across, especially if you bind result sets, multiple rows of data, and that is the eval. So the first thing to note on all of these is you'll notice this percent pound, or percent hash, however you want to look at it. That represents a data binding expression. And so you'll always know when you see that that there's some type of data binding going on. Whether it's hierarchical data with XML or whether it's custom object properties or whether it's fields from a result set that came from a database. So in this example the Eval this would be used if we want to bind a specific field into a control such as a grid view. So you might have a grid view and you'll define how every row should be written out and the type of HTML that should be generated. Well the Eval statement allows us to say hey I want to write out the value of. And then we can put for instance first name here. If that was one of our fields we're getting back from the database. So Eval's very common, something you use quite often in your ASP. NET webforms when data binding. Now here's an example of using Eval within a grid. So we have our GridView web control. You'll notice we have the id, the runat server, very, very standard, much like we've covered up to this point in the previous module. But we have something called a TemplateColumn. Now there's different ways to write out data with GridView, ListView, Repeater, and other controls. And the way we do this is something called templates. We don't want to have to write the looping code ourself to loop through every row in the data. And then write out the HTML. Instead what we're going to do is to find all the columns we want to write out in the grid. And then bind the data to the grid, and let it take care of writing out the HTML. So in this example, we have a custom column. That's why it's called a TemplateColumn. And for each row, that's what ItemTemplate represents, for each item, or each row in the data that we're binding. We want to write the value of contact name. And so that'll go in, grab the contact name out of each row and do it. So if we have 50 rows this ItemTemplate is going to get called 50 times, we'll have a single column though. That'll be written out, and then it'll write out the contact name 50 times. Now of course you can repeat this. So under this one, I could add more TemplateColumns, and more and more and more. And there's a lot of different ways you can do this. So that would be an example though, of actually applying a data binding expression. And so now that we've introduced the percent pound, and what a data binding expression looks like, let's go ahead and jump into data source controls. And see how this stuff can be used.

Using the SqlDataSource

The first data source control we're going to talk about is the sequel data source. And it can do a lot with very little. Makes it very, very easy to query different types of databases, get that data, and then get it into inbound. With controls in your website. Now, you can also do updates, inserts and deletes with it as well. So it's very, very flexible that way. So, the SqlDataSource allows for a declarative type of data binding. So, instead of having to write custom C# or VB code, we can actually go in and use the SqlDataSource within the aspx page itself. Now, it does have events. So it's, it is possible you might have some code that you want to tie into, such as when an insert occurs, or something like that. But out of the box, often times, you can just drag it onto the page, walk through a wizard. And then you're ready to go. You can get this going very very quickly. Now, it supports four different types of. net providers, out of the box. Now, a. net provider is just a set of classes that are able to hit. A back-end database. So we have a SQL provider that's specific to SQL Server. OleDb, which is more of a generic type. For instance, you could hit Access or DB2 databases. ODBC is really for anything out there. It's a more old-school type of driver. And then finally you have support for Oracle as well. And so these four managed providers are supported. And although it's called the SqlDataSource, it's not just for SQL Server. Now, by default, when it gets its data you can bind it and then do sorts and filters and all kinds of stuff. So it uses what's called a, a DataSet mode. And that's where it actually loads the records up into memory, and then binds those, but you can switch it to also stream the data. Now if you stream the data, paging and all that is not going to be available because it doesn't know how many records there are. It's a stream of data. Get one row at a time. But, you can do that by changing the control's data source mode. So if you're doing a read only operation where you just want to get the data into the page as fast as possible. But you don't need filtering and sorting and that type of stuff. Then, I would switch the data source mode to data raider. If on the other hand you want to do anything other than that such as paging, sorting, filtering, go ahead and leave the default there. Now, as I just mentioned it does support. Paging, filtering, sorting. It can do inserts, updates, and deletes, so what we call CRUD operations, create, read, update, delete. And it even has built-in support for caching which makes it very easy to not have to hit the database every time. If you have some data and that data only changes let's say once every 30 seconds, 60 seconds, or even longer then you can tell the control that hey I want to cache this data so that every time the page loads during that 30 second period it just pulls form memory. Doesn't have to touch the database. Really speeds up performance. And enhances the scalability of your website. Now here's an example of using the SqlDataSource control. Now normally you're not going to be typing this in. Normally you use something like a data list, which just simply spits out a list of items. And then you'll bind that data list to a SqlDataSource. So you'll notice the magic glue here. Is this DataSourceID, which I mentioned earlier, is available on all the item's controls, the different controls that can support multiple rows. They have this DataSourceID. Notice the name here, sqlCustomers. We'll come on down to our SqlDataSource. Now, this is a server control. So notice the id matches first off. Single customers. We have runat server. We have, what is the connection string? Talk to the database. And then, what's the command we'd like to execute. So the ConnectionString is actually using a connection string's expression. So we talked about data binding expressions which are the percent pound. Which you see one up here the Eval. Well the percent dollar are a different type of expression and this one allows us to go into our web config file, go find the ConnectionStrings area and then grab the ConnectionString. The wizard I'm going to show you in a moment. He actually puts all this in, you don't even have to type that piece. You can actually select what you'd like to hit all visually, and it'll create the ConnectionString for you, it pretty nice. Now the select command here is a special type. You'll notice in this case the command type is not just a text, it's not a sequel statement, a select field from table type of query. It's actually calling the stored procedure so, for those of you that have to support. Port store procedures, that absolutely is supported and those of you that just want to do SQL statements, that's also supported in this particular control. The different data source controls not just the SQLDataSource, but some of the others. They support different types of parameters. Now it's one thing to have a store procedure, or a select statement you execute that just returns all the records, but in reality a lot of times we're doing some filtering. We're doing a where clause. Well where do you get the data for that where clause? Does it come from the query string? Does it come from a drop down list that might be on the page? Is it stored in a cookie. Is it posted back in a form? There's a lot of areas so. These are the different parameters that are supported. So, you can get data from a control, from a cookie, from a form variable. Profiles relate to storing personalization data. So maybe, you have your favorite report stored. And, this will pull from that QueryString does what it says. It grabs data off of the QueryString. And then, session data is a way to store data while a users visiting a website so that every time they visit the next page, that data's available on the server. And so, SessionParameter allows us to pull from a session. So, an example of using some of these are show here. We have a SqlDataSource control, but you'll notice we're using a control parameter to specify what type of parameter should be passed, in this case to a StoreProcedure. So, this StoreProcedure. You can see is called GetCustomerByCountry, so it kind of goes without saying, we probably need to pass the country. So, we're going to get the country, in this case, from a ControlParameter. Now, what that means is all of our controls have IDs, typically, in our page. But, we're going to say, hey, I want to go grab the, the control ID of ddCountries in this case, a drop down list and I would like to then go to the selected value of that particular control and whatever value's there. I want to go ahead and assign to the country parameter that would be passed in to this stored procedure. So, this allows us to actually specify where the data's coming from, in this case, the DropDownList, what's the data type of it. It's a string, what's the property of that control we want to get to. And then, finally, what's the name of the parameter that this StoredProcedure needs. Now, if it was a SQL statement instead of a StoredProcedure here, you could also do parameterized queries. And, it'll actually help you generate that visually, as well. It's very nice and easy to work with. Now, you can also go in, in addition to grabbing from controls, oftentimes, you might actually have a page that posts back, but to a different page. And, the data's up on the QueryString, up in the URL in the browser. Well, that's easy to do as well. We can simply in our SelectParameters here if we want to grab our country we can say hey let's go up to the QueryString. I'd like to grab a QueryString field called country name. I'd like to map that as a string value to the country parameter and then that gets passed in to our stored procedure, or our select statement, either way. So, this provides a really really easy way to supply data dynamically when we're filtering our queries. And, you'll see, as I run the demos in the next section here, that I really don't even have to type any of this either. I can do it visually, and it's very very easy to work with. So, let's take a look at how we can use the SqlDataSource in ASP. NET Web Forms apps.

Demo: Binding Data with the SqlDataSource

To demonstrate using the SqlDataSource control, I've created an ASP. NET Web Forms application projects. So, it's empty, just has a web. config at this point, which doesn't have a lot of code in it. So, I'm going to go ahead and add in an ASP. NET Web Form and we'll just call this Default. aspx. All right. Now, what I'd like to do is hit a database that has customers in it. So, we're just going to use the ever popular Northwind database for this particular demo. We'll use some other databases coming up. But, the Northwind database has customers. And, customers are located in countries. It provides a very simple way to get started with this particular single data base control. So,, what I'd like to do is make it so a user can pick a country, and then drill down and show the customers in that country. Then, even click a customer. And then, show the individual data for that one customer. So, we're going to use three controls. We're going to use a drop down list, a grid view, and with something called a details view, which allows us to show one row of data. All right. So, to get started, we need a database. Now, I could hit a more industrial strength one that I might have out in the enterprise but for this demo we're going to use SQL express. So, I'm going to right click on my project and say add ASP. NET folder. And, you'll notice I can pick several different types of app folders here, so app global resources and local is for things like language strings. App data is for databases. Add browsers is for browser configuration files. You can use this to detect different types of browsers. And then, you can have a theme folder which is for CSS, images, and themes in general. So, we'll pick app, app data. Now, on the desktop, I have a database folder here. Dev databases, and I'm just going to select the North win file. And, I'm going to go ahead and copy that into my app data, so we'll just right click and paste that in. So now, I have a database that I can work with. And, as I said it has customers and countries for those customers. So, first thing I'm going to do is we need the user to come in and select from countries. So, what I'm going to say is select a country. All right. So, we're going to drag on a drop down list. Now, you'll notice right as we drag that control on I have this option to do Choose Data Source. We're going to do that so I'm going to say Choose Data Source. And, I'm going to say I want a New data source. Now, it's going to pull up a wizard now, which I showed a little bit earlier in the module, to let me select different types of data source controls. So, everything from Access to XML files, to Entity Framework type queries can be selected. Well, we're going to do a SQL database. And, I'm going to go ahead and call this Countries. Let's do SqlDataSource. Keep it kind of obvious. And, we'll hit okay. Now, it's going to say, what database do you want to hit. So, I can hit New Connection if I'd like. And, we could select everything from ODBC to Oracle to SQL Server, but normally, if I build we can come in and select the Northwind database now because that doesn't show up. Rather than cancelling the wizard, we could say new connection. We'll say a SQL Server Database file, because it is an MDF file, and we'll say continue. Now, that's going to let me browse to my file, so I'm going to run off to my desktop, find this project, and from this project we're going to go in and select the this MDF file. And, I'm going to go ahead and use the default authentication. We can test the connection. Looks like it worked, and then we'll hit okay. Now, what that's going to do is automatically make the connection string to hit the Northwind MDF file. So, you'll notice I have my connection string here. It's hitting SQLExpress, but you'll notice that there's something kind of unique in the connection string. It's actually pointing it to DataDirectory. Well, that happens to be the appdata folder. And then, I have this Northwind. mdf. All right. So, I'm going to hit next, and now it's going to say do I want to save that connection string into my web. config file and I do. And, you can change the name, but we'll leave it. Now, this is where it gets really fun. Super easy, all you need to know is how to work with these, different screens here, and they're very simple. So, I want to select all the customers, but, in the Northwind database there's actually not a customer's table, it's not a super normalized data base. So, in the countries there is a country field though. So, you'll notice that automatically, once I select the table, allows me to come in and then click on which field. So, I can do it that way. You'll notice as soon as I click these it writes my Sequel for me. Very nice. Or, I could come up and I could do a custom Sequel statement or store procedure, and when we hit next we'd have to specify that. But, for this demo, we're going to select country. Now. Because country is duplicated across, the customers, I only want to return the distinct countries. So, I'm going to simply check that return only unique rows, and we're off and running. So, I'll hit next, and now it's going to let me actually test the query without even running the webpage. Really, really nice feature. And, there's all my unique countries, that I can now reference. So, we'll hit finish. All right. So now, it's going to say, OK. You have a drop down list. What do you want to show for the value of, the behind the scenes value of the drop down list, which is a acml select, and for the actual text that the user sees. Well, in the case, I just want to view the country name for both. I want the backing value to be country and I want actual value of the usersys to be country, so we just hit okay. So now, what's it's done, is it added a SqlDataSource, it bound it into my drop down list. And then, the last think I'm going to do here is say I'd like to enable AutoPostback. Now, this'll add a little bit of JavaScript. So, that when the user of this page selects a country, it's automatically going to go and do a post back. So, what we can do is actually come over to our properties and come into our properties for our grid, or our drop down list. And, a really nice feature that we have is something called AppendData bound items. And, I'm going to set that to true. Now, what that's going to allow me to do, is, if I add in to the source, an actual list item. So, I'm going to say, ASP. listitem, and I'm going to go ahead and say that the value of this is nothing, but that the text is Select a Country. And, we'll do a shortcut close tag there. Now, what this will do is automatically add Select a Country at the very top. And, because we did this AppendData bound items. And, set that to True, all the countries from the database are going to be loaded after this particular item. Really nice. Didn't even have to write any code to do that, as far as C# or VB. Now, let's take a look while we're here at what we have so far. So, we have a drop-down list. And, we have that append data bound items, but the key is this guy. We have a datasource ID and that datasource ID points down to country, countries, SqlDataSource, which is there. We also have an expression. Now, this expression will go into WebConfig. Go into the connection strings and grab that ConnectionString. Now, that's been added for us by the wizard. You'll see there's my ConnectionStrings. There's my ConnectionString name. That's my key. And then, there's the ConnectionString it generated using the wizard. Very nice, very easy to work with. So, this ConnectionString, will automatically pull it out using this expression right here. Here is command execute. Now, this is a text command so it's a select statement, obviously. Okay. So, we're kind of on the right track. Now, if we run this as is, we should basically get some countries. And, it's not going to do much from there, but we can select them and then we'll keep going here. So, there's my select a country. There's my countries and it now posts back. Okay. So, the next thing I want to do is under the SqlDataSource, I want to grid view, Control. So, we're going to come down into the Data section and drag on a GridView. Now, the grid view, on auto format give it a little bit of coloration here. And, I also want a Data Source for this one. I want to select all the customers. So, I'm going to do the same process. I don't want the countries SqlDataSource, I want a new one. We're going to say SQL Database. We can call this customers SqlDataSource. Hit okay, and we can pick the same connection string we already have. And now, it's pretty easy. We can come down and say, I want to select customers. Let's say I'd like to maybe select the id in the first couple fields here. Notice, it generates all the SQL. But now, I want a where clause. I want to where the country equals the value from that dropdown list. So, we're going to select the WHERE button here. I'm going to say where, here's all the fields from my table. The country, equals. Now, the source is going to be a control. It's going to be that drop down list. Now, this is a good example of why you should give good names to your controls. Since, I didn't give an explicit name to it, it's probably going to be drop down list one. And, you'll see it right there. So, there's my grid, there's my drop down. So, we'll select it. And, if there's no value selected, I'm going to pass NULL to the query. So, what this is now doing is, it's going to do a where Country equals the value from, DropDownList1. SelectedValue. So, DropDownList1. SelectedValue will automatically get substituted in for this parameter. So, we're going to add that and hit Okay. You can see there's my query now. Now, from here what I'm going to do is hit Next and test it. It's going to make me supply a country, so we'll say USA in this particular example. Now, we'll go ahead and hit Okay to test it out. This will query the data base and we should get back all the customers and those fields we selected from the customers table there. So, really nice because I didn't have to write any custom connection command. Issue the command to the data base, it's going to do this for me. And, we'll hit finish there. All right. So now, we have a grid view, and I can even go in and enable paging, I can enable sorting, and even enable selecting a specific customer. And, I'm going to want that for this demo because I'd like to select a customer. And then, below this show all the details for that customer. So, the final thing we're going to do in this demo is I'm going to drag on another control called a details view. Now, the details view, kind of like its name implies allows us to basically show one row of data. So, I'm going to AutoFormat it. Just pick something that has a little bit of color to it. Choose a Data Source again. So, I don't want any of those. This time, I want to select the customer. Where the customer ID comes from, whenever they clicked on up here in the grid. So, we'll hit New data source again and you kind of get the pattern here. So, we'll say customer sequel. Data source. We'll pick the same connection string, and we do want the customer's table, and we're going to say I want everything. Now if you don't like doing a star, you can certainly check all these for the sake of, simplicity I'm just going to select the default here. And again we want a WHERE clause. Now this time, we want to select the customer ID from the grid. So I'm going to say where CustomerID equals. And the source will be a Control again. Which will be the GridView SelectedValue. Very similar to what we did with the drop down. And if by chance there wasn't one, and this got invoked, we want null. We don't want to show it. So you can see again it's going to filter where CustomerID equals a parameter. That parameter value is going to come from the grits we'll add that. Now like before I can hit next and I can test. Now one other interesting thing we can do here though, is if I hit advanced I can even have the SqlDataSource for this details view. Generate Insert Update and Delete Statements. So I'm going to go ahead and do that in this case, and we'll instantly get an update capability without having to write any code on our part. So let me hit Next here, and we can test it. So the only Customer ID I know off the top of my head is ALF. Ki that's the first customer, so we'll hit OK. Goes out to the database, there we go. We get back that record. And we can hit Finish. Now from here the Details view also allows us to do things like, do we want to enable an insert capability, editing, and deleting? So we'll go ahead and say yeah, we'd like. All that stuff in this case. So now that's going to show everything, including the image. I can actually come in, though, and we can even edit the fields that are displayed here. So, you'll notice the selected fields have all, everything. But I'm going to get rid of the image, let's say. Let's say I don't want to show that because that's just a a JPG path name. It's not the actual real image. And since I don't have that image, we don't want to show it. So we'll go ahead and. Just delete that and hit OK. Now you'll notice it takes that out of our details view. Now likewise up on the GridView I can edit the columns. So we can come in and add columns from our selection our bound fields. And delete columns, and do all that fun stuff as well. All right, so that's it. Now if we go look back at the source, you're going to see that each of these controls now has a data, a SqlDataSource. So there's the one for the, drop down list. There's the grid code. There's the one for the grid. Customer SqlDataSource. They're all using the expression to get the connection string. And then finally, here's the one. For the details view. Now, notice this one has quite a bit much or quite a bit more, I should say, going on. We have delete parameters, insert parameters select and even update parameters. And this is what it's doing is it's reading from the textboxes. That are in this details view once it's rendered. And it's going to render that to the screen. And then once we hit a button it'll put this back in the database using these different commands that you see. All right. So let's go ahead and see what we get. Just by kind of clicking around in the wizard there. All right, so we'll come in, we'll select a country, we'll say Finland. Now it just pulled all the customers from there, I can actually come in and sort these if I'd like. You'll see that works very nice. I can select a customer. We're now in the details view. I'll hit edit, and let's just put a Wilman, Kayla two just to see if it works and we'll hit update. Now it looks like it did but let's go ahead and remember I did Finland, we'll come back. Select Finland again. And select the appropriate one, which was this one. And you can see it did work. We have this Wilman, Kayla two. We'll hit edit and we'll take that off. So I could also delete. I can do a new record if I'd like. All that type of stuff is built in for me so it's very, very flexible and you can see how easy it was to do with the SqlDataSource. So that's an example of how we can use the sequel data source control, and bind it to different types of control such as the drop down list, the grid view. And the Details view control.

Using the ObjectDataSource

Although the SqlDataSource provides a really productive way to get data in to and out of your application, it's pretty easy to see early on that there's not a lot of code reuse going on. So I might end up duplicating code across multiple pages. And in larger apps, that may end up being a problem with maintenance and reuse. So another alternative to that if you don't want to hit the database directly, and kind of have a single tier type of application architecture. Is to use the object data source, and the object data source basically allows you to do declarative binding still, but instead of binding directly to the database and having those SQL statements right there in your ASPX. You'll instead rely on a more of a better architecture if you will. So we'll have more of an n tier n layer architecture, where you might have a data access class or classes that you hit. And the ObjectDataSource can then call those classes to get the data. Now the benefit there is now if something changes we. We can go to one place, and make that change versus with the SqlDataSource the only downside really is that if a query changes and that query is used across multiple pages. Well now I obviously have to go to all the pages to fix it. So this does support paging, sorting, filtering, insert update delete. All that. We can even do caching so it's very much like. The SqlDataSource, but the difference here is that the ObjectDataSource as you can see down here. It ties in to the GridView in this case. It's going to bind the data to that GridView, but you'll notice is has this SelectMethod and TypeName. So, what this is doing is the TypeName represents the class we want to call. This is our DAL. Our Data Access Layer class name. The SelectMethod represents the method that we want to call to do select statements and then of course, you can also have Delete, Insert and Update as well. So, you might have a class that's similar to this as far as the kind of stub methods that you see here. So, we have a GetCustomers, GetCustomersByState,. Get customer by ID, some select methods. But then you'll also notice we have update customer, delete record, insert record, and those types of things as well, and what we can do then, is use the object data source to call those different methods on that data access layer class. So by doing this we can now. Modularize or segregate our code into multiple areas. Makes for better reuse. Hopefully easier maintenance down the road as well. And we still get the benefits though of declarative binding to grid views, details view, drop-down lists. Those types of controls. So let's take a look at how we can use the object data source to make calls into a class that can provide us the data.

Demo: Binding Data with the ObjectDataSource

Using the object data source is much like using the SqlDataSource except for we're going to call. A custom object instead of calling the database directly. This allows us to break out our architecture into more modular classes so we can reuse those across pages or even across applications, potentially. So, in this example you'll see, I have what we created in the previous section covering SqlDataSource and I have a dropdown. A GridView control and a details view. And so what I'd like to do ultimately is wire these up to ObjectDataSource controls, which are going to call into a custom class. Now that custom class is here. It's called bal. We'll just call that business access layer in this case. And it's going to get countries. Get the customers by the country, and then get a specific customer based on a customer ID. So to do that, and return those pieces of data, we need some way to query the database. Now we could use adio. net, and then we'd have to write connection objects,. Command, data readers or data sets, and there's a little bit of code involved there. So, for this demo, what I'm going to use is LINQ to SQL, which is one of the data access options available in. NET. You really have two main ones, LINQ to SQL, and Entity Framework, and then those are called object-relational modeling tools. Now if you don't want to use those you'd have to go to the ADO. NET I mentioned, and you'll have to write a little bit of code in that scenario. So, I'm going to right click on Data and we're going to say Add New Item. And what I'd like to add into here is a LINQ to SQL Classes. And I'm going to call this Northwind. Now, this is going to give me a designer surface that I can then go to the Database Explorer. And let me point out that I have a database here, Northwind in the App Data folder. So, I can now drag on a table under here. And, or multiple tables and work with it. So the way you can get to the, database explorer is that you can just click right here or if you were trying to get to it another way. If you go up to view, other windows, database explorer, you can do that. Now, if you're in. Paid version of Visual Studio, then you can actually go view Server Explorer and you can get to it that way. So, what we'll do then is we'll go to the database explorer. You'll see I already have it open to our North WIN tables, so I'm just going to drag on this Customers. That's what we'd like to query. And there we go. You can see it renamed it to Customer because this represents a single entity, and then here's the different fields for that particular customer. So let me go ahead and save that. Now what I'm going to do is come into here, and we'll take out the nulls and I have some. Code snippets kind of prepackaged to just add. So we're going to add these and talk a little bit about link queries. So in this example you'll see I'm using this thing called a NorthWindDataContext, which lives in the data name space. Now, going back to here. The name of this file is Northwind, and therefore the, the class we'll use to access the database is NorthwindDataContext. So we're going to create a new instance of that. Then we're going to write a link query. And this link query basically says, for each customer in the customers, and I'm going to give it an alias of c. Select the customer's Country. But I only want distinct countries, I don't want to have duplicates. And then return those a list. So that's kind of our first query. What that'll do is give us back a list of string, and we're going to bind that to our drop down list. All right? For the next one, we're going to do a very similar thing. Expect for we're going to query and get customers now based on the country. So, you'll notice in this example I do the same thing. I create an instance of my Northwind data context. I say go grab the different customers but I do a where clause. So I want to say where the customers country equals, whatever's passed into this method as the country. And then we'll return that as a list and then finally. With this one, we'll do a very similar thing, but we'll grab a specific customer, but we're going to do a where clause based on customer ID. And so, provides a very easy way, you'll notice I don't have open and close connections or commands or. For data readers and work with data adapters, and the standard adio. net classes that you'd have to work with. This provides a nice way to work with databases and, later in this module we'll see another alternative which is entity framework as well. Very similar though in how you can do the queries. Okay, so that's going to be the class that actually supplies my data. Data is this particular case. And I might have some business rules once the data s gets returned here. I might do some further further filters based on security or something. But for our purposes now this will work great. So now I'm going to come back to customer viewer, and what I need to do is bind. The drop-down list to a Data Source. So we're going to say New data source. But instead of picking SqlDataSource, we're actually going to come in and we're going to pick an ObjectDataSource. We're going to query into this bal. Now we're going to go ahead and name this. Countries, object data source. And now it's going to say what is the object you would like to query? So you'll notice that it automatically picked up the different classes that I have available in my projects, specifically in this case, because it's a website project in my app code. Folder. So we'll select BAL. Now it's going to say what method would you like to call? Well, we want to return obviously countries. And then we'll hit finish. And it automatically then will come in and it puts length. Well, we don't want length in this case. It's because it's returning a string. We want to go in and just bind to that string. So, we'll, we'll go ahead and fix that real quick. So if I go in what happened here is because it detected we're binding to a string. It thought that well, we must want to bind to a property of the string so it picked length. Well, in this case, we just want to buy into the actually strings. So we're going to take out, that code there, and let's go ahead and run it in the browser and see what we have so far. Okay, and there we go. Now we can. As was shown in an earlier demo, I like to come in and usually have the user pick something. So we'll go ahead and say, value is empty strings, text is select a country and then we'll come in to the, properties here. It's just a little bit easier to do, and say append data bound items, true. And that'll put all the data from the database below that particular list item. So, a really nice feature. So let's try that one more time. Looks good. And now we're on our way. So the next part's pretty easy. We could do a very similar thing. I'm going to come in. We can choose a data source on our grid now. Again we'll pick object data source. This will be our. Customer's object data source. We'll select our business access layer and then we'll pick. Get customers by country. Now, we need a country. Well, we of course have a country. So it's going to say, where do you want to get the value of that? Well, we're going to get from a control. The name of the control is our ddCountries, in this case, our drop-down. And if there wasn't a value, we'll just. Apply null for that. And, that's automatically going to pass the country in to this method that you saw earlier. It makes it very, very easy to do that with the object data source. So, we'll go ahead and refresh these fields here and I probably don't want to show all those. So, we'll click on that, we'll say edit columns, and we'll take out from city on down here. All right. So, that will get rid of the columns. I just want to show a few in this example. So, we'll hit okay. Now, you'll notice that my data grid is automatically set up to show those particular items. Now, I'm going to come in and we're going to say enable selection. Because we do want them to be able to click on a row, because once they do that, we want to load the details. So, now we go through the same process, do a new datasource object. This will be a customer object datasource. Pick our business access layer, get customer takes a customer ID. Well, we can get that a control and the control is going to be our gridview customers. So, we'll go ahead and select that. If there isn't a value for some reason, we'll pass null. And, that's automatically going to grab from the selected value of my grid view so we'll hit finish there. Refresh these fields and we want to show all of them in this case. So, we'll show all these. Okay, And, that's it. You'll see that I now have a reusable class. So, if I have other pages that maybe need to get countries or customers by country or whatever it may be, I can now come to this object and update it. Now, that updates across my whole site, and any object datasource that maybe binds to that particular method, it's automatically updated there. So, it's very good for maintenance and reuse. All right. So, we need, it says, data keys must be specified. So, right now, we need to go on and make sure that a customer ID is specified here. So, let's come into the properties, and we'll come to a property called data key names and I'm going to put customer ID. And, I'm going to do the same for the details view. And, let's run that. All right. So real quick, before we go any further. What happened there is because I didn't specify what the primary key was, the unique identifier for these rows. I got an error there because it says, hey, I don't know what to select. Or, if you're doing edits or something here, how would it to know what the key is? And, we're telling it through the data key names property. All right. So, we'll select a country, there we go, there's Austria. We'll select and now we have our details. So, that's an example now of how we've broken out our code a little bit different than sequel data source because you'll notice if I go to the source, I'm not embedding any sequel statements into my page. Now those are secure from the standpoint of nobody can read that source code unless they have access to the server. But, as mentioned earlier, if that query changes, and it's used across multiple pages, it's a little bit of a maintenance headache. So, this is good. Now, if I wanted to come in and add, update, insert, delete, all that, that is fully supported with the object data source. In fact, if we come on back and I can click on my, my one for the details view. We'll say configure data source on the smart tag. And, you'll notice there's update, insert, and delete. So, if I had a method available for that then I could simply select it. Specify where the data's coming from and all that stuff we've already done and we'll be off and running. So, that's an example of how the object data source can be used to call into a custom class that has the methods that actually return the data from some type of data source.

Using the EntityDataSource

The next data source up on the agenda is the EntityDataSource. And, this particular data source control allows us to perform LINQ queries against an Entity Framework model. Now, in the previous demonstration on using the object data source, I showed how LINQ the SQL could be used. And, in this one, we're going to talk about another option that's available in. net, and one that I prefer with. net 4, called Entity Framework. And so, that's where this control is used, it can be used in conjunction with this entity framework that I'll demo. So, this control, like the others we've covered up to this point, allows us to do filtering and sorting. We can do select, insert, update, delete operations, caching, all that good stuff is available with it. Now, the basic steps you're going to need to go through to use this are shown here. So, the first thing I'm going to do is we're going to create an Entity Framework model in Visual Studio. From there, we're going to go in and then configure the Entity Data Source in our page to build a query against a specific entity, a, a table really. And then, you could from there work with parameters, filters and do that type of stuff. And, adjust this Entity data source to do what you need. So, let's take a look at how we can use it.

Demo: Binding Data with the EntityDataSource

Let's take a look at how we can work with Entity Framework and use the Entity data source to query our Entity Framework model and get data into a web page. So, to start off, I have empty web application project here and you'll notice there's a webconfig that doesn't have anything it in at this point. And then, I have an app data folder and just as a review, I added that by right clicking on the project and saying add asp. net folder. And, I simply selected app app data. And then, I added this adventure works Light data base into it. Now, let's go look what's in there real quick, because we're going to query that. So, I'm going to come up to view other windows, and go to data base explorer. Now, if you're on other versions of VisualStudio other than the Web Developer Express that I'm on. Then, you would just go to the server explorer. So, you'd say View Server Explorer, and you can get right to your database explorer from there. But, in Web Developer Express, you go this way. All right. So, that shows me the database. And now, I can drill down and actually see the tables. I can actually right click on one and say Show Table Data, which is very nice. And, this eq, bring back the select star type query. So, this brings back everything. You can see we have about 437, records in this particular table. And then, I can even drill down and just look at the different columns, this way as well, and see what their properties are, too, as far as data types and things. So, very easy to with using the, database explorer. All right. So, we're going to work with this customer table here. We're going to load customers, based upon salesperson. So, to do that, going back to the steps that I just covered previously for youth in the entity data source. We want to right click and we're going to add a new item. And, I'm going to go to data. Now, I have a couple options here. I can do link to sequel classes, or Entity. net, entity data model. That's of course what we want to do so, I'm going to call this adventure works. LT for light. Now, this EDMX is a metadata file. It's going to generate behind the scenes an XML file that ultimately will be used to generate a bunch of code for us to build a query in this database, but we're going to query it using link technology, language integrated query. Let me go ahead and add that in. Okay. Now, we have two options here. I can actually create the model from scratch. In other words, create my entities and it'll give me a blank design service if I go this route and then it would be up to me to add the entities on to the design surface. And then, I could actually generate a database from that. That is an option. Or, in this case we want to just generate from a database. We want to take that customer table and make an entity for it, so we can query it. So, I'm just going to select that. We'll hit next. It's going to ask me about my connection string. You can see it puts all that in since this is a local database. Were it not and I had to actually hit an enterprise database. I'd simply hit new connection and then pick, what are we doing, SQL Server other and then walk through the wizard here. But, it's already picked it for us, it's done the connection strength for Entity framework and it's going to update our Web. Config, Config file as well. So, let me hit next. And now, it's going to go out and query this database and bring back all the tables, views and stored procedures. Now, in this particular case, we just want to go select customer, but I would check whatever I want and would add those tables as entities on to this designer surface that you're going to see. So, it's very, very easy to get started with this. Now, I'm going to go with the default as far as including foreign key columns, naming and the model name space go ahead and leave it as is, and then hit finish. Now, what it'll do is generate the designer. And, it's going to add the customer table onto that. And, that'll now be an entity. So, it's actually creating a class for me with the different properties inside of that class. Now, from here, any of the non-nullable fields, if I didn't want those which are now properties in my, in my entity. Then, I could just remove those. I could click on this and delete. And, do that type of thing, but for this demo we're going to go ahead and leave is as is. Okay, so let me build, because it did generate some c-sharp code in this case behind the scenes. In fact, you can see that right here. And, there's quite a bit of code it wrote for us down in here so, all right, so now what we're going to do is add a web form. So, we'll right click, add new item, go back to the web and we'll do web form and we'll just call it default. And now, I'd like to come in to the web form and start to use the entity data source. Now, I already showed how their sales, people in this database. And there's customers, of course. So, we're going to do a dropdown type approach, where you pick a salesperson, and that's going to show the customers that that salesperson has worked with. So, let's go ahead and we'll add a little text here Select a sales Person. And, we'll do a drop down. And then, under that, I'd like to show the customers. And, we'll just do a standard grid view for that. Give it a little formatting. Okay. So now, we have this model that's available. And, we need to build a query the model. Now, the easiest way to do that is the entity data source, because this is after all an entity framework model. So, that's what the entity data source, data source does. So, we're going to do very much like you've seen me do several times in, in this module actually. We're going to come to the smart tag and say choose data source and while I'm at it, I'm going to check that enable autopostback so we don't have to add a button for them to click. And, we'll choose a data source. And now, we're going to pick the entity data source that you see right here. And, we'll give this one a name of salesperson entity data source. Okay. Now, it's going to go and ask us what is the name connection you want to use? Which will help drive? What is the actually what is the container that we want to query again. So, you'll notice this adventureworkslite\_dataentities. Now, that was created behind the scenes when we named ours Aventure Works Lite and it tagged on this data entities. So, we're going to be using that information to actually query this particular NU framework model. Now, since I only have one, that's pretty easy because there's only one thing to pick, you can see. So now, we only have one entity and we only have one entity set. Now, an entity set is a collection of entities. So, we have a customer entity and the entity set. They just tack an S on it. It'll be customers. So, if we come up here, there's our customers. Now, for this first drop down we want the salesperson. And, you'll notice right off the bat there's not a lot of other activities or buttons I can click here. You know, there's no filtering really, it's a little bit up here or revisit in later section but, that's about it. So, we hit finish. Now, you'll notice that nothing shows up as far as what should the dropdown list display for the data text field and the data value, the backing data for each item. So, we're going to have to do a little bit of manual coding here, very simple though. So, we'll hit Okay. All right. So, let's come into the Source. We're going to fix this up just a little bit. So, up in my dropdown list. What I'd like to do is say that the data texts field is salesperson. Because I want to make sure that we actually query that salesperson and, and, work with it appropriately. Now, here's what it added for the entity data source. So, it gave the name of the connection string in web dot config to get to. The default container is the object we're going to query through and then the entity set, we already talked about, we want to get the, from the customers. And then, we want the sales person, you'll always have this IT, dot, and then the name of the object, and sales person in this case. Okay, now let's run that as is. All right, and you'll notice that we have all our data, but. It looks like there's a lot of duplication, there's a Garrett one, Garrett one, so quite a bit of duplication. Well, I showed earlier how in the sequel data source wizard. You just check a box and you can very easily select a stink rose. Well, in this one there's not a place to check for that, but it's very easy to add. All we have to do is come into the select and add the word distinct on the front of it and that'll take care of it for us. So, let's run it again and now we should have a much smaller result set. And, there we go, of our actual sales people. Now, what we want to do, is when they select one of these. We want to show the customers down here. So, we're going to go through the same type of process for this example, and we'll choose a new data source. We'll do an entity, and we'll say, customer's entity will be our source. And, go through the same process. It picks it for us. And we'll again do customers. All right. Now in this case, I do want to select the customers, but I want to do a where clause where the customer is equal to that salesperson that we saw earlier. So to do that I am going to select all the entities and we can do insert update to lease support if we like down here. That's pretty easy to get to and we'll just select all of them and then I'm going to hit Finish. Now the challenge though and we can go ahead and drill in the here if we like. The challenges is that now as it's doing this, number one it's going to spit out all the different columns. So if I come up to Edit Columns you'll notice we don't see a whole lot it's just going to auto generate them at this point. So we'd have to specify the specific ones if we want those to show up. I'm going to go ahead and for now leave it at auto generated. But instead of selecting that show all checkbox, I could have picked very specific ones. And then another thing we're going to need to do is, I need a where clause. Well, for this particular entity data source, you'll see in the wizard there wasn't a whole lot of options there for doing that. It did come in and say hey, query from the entity set. But there wasn't really a way to do a where clause there. Now we're going to see another way to do this, in the next remaining section in this module. But for now what I'm going to do is just type the where clause in. And we're going to say, where the sales person equals some sales person parameter. Like that. Now, where's it get the data from? Well, since we have that drop down up top we of course, would like to get the data from the name that was selected from that drop down. Well, we've already done that before, we can come in and do a control parameter. Now the way we do this though, is we have to add in a where parameters tag. So we want a where clause. You can see there's update, select, order by, insert, delete, those types of things. So we're going to say that the Where parameters and we just have one parameter which is SalesPerson in this case is going to get its data from a control parameter. The name of the parameter is salesperson. Now you don't put the at symbol though. K, we need to say, all right, what is the control ID? Well, the control ID is drop down list one. So we'll go ahead and add that. And what's the data type? Well, it is a string. And finally what's the DefaultValue if there wasn't one? And we'll just say the DefaultValue is NULL for this particular example. Okay so now we've specified a where clause. But this is using link syntax behind the scenes. It actually converts this into an expression tree. And goes and queries that entity framework model through this entity data source. All right so let's go ahead and give that a shot. So you can see our data's now loaded from our entity data source query that was performed, and as we go through this, you'll see the data loads below changes. We can go on and edit, we can hit Delete which will delete immediately and we have full support for paging and we can do sorting as well and that's all built in out of the box. So this provides a very, very easy way to query an Entity Framework model. And we can do where clauses. We can even do OrderBys if we want. I could come in and there's an OrderBy and we could say, you know, I want to OrderBy maybe FirstName or something like that. Let's run that again. And, let's find first name, so there we go. So we have Andrew, Angela, Bart, Betty and you can see it's now working. So all that type of stuff can be done, and even much more, but this provides a good intro to how to you get started with the entity data source. Now, in the next section we're going to talk about how we can extend this using the query extender. So let's go ahead and move on and see how we can do that.

Using the QueryExtender

The final topic that we'll discuss in this module is the query extender. Now this is a control that was made available in ASP. NET four. And it provides a way to kind of extract out your filtering features from your linked data source or your entity data source, regardless of which you use, and puts your where clauses, your order buys or even search expressions into one little nice place. What's nice about it is by moving that functionality out, you don't have to know what the underlying data source is. So whether it's Entity framework, or link to SQL, or even something else in the future, you can do this QueryExtender with filters and not really have to know anything about what's going on behind the scenes and it'll still work to do the filter for you. So it allows you to kind of reuse things across different potential data sources. So the QueryExtender, as I just mentioned, it filters data it can be used currently with LinqDataSource and EntityDataSource. Now, that may change in the future but that's where we're at right now. It makes it so you don't have to put the explicit where clause into the data source control. And basically allows different views of data to be shown, so an example is shown here. And we have an entity data source with an ID of EDS customers. And you'll notice down here in the query extender it has a target control ID. The target control ID is what is the entity data source or link data source that we want to target and extend in essence. So in this case we're going to use what's called a search expression. Now there's different types of expressions, there's order by expressions, property expressions, source expressions and some others. A source expression's interesting because it allows us to search one or more fields based upon some specific data. So in this example we're going to search the contact name field for the search type contains. And then we're going to get the value for that contact name, from the txtFilter, a text box. And that would allow us to then do a contains or you can even do starts with, or you can do ends with. So it provides three options there, and then you can even specify if it's case sensitive or not when you're working with it. So the other one I'm going to show in the demonstration is the order by expression and the property expression those are two of several that you can use within the query extender to basically work with a single entity or link data source. And then extend it, or filter, or sort very easily. So let's take a look at how we can use it.

Demo: Filtering Data with the QueryExtender

Let's take a look at how we can use the query extender in this asp. net webforms application. So, right now when we run this, this is based upon a demo I did in the previous section. We have a dropdown with sales people. I can pick those in Filter, but the filtering code is actually in this bottom data source. So if we go to the source there, there's my where clause. And you can see we have a where and an order by. Now, there's nothing wrong with that per se, but if I like to leave the enityt data source alone I can extend it with query extender and have it drive the filtering feature. So, I'm going to strip all that stuff there. So, now if we were to run it, would just always show every customer regardless of the sales person that was selected. So to extend this and move out the query functionality into a separate control, that it's goal in life is to filter. I'm going to add a query extender here. And we'll give it an ID. We'll just call it customer query extender. And then I have to say what is the target that this query extender is going to in essence extend, what's it going to target for filtering? Well it's going to be this guy. So you'll see there's this target control ID. And I simply give it the entity data source or the link data source that I want to extend. Now inside of here, I have a lot of different options on what I want to extend. So some of these are appropriate for more standard web forms like we're doing and some are used for dynamic data. A good example of that would be these dynamic filter expressions. Okay, now you're not going to use that in this scenario right here. If you also come in there is a control filter expression. That is also used for dynamic data. But what we can use, we want to filter based upon the value selected in the drop down list. So that's called a either search expression or property expression. Now let me show a search expression. We're not going to use that one here but let me show it really quick because it's pretty useful. We can actually go in and give it a search type of contains, ends with or starts with. So if you've done a SQL query before where you use the percent symbol to say starts with or ends with. Or you can use that for a contains, if you put it on both sides of a word. Then we can do the same type of thing with a custom search expression. And that makes it really nice and easy, whereas with the where clause up here, that's actually quite a bit more difficult to do. Using the search expression though it's very flexible, I could say for instance where it, you know ends with some particular value. And then I can come in and get that value even from a control or a cookie or whatever I'd like there. So that's what a search expression can do. Is allow us to have a little more flexibility on how we search. Now on this example though I'm going to use a property expression. And the property expression's very simple. You simply say okay well where do I want to get the value for this property and what's the name of the property? Well the name of the property I want to filter this by its sales person. So we'll come in and we'll do a ControlParameter for our property expression. The control ID is going to be our drop down list. So I'm going to grab that guy right there. And then, really all we have to do other than that and say what is the property based on that drop down list value that's selected. What's the property from my data source that I want a filter on, and that is salesperson. And that's it. I'm done and that's how easy it is to work with the control parameter and use that within a property expression. Now what's going to happen is, because the query extender is attached to our customer's entity data source. This property expression's going to be used to do the filtering. So we can kind of abstract out the code that filters outside of this guy and we could technically use it in multiple ways that means because you can also programmatically work with the query extender. So let's go ahead and save and run this. And you'll see I have david8 right now so let's see if it worked right off the bat. We'll come over to our SalesPerson. You'll notice they're all david8 down this column. Now let's go ahead and do another filter here. Let's do this last one. Chose zero. And you can see chose zero is now filtered. So really what we've done is just moved out the query functionality from the any data source. Now if all you we're going to do is a simple where clause you could certainly just do it right here but there's actually a little less code involved. I can do multiple queries in here as well if I have multiple parameters I want to filter on not just salesperson but also maybe last name or something, that I can do that as well. So that's an example of how we can use the QueryExtender to do a filtering operation. Now we can also do orderbys. So previously up in the EntityDataSource, I had this order by property. And I said IT dot and I gave it the name of the property there. Well if we'd like to use a query extender for filtering, we might as well use it for sorting as well. And that's easy to do. We can come in and we can do an order by expression. And this one's very simple. Simply say the name of the field. We'll say we want to sort by last name and then come in and say the direction. Now we can do ascending, descending. Let me just show descending, so we start kind of Z back down to A. And that's it. It makes it very easy to work with. So, if we run this now. We should see that our records are now sorted by last name. You can see that it is descending or likewise, I can go ahead and switch that around. And it'll do that for every item. You can see we have V, M, L, K, so it looks like it's working properly there. So that's an example of how we can use the query extender to filter by properties or to do order-bys, or I showed a little bit on how you can use a search expression to do starts with, ends with or contains.

Summary

In this module we've talked about how we can get data into and out of our asp. net web form applications. We've seen that ASP. NET controls provide built in support for data binding, we've looked at a few of those drop down list grid view details view. And we've also talked about some of the syntax that you might come across when you're binding data so we briefly talked about the percent pound or hash. Or you can do an eval to write out a field name. I showed you on a few occasions how connection strings can be used. That's an expression with the percent dollar to pull a connection string from web. config. And then we also talked about several of the different data source controls including the SqlDataSource. The object data source and the entity data source as well as how the query extender can be used. Now, throughout that whole process, we've looked at some of the different controls and seen how they really do save us a lot of work once we get our data, we could just simply bind those controls through the data source ID. Up to our data source and then they handle rendering the HTML and doing their thing there. So, a lot of great stuff there. We have even more coming up in the next module. We'll get more into data and talk about some other options.

Entity Framework

Introduction

Hi. This is Dan Wahlin with Pluralsight. In this module, we're going to talk about the Entity Framework and explain what this framework offers us for data access. Now, I'll first introduce three different roads you can travel with the Entity Framework and we've seen a little bit of this up to this point in the video series. But we're going to switch gears and focus on a new feature as of Entity Framework 4. 1 called Code First. Code First provides a great way to write classes that can just interact with a database without a lot of plumbing in between. So, we'll talk about what Code First is, how to create Code First classes, something called a DbContext, and then how to create a database. And then from there, we'll talk about how to handle change, how we can do LINQ queries against our DbContext in query or database, and even how we can filter our data. So, we're going to start off with what is Entity Framework as a whole and how can we use it.

Introduction to the Entity Framework

Microsoft's Entity Framework provides a great way to get data into and out of our ASP. NET web forms applications. So in this section, we're going to talk about what is Entity Framework and what are the different options that are available to me and you as a developer. So let's first start off by answering the question, what is Entity Framework? Well, you've already seen it a little bit up to this point. We've used the Entity data source and we've even created a simple Entity model to query a database without writing much code. Well, we'll get into a little more details in this section, so let's first off talk about what is Entity Framework. Well, it's an Object Relational Mapping framework or you'll often just hear people say it's an ORM. Now, there's a lot of different ORMs out there that are popular, but Entity Framework is available in the. NET Framework and Microsoft has shipped, at the time this video was made, a new release which is Entity Framework 4. 1. And we'll be talking about some new functionality available with this release. What an ORM does is it allows us to take objects and persist them back to the database. And that's great when you're doing inserts updates to leads. But it also provides a really easy way to select data from the database. So, oftentimes the standard workflow, if you use traditional ADO. NET code, which is also an option that you can choose, is we'll write a query and we'll do that with a SQL command as an example. We'll create a connection to the database. We'll get back the results and then we'll iterate through the result set and fill custom objects and fill the appropriate properties. And that takes quite a bit of code to do that and it's not exactly the most fun code to write. Well, with Entity Framework, I can have it generate the SQL for me, query the database, and then also have it map the result sets back to my strongly-typed object. So if I had a customer table in the database, we can query it, get back that data, and fill, have it fill customer objects. And as I mentioned, we can also persist changes to the customer object or objects back to the database. So it will also generate insert, update and delete statements. Along the way, for performing different actions such as, maybe we're batching together inserts and updates, it'll also do transactions. So it has implicit support, built-in support for transactions to roll back any issues that might occur. So, it's a great way to go. It's a very, very productive way to go if you need to get data into and out of your apps and let's face it, that's typically what we do. So, let's take a quick look at the different options available to us as developers in the Entity Framework. Now, when you use Entity Framework, there's three options you can choose. The first is Schema First, where you have an existing database and you want to take those tables and then create classes from those tables. And that's Schema First. The database schema already exists. We also have Model First. This is where you create an Entity Framework design or diagram. And this is something you do right in Visual Studio using the wizard. And then you'll add the entities that'll ultimately become the tables in your database. So you'll right-click on the designer, you'll add in your entities, and then you'll create your database once you're done. So, a little bit backwards approach than the Schema First, where the database already exists. Now, the one that's new in Entity Framework 4. 1 is Code First. Now, this is actually my preference nowadays because this is a very code-centric approach. It allows us to write classes like we normally would do with simple properties in them, POCO classes, Plain Old CLR Objects, that we can then persist to a database using something called a DbContext. Now, with this model, it's much like Model First except we'll actually write custom C# or DB classes and then we'll use those classes to actually automatically generate the database and those tables we created for us. So we'll be talking about that throughout this module. Now, Schema First and Model First certainly warrant some attention though. It's very possible you might be in a company where you have a DBA that creates the database and that's not something you do. Well, with Schema First, we can go through and use a wizard to easily take what we have with our database and then reverse engineer that back into actual classes that map table to class and property to field. So, the steps to do that are shown here. You'll add an ADO. NET entity data model into your project. You'll then go in and select the database, as you see here. From there on the next screen, you can select your tables, and you saw me do this again in an earlier module, and then you can write some LINQ queries against those entities that get generated. Makes it very, very easy to work with. Once you're done walking through that wizard, then you'll get a database diagram similar to this. So, you'll see we have our entities. We have relationships, what we call navigation properties, between the entities and now we can query this using a LINQ query or we can even use a lambda expression if we'd like. In the remainder of this module, I'll focus on the Code First approach. And we're actually going to see how to write classes that can ultimately generate the tables that get created in our database. So let's jump into that and take a look.

Code First

Code First is the newest way to use the Entity Framework to query a database or perform different types of CRUD operations create, read, update and delete. So, what Code Frist allows us to do is interact through the Entity Framework with a database and automatically select result sets and fill our target objects, our classes that we write. Or even push our classes back to the database for processing as far as the data anyway in the classes for insert, update and delete scenarios. So, Code First itself relies on something called a DbContext. And the way it works is you'll create a class, such as an Employee class, and you'll add in the different properties that you'd want for that class, first name, last name, those types of things. And for this example, let's say that employees can have one or more time cards associated with them based on date. Well, these classes are just normal Plain Old CLR or POCO classes. They're just normal classes with properties. Nothing fancy, no custom attributes, no custom inheritance going on or interfaces or anything like that. They're just very standard. NET classes, which makes it nice. Well, these classes can interact with a DbContext class that we'll write in the upcoming demonstration. And ultimately, as they interact with this class, we can create a database based upon the Employee class and the TimeCard Class, and have something like an Employees table and a TimeCards table automatically generated in this database. So, with a Code First approach, you write the classes based upon what your application needs, which makes sense. Then you'll add this DbContext class and the Entity Framework that's supporting behind it automatically generate the database tables. And then, from there on, it just works. It's a very simple, and it's very clean once you understand the fundamentals of it. So let me show you a demonstration of how we can get started installing Code First with Entity Framework 4. 1 and how we can start writing our classes and our DbContext.

Demo: Installing Code First With Nuget

The easiest way to get started with Entity Framework and Code First is to get NuGet installed. Now, NuGet is a package manager. It's a way to get modules, including different types of files, DLLs, configuration, all that stuff into your project. So you can either go to nuget. org, you can download it from here, or the easiest way to do this is simply go up to Tools > Extension Manager. And go to the Online Gallery and search for NuGet, N-U-G-E-T. So you'll see I already have it installed, came with the version I have of Visual Web Developer Express, but I can go out to type NuGet in the search, go there, and there would be a little install here. Now, once you've done that, there's a couple ways to run something called a package manager using NuGet to get in this Entity Framework Code First feature. 'Kay, one way is I can right-click on References and say Add Library Package Reference. I can go online. This will go search NuGet, where these packages are available, and then type EntityFramework. All right, there it is. If I hit Install here though, you see I get this message. It says the Operation Failed, the package contains some PowerShell scripts, and I have to use the Package Manager Console. Okay, so no big deal. What we'll do is we'll go to Tools > Library Package Manager, and we'll go to Package Manager Console. Now, this uses PowerShell and you can see I've already typed in the cmdlet for PowerShell. Install Package is called a cmdlet that we can actually run to install Entity Framework. So we simply Install Package called EntityFramework and hit Enter. Now, what this'll do is automatically download all the stuff we need, it's already done it, to get Entity Framework 4. 1 going. And it actually puts references into our project, there's EntityFramework right there, so that we can now work with this Code First technology. Very, very nice and easy to work with. So let me show you what we can do with Code First.

Demo: Using Code First

Now that I have Entity Framework Code First available in the project, I can start to write some code and you're going to see I can generate a database from the different classes that I write and ultimately, my classes will match up with tables in the database. So I've already created two classes. I have one called Employee. You'll notice it has an ID, has FirstName, LastName, Department, and then a collection of TimeCards. I also have a TimeCard class. It also has an ID, SubmissionDate, and then the different hours for each day of the week. So what we can do now is we can provide a way, using Code First, to be able to query against a database, but the problem is we don't have a database at this point. Well, with Code First, you'll do what it says, you'll write your classes first. And then you'll use those classes in something called a DbContext to ultimately generate the database and the associated tables that map to those classes. Now, we're going to cover the fundamentals here. You can actually do some pretty cool stuff with table mappings in Entity Framework Code First. But to get started, what I need is something called a DbContext class. So, because my project's called TimeTracker, I'm just going to go ahead and add in a new class. And I'm going to call this TimeTrackerContext. Or let's just do DbContexts so we actually know for sure. We'll call it DbContext. And we'll add that. Now, I'm going to derive this class from another class that's part of Entity Framework, called DbContext. So let me resolve that, and because we already had the reference added by the NuGet package to Entity Framework, we can simply resolve System. Data. Entity as the namespace and we're ready to go. So what I want to do is say DbSet and I want to make a property of DbSet of Employee or DB set of TimeCard. And the way we do that, we can do our properties, do DbSet of Employee, and we'll call this Employees. And then we'll do the same thing for TimeCards. And we are done with a very simple DbContext class. Now, the way this works is you need a connection string though for a database, of course, and we don't have a database yet. Well, what'll happen is when you run the DbContext and query against it, which I'll do in a moment, it'll actually use a web config connection string that you define and then go out and make the database based upon your Code First classes. So what we need to do is go into Web. config and add in our connection string. So let me open up Web. config, and we'll come in and simply add a connection string section. And then we'll follow the standard ASP. NET way of adding connection strings, which is to do Add. We're going to do name equals and the name is going to equal our DbContext class. Now, that is something you can change and modify, but that's what it does by default. We need a connectionString, and for the connectionString, I'm going to use SQL Express. So, I'm going to use my SQL Express instance and we're going to say server equals. We want to use my credentials to log in, so we'll say integrated security equals SSPI, or you could do true in this case. And we'll say database is, we'll just call it TimeTracker. And we'll go ahead and save that. Now, the other thing that we need to add is the type of provider because Entity Framework is capable of working with more than just SQL Server. So we can come in and do our provider name. And this is just the namespace of SQL Server, so System. Data. SqlClient is what we can put there. Okay, so now we have our connectionString, we have our DbContext, and then we have our two classes that ultimately, we'd like to create tables for those. But what I'm doing here and what's nice about Code First is I'm writing the classes that my application actually needs. I'm not basing it on what somebody else thought I might need from a relational database standpoint. And in certain situations, this works very, very well. So, now to use this guy, I need to be able to come in and hook up a page and run it to this DbContext. Now what I'm going to do though, is instead of putting any query methods in the DB Context, I'm going to create a class I like to call Repository. And we're just going to call this TimeTrackerRepository. And in this class, I'm going to add a public method that returns a list of Employee. We'll call it getEmloyees. All right, now from here. We can go in and do a very simple query to actually get the employees return those back, and then I'll bind that with an ObjectDataSource. Which I already have that code ready to go. And so we'll come in and say return. And we want to say for each employee from employee in. and we need a TimeTracker class. Well, we have one, but I need a new instance of it. So right above here, in case we have other methods, I'm going to create a context variable. Just like that, and now we can do from each employee in are context employees. Notice that's reading the DB set I just added. Then, let's go ahead and return that as a list. We're going to wrap this. And we're going to come and say select each employee. So a very simple LINQ query here, ToList. And there we go. So we had, now have a way to query against our DB context, and get the employees and return those out. So, let me go ahead and do a quick build to make sure I'm not having any typos. Looks good. And then what I've already done is I've already hooked up a grid, and the grid has an object data-source, let me show you the configuration for that. That simply calls into a repository which we don't have that one, but we do have a time tracker repository. And there's my GetEmployees. And then we'll hit finish, and we're ready to go. Now I'm going to run this, you're not going to see anything, because not only do we not have a database, but we also have no employees in the database, but you'll see I don't get an error or anything. All right, well let's see what happened here. I'm going to go to the data connections tab, and we're going to add a connection. I'll change this to go to SQL server, and I'm going to do dot slash SQL express. Now I'm going to come down and notice I now have a timetracker database. So let's test that, looks like it works, hit Okay, and drill down. So here is what it added. It automatically added an employee's table for us based upon our different properties that we wanted, and then it added a, you can see, time cards and it even has an employee id for the primary foreign key relationship between employees and time cards. And it did that because I had that list of employees in the time card class. Now, it also does a little bit of tracking in this EDM metadata. So, each object gets a hash. And this is how it knows if things have changed. And so if we do Show Table Data here, you'll see that it generated for what I have currently this hash, and if things change it knows that we need to throw an exception or deal with creating a new database. Now there's ways around that. But that's what it does by default. It's kind of how it detects changes in your objects. So now what we can do is we can go in and add employees into our table. So let's just do an employee or two real quick. So we'll say first name, and let me show you also that when we go into properties here that we have a, is identity. There it is. Is true. And so this is an identity or a, a key. Automatically took care of that for us. Pretty nice. So we'll go ahead and just do some names here. And for Department we'll say Finance. We'll do Jane Doe is in HR. And we could add as many as we'd like. But we'll go with that. Now let's rerun it in the browser. And you'll notice I instantly get. Records back. Very cool. The reason I like this is because with Code First I'm now able to actually take a code-centric approach. Design the actual model classes, is what I like to call them, with the properties. Then hook up my, DB context, and then we can simply write queries against this DB context, which we did with our repository. We wrote a really simple one here, but we could do filtering and all that good stuff, by simply querying the DB context. Makes it very, very easy to work with. So that's an example of how you get started with Code First.

When Classes Change

As you work with the Code First approach, you'll certainly run into situations where you need to make changes to your classes. So how's that affect things? Because the classes we know are tied ultimately to the database tables. And if you make a change to the class, how's that going to affect things? Well, Code First does offer a way to change your mappings, so if you want to change names that's certainly possible but, what if you add for instance, a new property into the class, that doesn't exist in the database? Well, we need to be able to deal with that. So, in this section I'm going to talk about how I can handle changes, and explain how the process works with the Code First model in the DB context. In a previous section, I showed how you can create an employee class that had properties like ID, first name, last name, department, and then map that over a table in a database. And as was demonstrated, I didn't actually have to create the database. It did all the heavy lifting for us, with Entity Framework Code First. So the database itself had an employee's table, and the timecards table. And then it also had this EDM metadata which I mentioned earlier, provides a hash representing our model. Our model classes such as employee and timecard. And stores that, and what this is used for is tracking changes. So let's see how this comes into play. Let's say that down the road in my application or even as I'm in the development life cycle, I want to add a new property. We'll say, we're going to add a hire date. All right, let's go ahead and build that. And let's run the page. And let's see what happens here. You can see nothing good. We actually get an error. But it's makes sense because our database no longer matches, the hash is now different, so we need to figure out what's going on here as it's doing the query for employees. So you can see the air is pretty explicit. The model backing our db context has changed since the database was created. It's basically said that hey everything's not matching up. Things aren't good. We're going to have problems, so we get an error. So couple of options in are we could come in and delete and update our database ourself, we can do what I'm going to show you, and we can use this thing called database dot set initializer to drop create database if model changes, and I'll show you how this works here. What we need to do is ultimately set up in one place this database set initialize, and we need to say what happens in this scenario. Where our model might have changed, such as the hire date field, or property getting added. I'm going to go off to our solution explorer, and one place we could add the single database dot set initialized call is in a file called global ASAX. So I'm going to add a new item, we'll scroll on down to. Global Application Class. We want to leave it that name, and if you haven't used this before, this is a class that gets hit the first time when an application starts, when a session starts, or even when any request comes in. And then the same thing for Session End, Application End. All that type of stuff. What we can do is in Application\_Start, this only gets called once and we could do Database. Set and then we can do the set initialize. Well, we need to resolve the namespace for our code first. So I'll go ahead and do that. Notice that goes to our, kind of our aqua color. And we'll say set initializer, and then you'll notice that we have this option here to do a drop create data base if model changes, and you'll see I actually have something else. So what we're going to do is specify what's called the strategy for how to handle changes. So that when our model changes, our database can automatically be kept in sync. So I'm going to create a new instance of this drop create database if model changes. Alright. So what'll happen is, you could say hey anytime you run this. Let's always recreate the database. All right, Drop, Create Database always. Or we could have it where it's going to drop the database, if only the model changes. Otherwise, it just leaves it alone. Now, obviously that can be. A little bit scary if you're doing this. But in development mode, this is great because we can set it and automatically have it handle things. So I'm going to go ahead and select this guy. And then, in the generic here, we need to supply our DbContext. So, our DBContext, going back over is our TimeTrackerDbContext. Let's go ahead and add that. And let me just right click on that, we'll Resolve the name space. Okay, so now what's going to happen is, if there's any changes that occur it's automatically going to drop that database, and then recreate it based on the new model classes. So we should pick up our higher date, as an example. So, let's go ahead and build this. We'll re-run the page, now keep in mind we're probably not going to be too impressed with the data because. Now it's going to delete the data in the database. You'll notice it's taking a little bit longer here. And once it returns, we're won't see too much but we'll go check out the database, and then I'm going to show you another technique that can be used so we can get some sample data in there. Okay, so there we go, it loaded. Not real impressive because we don't have any data, as I said it wiped out the database. It dropped it and then re-created it. But let's go take a look at what it did. I'm going to come back to my data, database explorer. Let's come on in and do a refresh. I've already disconnected from it so we'll refresh here. Come into employees and you can see hire date, is now a column in that database. And if we go to the Properties you'll see it's even a daytime column based on my property. So that's pretty nice. And that provides a really easy way to handle changes. While this option works it be pretty nice if we could go in and add some data. So let's take a look at how we can enhance this a little bit. So, for now, I'm going to go ahead and comment out this particular line of code, I'm going to come back over to our solution explorer, and in my models, I'm going to add in a new class. Now this is class is going to be responsible for setting the strategy for what happens with the database, but I'm also going to be able to seed. Some sample data into the process. Basically add in some sample rows so that we have some data to develop with. Makes it a little bit easier to work with things. Now of course you don't have to do this. You could always once the database is recreated you could certainly just run a sql script which does some inserts, and that would be an option as well. But this will show off a little bit more of. What entity framework code first offers us. So, I'm going to come in, and we're going to create a time card DB context initializer. Not the shortest name in the world, but let me go ahead and copy that into the clipboard. All right, and what I'm going to do is actually drive this class from the drop create data base if model changes. So, I'm going to go ahead and resolve the name space early, here. And now we'll go ahead and, do drop create data base if model changes, and we'll give it our context. So we have our TimeTrackerDbContext. Okay, so now I have a class that actually is going to be used to override some of the default behavior for DropCreate if model changes. And so what we'll do here is we'll just type the keyword override and hit space. And you'll notice there's really only one thing useful to override, and that is this seed. Well what seed allows me to do is seed this with some data. So, I'm actually going to come in, and we're going to seed this with some new data. So I'm actually going to take that out. And now what I'd like to do is I want to create maybe a couple employees. And so we can actually go in create a new list if we wanted. And then we could iterate through those. So, let's go ahead and say IMPS equals new list of employee and we're just going to go ahead and in the employee here, fill in some data. So we'll say new employee, and we're going to go ahead and add the properties. So let me go ahead and do that, and then we'll continue on from there, and through the magic of video. We've now added in two new Employees with two TimeCards per Employee. So now that I have Employees we need to get them into the database. Well fortunately we have a DbContext object that we can use. So all I need to do is add each of these Employees into our DbContext. And we can do that by calling an add method. So, you'll see if I come down I can do context dot, we can come down to Employees. Add but I of course would want to iterate through these. So, let's come in and we're going to do just a simple, for each here and which will pass us in an employee, and we're going to do what's called a lambda expression. So, we're going to pass in each employee. Name at e, and then we're going to say okay, as each one's passed in let's go ahead and do context. employees. ad. And we'll add in that employee. Now, once the employee's been added in, then we need to go ahead and actually, update the database. So, if you recall I took out a Base. SeedContext. And so, we're simply going to pass along the context that now has our two employees with the two time cards each into the base class, which will be this guy. And, let it take care of updating these for us. And now, when we run this let's see what happens out of the box. So, first thing I'm going to do is just run the page to cause this Global. asax to run. But, you'll notice that number one, if, you won't see anything here because, nothing ever happened. Number one, there is nothing in the database still. And, I'll show you why, still null. Well, if we go back to the Global. asax remember I commented that out. So, we're going to comment it in, but instead of passing the built in code first class. The DropCreateDatabaseIfModelChanges, we're now going to create a new instance of our TimeCardDBContextInitializer, which of course inherits from that other base class. So, what we can do now is Build, and let's run it one more time and let's see what happens now. Okay so we still don't see anything, let's go back over to the database, let's refresh this. And, we'll do Show Table Data, so what's going on? Well, remember, this particular class is designed to drop and create the database if the model changes. Now, we didn't change the model, so nothing really happened there. Now, we could do that always if we wanted, but that would always delete everything in the database and then reseed it, with this data here. So, I'm going to go ahead and just get our database with some data. I'm going to come in and just add one more property. And, let's say that we want to track the company ID, which we'll say is different than the unique identifier, the ID up here. Let's go ahead and build that, and run it. And now, you can see, it must be dropping, because it's definitely taking a while. So, now it's actually dropping the database, it's hitting our seed. Our seed should add two employees and two time cards that should automatically fill those two tables. And then, we'll hopefully get back some data, once this is done here. Okay now, let me go ahead and I didn't even mean to show this but this is a good, demonstration. Cannot drop the database, it's in use, why? Well, remember I was over here in the Database Explorer, this is still open. So, I'm going to go ahead and right-click on it and say Close Connection. And, that's something that definitely I've run into quite a bit. Because you'll be going and looking in the database making sure your seed took and you can see what happens when that's open. So, now you can see it's read, it's closed. And, let's run it one final time, let it try to drop it. And, there we go. So, you can see it added our John and Jane Doe, there's our departments. We can go look in the database now, let's open it back up. We're going to do a refresh. There is my CompanyID by the way, shows up. And let's go ahead and say Show Table Data, and there we go. Likewise, if we go over to our TimeCards, and say Show Table Data, and you can see we now have four time cards. So, perfect. So, that's an example of not only how you can just do the basic, DropCreateDatabaseIfModelChanges in our Global. asax. And, how we could feed that into our Database\_SetInitializer but now, if we do change our model, it will automatically update the database. It'll drop it, and then put that test data back in, so it provides a nice little feature that you can use.

LINQ Queries

Now that you know some of the techniques that can be used to handle changes with classes, let's take a look at how we can use our DbContext to perform LINQ queries, and some of the different features that are available with Language Integrated Query. Language Integrated Query provides a great way to perform queries against objects. Or, in the case of Code First, we can query a database. And, you've already seen me do some LINQ queries. But, we're going to talk a little bit more about it and what features are provided. So, a LINQ query is kind of like writing a SQL query if you've ever done that before. It's a little bit backwards looking, but it really does a very simple thing for us, and that is it generates a SQL statement. This is a very simple SQL statement it would generate in this example. In other examples, though, it can do very complex type things. And so, ultimately, it boils down to its very productive once you learn how link, lambda's and something called extension methods work. So, we have a LINQ query here, and you can see that were going to say from e and we can name that whatever we want. We can call it imp, we call it foo for that matter. In Context. Employees, let's go ahead and orderby LastName, and notice e is used there. And then, let's select that employee. So, it's going to select all the employees, as it does that each employee goes into e, it's going to do an orderby, and then select it. Now, if you actually turned on something like SQL Profiler, which is a way to watch your SQL queries as they come into the database. You'd see it and this actually generates a SQL statement that has an orderby clause. And then it would automatically fill those into our list of employee here. So, I'm actually taking this query, which returns an object called an iqueryable, and iqueryable is an object that you can even continue to query. I could pass it out of this method as an iqueryable. And another method can tack on another query to it, but in this case I don't want to do anymore queries. I just want to say hey, go ahead and get this data from the database and let's get it now, and put it into a list. So, that would be an example of a LINQ query. Now, there's different ways we can do this though. I could come in and also do a lambda expression. And, lambdas in many cases are actually my favorite way to do it because often times the syntax is more compact. So, a lambda would look something like this. We'd say Context. Employees. orderBy and then what I'm going to do is I need to say what to order by. So, it's going to let me pass in each employee. Again, I'm just going to name it e, but I could easily put foo there, it doesn't really matter. And then I could say let's order by foo dot. And, the reason it knows the different fields is we are after all, doing an order by on employees. So, the order by actually knows the collection we're ordering and that's how it's strongly types. So, we could say foo. LastName as an example to list. So, that would be an example of something you'll definitely see a lot with ASP. NET Web App especially modern ones where we could use a lambda. Now, I probably wouldn't put foo, I would do something like e or m for you know, whatever you like there. Now, I'm just going to comment that one out, but that is a lambda expression. Now, another nice thing you can do is aggregates with length. And an aggregate is a way to do things like min, max, average, count, things like that. So, let's say we wanted to also have a method here that returns an integer, and we'll call it GetEmployeeCount. And, in here, all I have to do is say, return Context. Employees. cou. Now, you'll notice this kind of, what I call the red running box. Normally, methods just have the box. But you'll notice all these guys, except for well like SqlQuery here does not, have this little arrow pointing down. That's an extension method, any time we see that. So, we'll say Count, and then return that out. Now, we could also go in and we could do averages, we could get min, we could get max, we could do all types of things, that way as well. To do that type of thing, we could go in for instance and say, let's select the employee and let's grab the Employee, let's grab the ID here. And then, we can say hey let's go get the Max of that, and return it. Or, maybe from all the employees that are returned. Let's go ahead and grab the Average of those, makes it very very easy to work with. Now, Average the reason it's there right now, you can see it returns a double, because of course Average you might get some decimal points. And, we could do Min, we can do Max, we can do Count, we can do all that type of stuff. So, this is another lambda expression so we're going to say go Select the ID from our Employees, and then do the Average. And that would be an example there but, let's go ahead and we'll just change it back to Count. So, this shows an example of using extension methods. We have an example of doing a lambda, and then we have an example of doing a LINQ query. So, that's a quick overview, but that'll definitely help get you started with this because this is the type of stuff we'll be doing against our Code First DbContext.

Filtering Data

In this final section, we'll talk about how we can use LINQ queries to filter data. Very, very simple process and very powerful, because the data is actually filtered at the database level. It doesn't actually query the database, grab all the employees with time cards and then filter it. The query will actually run in the database which is very efficient. So, let's see how we can filter data with our LINQ queries. LINQ provides a great way to query data in a database and then fill our objects without having to write a lot of code. In the previous section, I showed how we can use LINQ to not only get our data, but we can also sort it, and we can do aggregates. And, I even showed how we can do lambda expressions. But, one of the things you'll also do quite often is you need to filter the data that comes from the database. But, typically we want to do that in the database. I mean, that's what the database is good at, and very fast at. So, let's say as an example that we want to be able to go grab time cards based upon the submission date. So maybe, we want to grab time cards from last week. Let's say maybe Fridays are when we submit those. So, to do that, we can come in and say, we want to return a list of time card. And we'll say, GetTimeCardsByDate. And then, this will pass in a DateTime object. And, we'll have, we'll call it a submissionDate. All right so we'll go in, and now what we're going to do is very similar to what you see up here. We'll do a query, but this time instead of just an order by, we're going to add a where clause to do a filter. So, we'll simply return, and I'm going to say from tc in our DB Context. TimeCards collection. Where the tc. SubmissionDate equals our submission Date that was passed in, select all those time cards. And, then we'll simply convert all those to a list. That means execute the query right now, and return it back. Now, what this will do is I've already mentioned a couple of times is it actually generates the SQL and it embeds the where clause in the SQL that is actually generated and run at the database, so very, very efficient from that standpoint. Now, if you wanted a filter based on other things, let's go grab all the ones from a certain date where it equals that date and, where maybe the time card Monday hours is greater than ten. We want to see everybody that had overtime or something like that, then you can do that as well, we can do ors. So, all that's available when we're doing our filtering and then I can even come in and then sort, based on a particular field if I'd like. As an example, we could say orderby, maybe the ID of the time card. And, so what that'll do is we'll go in, we'll get all the time cards, it'll be by a specific date, though. And then, we'll go ahead and return those time cards out. very, very easy to work with and something that's really nice to work with when we want to filter in that type of a manner. So, that's an example of how we can do a filter expression with our LINQ. So, to wrap up, let's go ahead and add one more filter method and then we're going to wire that up to a web page and see it in action. So, I'm going to come in and add one more that gets employee time card. So, we'll pass in an employee ID, we'll do a where clause very similar to what we had here, and then we'll kind of go from there and work with it. So, moving in, we'll say return list of TimeCard again. And, we'll call this GetEmployeeTimeCards and we'll pass in the ID of the employee. Okay, so we're going to do another LINQ expression. And then, with that we'll do a where clause very similar to what we did up here. And so, we'll come in and say return from each employee in Context. Employees where the employee ID equals the ID that's passed in here as a parameter. Let's go ahead and select that employee's TimeCards. Now, if we just do that you'll see that we're going to get an error on the select. It says it can't implicitly convert this thing called an Iqueryable to the list that we want to do. So, right now it's an Iqueryable of list of time card. So, the way we can just get the list out of it is we can say SingleOrDefault. And, that basically says, hey grab the first collection that you have inside of the IQueryable. And so we only have one, which is all the employees' time cards, and return it back out. Now, what. SingleOrDefault also does though is, it'll return the single object, and if it doesn't have one, instead of erroring out, it'll return a null. So, you just get a null there that would be hooked up, and returned. So, if I go to Default. aspx, what I have here is just a, pretty simple example of the two grid views, two object data sources, the first object data source. You'll see here points to our repository and it is bound to the Gitemployees that was shown earlier. And then, the second one, this grid little more custom actually went into this one and did EditColumns, and I added the columns that I wanted here. And, to do that you simply hit Boundcolumn and hit Add. Then you can simply name the header of the column. And, I bound that to an object data source. But, it goes to the repository and binds to GetEmployeeTimeCards. Now, of course, that takes an ID, we're going to grab that ID from the top grid over here. And then, we'll pass that in to this particular method, and that's kind of how this one works. So, when we run it, now it's going to use our DBContext of Code First, to execute the query that gets all the employees, that orders them by last name. So, let's go ahead and right now there both Doe I believe, so you won't see much there, but we have Jane and John. And then, when you click, you can come in and see their time cards. So now, this will pass the ID in, to the new method that we just made, into our repository class and then call, GetEmployeeTimeCards. We then query using our DbContext we get back our list of time card. That's ultimately what's bound, down here. So, as I click on these, you'll see the data changes. And, we can see the different hours for the two different people. So, that's a wrap-up on this module on how to use any framework in Code First as well as how we can use Language Integrated Query.

Summary

In this module, we've talked about Entity Framework, and focused on the Code First approach. We know that Entity Framework provides three different options, including Schema First, where you'll generate an entity model from an existing database. We have Model First, where you use a designer to design your entities, and then those will be used to create the database. And then finally, we have Code First, where you'll actually write plain old CLR objects or POCOs. And, these are just classes with properties that ultimately through a DbContext can update a database and actually generate the database tables. Now, we've only scratched the surface with Code First, there's quite a bit more you can do with it. But, you can see just with what we covered, how powerful and easy it is to use. Now, Code First, we know provides a way to generate a database from our POCO classes. We know that DbContext is the central class that we use along with DbSet properties, to actually query against the database and then fill our classes with the result set, the rows that come back. The final thing we talked about was Language Integrated Query and how you can write queries. We also talked a little bit about lambdas and how you can also do and use extension methods in your queries. And then finally, we talked about how we can filter data, which is a very powerful feature because the filter, in the case of Code First, is actually done in the database. So, it's very performant, as opposed to loading all the data back and then filtering it. So, that's a wrap up on this module on Entity Framework.

Master Pages

Introduction

Hi, this is Dan Wahlin with Pluralsight. In this module we'll talk about Master Pages, and how they can be used to provide consistency as far as look and feel go across your asp. net applications. So we'll start off by talking about what are Master Pages and how do you get started using them. From there, I'll provide some demonstrations of how to create a master page, how to then use a master page in what we call content pages, and then show a nice little trick we can do as far as changing master pages out dynamically. Sometimes you might, for instance, want a printable page, where you strip out a lot of the stuff in the page and only print the actual content. So let's get started by jumping in to an overview of what master pages are and how we can use them.

Working With Master Pages

Master pages provide a way to define a common page layout across different pages in a web forms application. An example of that's shown here. We have a site called GolfClubShack. And you'll notice that we have a consistent header across the top. So we have a login. We can do searches and perform other activities. And then on the left we have a navigation menu for getting to the different products that this site would sell. Now, of course, we'd want users as they navigate through the site to always be able to get to the different categories of products, be able to perform searches, login/logout and do other activities. Now, of course, the page content will go here in the middle, in this blue area. And that'll be defined with something called a content page. So by writing a master page, you can define all of this common look and fill in one place and then all of your different pages can plugin to that common fill or template, if you will. So a little more information on master pages. So I've already said they provide a common page layout. And you can define that in a single file. That particular file, by default, will inherit from a control called UserControl. Now we've created user controls in a previous module in this video series, and master pages are actually just a specialized type of UserControl. Now you can nest master pages, that's possible. So you could have a main master page that has some children inside of it that have a different template that's nested. You can also within a master page define the content area, so that when each page plugs into the master, they can take the content they have and it'll get put in the proper position, proper spot. As you'll see, master pages just have a. master extension. And they can also have a code behind file, so programming logic written in C sharp or Visual Basic can certainly be added in. And this can be used in situations where maybe a master page defines some advertisements and you want to show or hide those based on the context of the content page. Let's take a look at how we can create master pages.

Creating Master Pages

Master Pages can be created directly in Visual Web Developer Express or Visual Studio, by simply adding a new item of typed Master Page into your project. So before I demonstrate doing that, let's take a look at what's in a Master Page. So you'll notice that I have a very simple Master Page here, that would define the overall layout of our particular web application. In this case, it's called Mysite. master. In this page, you can see that I have a %@ directive called Master. And we talked about directives earlier such as the page directive, register directive with user controls and now, we're seeing the master directive. The master directive simply marks this particular page as being a master page of course. Now we can put all the different content that we want in our master page and anything you put in here will simply get output. So in this case, I have my body tag, my HTML tag, I can have a heading here if I'd like, with some javascript files, some CSS that I want on every page. And then the key thing that you need to put though is at least one content placeholder server control. So in this example I have one with and id of ph1 and what this does is it says where the content from the different pages go, once those pages are rendered and merged in with this master page. So what'll happen is a content page will define a content area. That will match up with the id. Ph1 of this content placeholder and then at run time it'll automatically get put in, in this area right below the image that you see here. So that's kind of the fundamentals. So the big thing to grasp at this point is that master pages have the html and the body and even the head tags in them. You won't put those types of tags into your content pages because, of course, we only want to define it once. So let me show a demonstration of creating a master page and adding it into an existing web application.

Demo: Creating Master Pages

Adding a master page into your project is as simple as right-clicking on the projects, going to Add New Item. And selecting Master Page. Now for this example, I'm going to call it default. master. And you'll see that Visual Web Developer Express automatically adds in that Master Page. And really starts us off with everything we need to get started here. So you'll notice at the top we have the master directive. And then it's added the standard HTML tags that you would see in a full HTML page, because this does represent the overall template, the overall page layout that all of our different pages in our site are going to use. So we have our html tag, we have our head and we have our body. So I'm going to come in and just change the title to time tracker. I'm going to go ahead and shorten up two server controls that I added, which are content placeholder controls just to make it a little easier to see everything. And let's talk about those. So the first one, the head, this is going to be a area where content pages can dynamically inject in CSS files, JavaScript files, or anything that you want to go up in the head section. So it's optional, you don't have to use it, but it's very useful when a page might have a particular JavaScript file that page needs. But it doesn't want to, you don't want to load it up into the master page itself because, you know, the script's only used on this one page. So by using a content placeholder and putting it up in the head area, the page can actually supply that custom JavaScript file and have it go in the appropriate spot you'd like it. Once it renders to the browser. Now this one here is definitely one we'll be using a lot though. This is our main content area. I'm going to rename this content placeholder main. And content placeholder main is where pages as they have their content and that gets merged into this template, that's where the content will go at run-time. So we're basically marking a placeholder for this is where the content should go that I want to use in themcontent page. Okay, now from here, it's up to you on what you want to add, so I'm just going to real quick add in some divs, I'm going to do some in line styles and a couple floats. So we're going to float left on this one. I'm going to give it a width of 50% and we'll do a little bit with fonts here so we'll do first a text align left, we'll do a font size of say sixteen points and finally font weight of bold. And in the first div, what I'm going to put is just the title of this website that would appear on every page. So this is our header area. We'll just call it Time Tracker. Below that I'm going to do a very similar thing. Except for we'll float it right. I'm going to account for any, any padding we might need. So I'm going to do about 47%. Float the text right. And that'll do it for that guy. And then in here what I'm going to put is a ASP server control label. And this will be used to write out the current date if we'd like. So I want to demonstrate that we do have code behind files you'll see that over here. Here that can be used to dynamically influence what the master page is actually outputting to the browser. Alright I'd also like to come down under the form and let's just put kind of a fake footer here. So this would be our footer and then let me just make a quick comment up here. This would be our header. And then we have our form which will hold our main content here in the middle. Now of course, we could add a menu and we could do a lot of other stuff to this. But for this simple demo, we'll just do this. So I'm going to come in and do copyright. And let's check out design view, and looks pretty good. I'm going to add a br tag I think right below our header. But we have our header, we have our footer, and we have the area where the content's going to go. So let me go ahead and add a <br /> right here. Go back to design, and that looks pretty good. Looks good to get us started. Now, If I wanted to add images up in here, anything I want to put in the master that would be consistent across all the pages, this is the place to do it. Now the final thing I'm going to do is we're going to right click and go to view code and I want to update that label with the current date. So I'm going to say datelabel. text is datetime. now and we'll do to short date string. Just to demonstrate that you can manipulate controls inside of a master page much like you would do inside of a normal asp. net webform page. Okay and that's it. So whatever time you want to put into your master, whatever graphics, whatever headers and footers and menus you put in, these will all be used now across all pages in your site as long as they reference this master page. And that's what we'll be doing in the next section. Now to test this, we really can't quite yet at this point. I can certainly go design view as you saw me do earlier. But I can't right click in view and browser. But you'll notice I can add a content page and that's what we're going to do in the next section. So let's go ahead and now talk about how do we use this master and reference it from a content page.

Creating Content Pages

After creating the master page you want to add content pages into your site that leverages that master page and add content into the appropriate spot defined in the master page. So in the section were going to talk about how to create content pages. And it's a really easy process with either Visual Web Developer Express or Visual Studio. So I have a very simple content page here. And you'll notice that it has a page directive. But the page directive has an attribute called Master Page File. That's obviously used to point to our master page. So what'll happen is, at run time, this particular attribute defines the overall template for the page. And then we're going to define the content that should be plugged into that template. Now plugging in the content is done using our content server control. The key part of this control is, you'll notice, there's a ContentPlaceHolderID in this case named ph1 and that matches up with the content placeholder control defined in the Master Page. From there, any content you put between the start and the end tags of this content control will automatically be merged into the appropriate spot of the master page. Now as shown earlier master pages can have more than one content place holder so we can certainly add another into here if the master page had one that matches. And an example that would be the content control that's up in the head section of a content page. In this page, you might want to define some Java Script or CSS and have it added directly up in to the head section of the master page. Now other than that, a content page, it's pretty standard. It has an aspx extension, it has a code behind file and can do all the normal things that you can do with server controls. So let's take a look at how we can create content pages using Visual Web Developer Express.

Demo: Creating Content Pages

Content pages can be added into your website or web application projects in multiple ways. One way is we can right click on our master page and we can select Add Content Page. Now what this will do is create a content page that's, of course, based upon that template. And it will give it a default name here. And on the page directive you'll notice that it adds this attribute called MasterPageFile. And that points to the Master Page. You'll notice this tilde slash here and that means start at the root of the web application, the website and go down until you find this master right under the root. You will also notice that it added two Content Web server controls. The content placeholder id shown here matches up with the two content placeholder controls in our master page. So if we go back to the master. If you recall earlier I showed we had a head. And then we had our main content here. So this automatically pulls those out and now I can put whatever content I'd like. So that's one option for doing this. Now I'm going to go ahead and delete that one and show a different way, though. If I right-click on the project, I can say Add New Item. And we can go to Web Form using Master Page. Now I'm going to call this timesheet viewer and we'll hit Add. And now it's going to let me select the Master Page because I didn't tell it what to use up front. And you could have multiple Master Pages based on different views or different devices. That might be hitting your website. So, I'll call it Default. Master. Hit OK. And now this adds in the same thing we just saw, except for now I have a little more control over the page name. But it does have my Master Page file that I selected and the two content areas. Now inside of the top one here for the head, we can go ahead and drag in some JavaScript files or CSS files so let's say that we need this custom. js file but I only need it on this one page. Now if I did need it across all pages I could certainly drag that right below the title in the master page and every page would have access to that script. But in this example, we're going to assume only this one page needs it. So I'll drag it right there. The second thing I'm going to do is we are going to come in the content area. We'll say this is my main page content. We'll save it and let's right-click and view in browser. Alright so this will load up and you'll notice it worked. We have our header. We have our footer and then we have, this is my main page content. Now we could probably add a little bit of a line break there. So I'm going to come right here on our Master Page. We'll add a BR. Let's do it again. And you'll see that it should take that into account and things'll be bumped down a little better. Okay, so that's an example of two different ways you can create a content page. Now we don't have much content though. In default aspx, based on an earlier module, we did some code first with entity framework and built a simple time sheet viewer. And what I'm going to do is take that code out, so I'm going to copy the div, down to the div right here. And we're going to paste that into our content area. And save it. So what'll happen is now I have some real content in a real master page and you'll see that we get the header, we get the footer, and we get the content showing up appropriately. And now as I select different items we can view it, but we have the consistency and as I add in different pages, we'll have that consistent look and feel or page layout, across all of the different pages. So that's an example of how easy it is to add a content page in. And even work with a JavaScript file that gets added in the head section or custom controls or whatever you'd like. And one thing actually to add is if we view that one more time. Because I added the script and actually let me go to the content page and run it. Because I added the script into the head content placeholder. Let's view page source and you'll notice here's our head section, and there is the script. So normally you know, when you add a script in it will add it wherever you put it in the page. But this time, because we said put it in the content placeholder, let's go back to that page right there. That automatically put it up where that content placeholder was in our master page and you can add multiples of these. We only have two in this example. One for the head, one for the main content but you could certainly add others for a menu section that you want to dynamically input based upon the page or whatever it may be. So that's an example of working with content pages and master pages in your ASP. NET applications.

Changing Master Pages

Master pages can be changed out dynamically using a page preinit event. So, in this section, I'll show how we can make a printable version of a master page. And what we'll do is make it so a user can click on a print link. We'll switch out the master page, and that way, when they print it, only the content will show up in the printed page. And to do that we can leverage the page classes. Page PreInit event, an example of that shown here. The page class itself has a MasterPageFile property and we can simply assign a different master page to that using the Page PreInit event. Now this can only happen at this level. Any later in the page lifecycle and you won't be able to switch out the master page because at that point it will be too late and the content will start to be merged in with the master page content place holders. But, within the Page PreInit it's quite easy to simply give a different path to a Master Page and switch it out. So let's take a look at how we can do that.

Demo: Changing Master Pages

ASP. net makes it really easy to switch out Master Pages dynamically at runtime by simple handling the page PreInit event and then changing the value of the Master Page file property of our page class. So in this section, I'm going to show you how we can do that type of scenario and make a printable version of a page by making a different master page. So earlier in the module I created this simple master page with a footer and a simple header. Now, we could come in and wrap our header and our footer in a ASP panel control and then just hide each of those panels when we want to print. And that would be an option. There's actually several options to make a printable version. But I'd like to demonstrate how we could make a separate printable master page. That just doesn't have a header and a footer. So to do that I'll right-click on the project and add a new item. We'll select a Master Page and I'll just call this print. master. Now, what I'm going to do is go into Default Master and we'll grab from form up to the form, so everything in between the form text. And we'll just paste over. And I'm mainly doing that to make sure that my ID matches up. I'm also going to take out this head placeholder because we don't really need that in this example. But, if your page adds anything up into that control, if it references the head using the content control, you would want it. So for our page, we're going to do this TimesheetViewer and you'll notice we do leverage the head. So, bottom line here is before you go stripping out content controls, even if you're not going to be using that much, you want to make sure that that's not referenced by the page that'll switch out and use this master page. It needs to match up, in other words. So now what I can do is going back to time sheet viewer, you'll notice I've added a hyper link here that just links right back to itself, with print equals true. So, I'm going to go ahead and add one more line break and then let's run this as is and you'll see that as I click the link it's just going to link right back to itself. But if we click print me, you notice up on top I add a query string variable. And we don't even really have to add equals true we could even just put print up there and this would still work. Now what I'm going to do though is now that we have a print. master, I need to change the pages, master page file path to print. master. So we'll come in and we'll go into our code behind. And this is the code behind of course of the page that we want to work with. And we'll go in and add a page pre emit that takes an object sender and it eventArcsE. And now, when PreInit is called, I want to switch out the Master from Default back to Print. Master. So that's pretty easy to do. TimesheetViewer inherits from Page. Page happens to have a master page file. So we'll simply say, MasterPageFile equals and we want to start at the root of the application, so I'm going to put a tilde, slash and then we'll go ahead and give it print. master. Now if we do that currently, though, let's run it as is. We'll go to, time sheet viewer here and you'll notice it automatically stripped off. I mean, it did what we want but we didn't click on the print me yet and so what's happening right now is it's automatically switching it, no matter what happens, to print. master. So what we want to do here is say if the request, query string, has inside of it, there's an over low here for the string name, has print available, if that does not equal null. Then we want to go ahead and switch out the master page. So let's go ahead and build that. Looks like we're good, we succeeded. And now let's go head and run time sheet viewer. All right, so you can see that we got our default header and our footer. Now when I click print me that query string variable will show up, and you'll see that we're now on a printable page. And now when I click on things it stays on that master page. Because of the presence of this print equals true. Now we could have certainly checked and if print equals true but, in reality, even that will work. Just doing print. So that's an example of how we can dynamically change out Master Pages by using the Page\_PreInit along with the page class's MasterPageFile property.

Summary

Master pages provide a great way to add consistency across pages in a website. They allow us to define a template that ultimately has our page layout and then we can apply that page layout to content pages. So we started off by talking about what master pages are and then I showed how we can create master pages in Visual Web Developer Express. We also talked about the role of the master directive and content place holder controls for saying where content goes in the overall page layout template. From there I showed how content pages can be created and used in our application and hooked up to a master page. And how content controls ultimately have the content that would go into the Master Pages content placeholder areas. From there I showed how we can switch out Master Pages dynamically using the Page PreInit event. By using Master Pages, you definitely will save yourself a lot of time and provide a simple way to add consistency across pages in your web application.

ASP.NET AJAX Controls

Introduction

Hi, this is Dan Wahlin with Pluralsight. In this module, we're going to talk about one of the more fun and one of the more popular web technologies out there and that is AJAX. Asynchronous JavaScript and XML. Happens to be one of my favorite technologies. Something, I use a lot and something that you'll run into a lot out there on different websites. So, we're going to start off by talking about what AJAX is, and some of the different key technologies and terms that you need to know to get started with it. From there, we're going to jump right in to, what is ASP. NET Web Forms, have to offer. And there's several different controls that are available that you can use without having to be a master of JavaScript. In fact, you can drag and drop these controls and instantly AJAX-enable a page, without having to write hardly any code. So, we'll talk about controls like the ScriptManager, the UpdatePanel, the UpdateProgress, and explain how these three core controls can be used together to provide a nice, and simple AJAX experience. So, that if the user does click on a button or performs some other type of action, it doesn't reload the whole page, we can load parts of the page. So, let's go ahead and jump in and talk about some of the different AJAX Technologies and terms that you need to know to get started.

Introduction To AJAX Technologies

In this section, I'll introduce several of the key AJAX Technologies, terms and acronyms that you might hear or run across as you're integrating AJAX into your Web Forms Applications. So, we'll start off by talking about some of the terms, clarifying acronyms, and then I'll jump into a more formal discussion of what AJAX offers us as developers. So, to get started let's talk about traditional Web Applications and what they offer. Normally, if you go fill out a form, and you click a button, the data from that form is posted back to the server. And we call that a Postback Operation. That data is then processed, and then either the same page or a different page is reloaded in the browser. So, the entire page is refreshed as we submit our data. Well, with AJAX, it's kind of opposite. We don't want to refresh the entire page. And so we want to do something called a Partial-Page Update. And that's where we only update maybe a section or a part of the overall page as they click a button, click on a hyperlink or integrate with a menu. So, Partial-Page Updates are designed to allow us to get the data we need into the application and display it to the User. But not have to reload the entire page, like traditional apps have done back in the past. And still do today, quite honestly. Now this is all made possible, using this AJAX Technology. Asynchronous JavaScript and XML. AJAX is simply an acronym. Really, AJAX is just a bunch of things behind the scenes at work to make exchanging data and updating parts of a page possible. Some of those technologies that could be involved include, XML, Extensible Markup Language. That's a way we can exchange data And mark it up using tags. You can also use, JSON, JavaScript Object Notation. Now, this is what's more commonly used, although we call it AJAX, it really should be called AJAJ or something, because it's really Asynchronous JavaScript and JSON. Because that's what all the browsers understand and JSON like XML is a way to exchange data between a server and a browser very easily and mark up the data. The DOM represents the memory that we interact with, when we're using AJAX. So, when we get back for instance a, an XML or a JSON message we want to take that data and update the page. Well, that means we need to update the Document Object Model. The Document Object Model gets created as a page loads, and you can think of it kind of like a hierarchical chart or a genealogical chart where you'll have nodes that are parents, they have children and they have children and so on and so forth. Finally, another term you might come across is something called REST, Representational State Transfer. Now, REST is simply a way to identify resources on a server and we typically identify them using URLs. And then retrieve data from the server. Back to the client. And that would be typically JSON data in the case of AJAX. So, REST is something that you don't have to use, but it's very popular in the world of AJAX. But another option is Web Services. So, REST is one option for identifying and getting data. Web Services are another option. And both can serve up JSON data or XML data. So what exactly is AJAX? Well, I've already said it's just an acronym. But when you put several technologies together, we get this AJAX functionality. The technologies are all reliant upon one key object that's built into all modern browsers called the XMLHttpRequest object. Now, this object allows us to make what's called asynchronous calls to and from the browser and back to the server and we can go back and forth that way. The object initiates a request to the server and then receives data back but the difference is, we don't have to reload the entire page for this to happen. In fact, we can just load. XML or JSON data. And so the actual payload, the data that's being sent back and forth, is minimized. Now with ASP. NET AJAX we're actually, going to be sending back some data plus some HTML, as I'll show you coming up. But understand that we can use this object to send whatever we'd like. It's not limited to just XML or JSON. We can even send HTML data if we'd like. Now, the other technologies that play a big role are the DOM that I've mentioned. The Document Object Model. Of course JavaScript is needed to trigger the request to the server. And to get the data back in, work with it. And then CSS is even a big player in this. As far as data formats I just mentioned JSON, XTML and HTML, you can even send clear texts if you'd like. Now, these calls are made asynchronously and what that means is. Rather than locking up the browser, when it calls made to the server. What will happen instead is a separate background process will make the call. That way the user can still interact with other controls on the page. And not feel like it's locked up. That's a big deal. Messages are sent back and forth using the XMLHttpRequest. We eliminate page refreshes, and that's our Partial Page Update, term that I mentioned earlier. And we can integrate pretty much with any ACTP Technology out there. So whether, it's Web Services, Web pages, REST API's or some other technology that serves up data, it would be fair game. As long as it's able to handle ACTP requests and responses. As mentioned, this works with all the main strees, stream browsers, so it's really, really easy to get this going and added into your pages and you're going to see in the next section here. That you don't even have to be a JavaScript master to get started with AJAX. So let's take a look now, at what are the different players in ASP. NET AJAX that we can use to minimize the amount of code we have to write.

ASP.NET AJAX Controls

Let's take a look at the different controls available with ASP. NET Web Forms that can easily enable AJAX functionality in an application. So, the heart and soul of AJAX when it comes to ASP. NET Web Forms is the ScriptManager Control. The ScriptManager can actually, do several different things, but one of its primary purposes is to load the necessary scripts that allow AJAX to be used in a particular page. So, you'll always add a ScriptManager control if you want to use some of the other controls that are listed here such as the key one with is the UpdatePanel. Now, we'll talk more about the ScriptManager in the next section. But for now, it's sufficient to say that we have to have it. And then we can use controls like the UpdatePanel, the UpdateProgress Control, the Timer, or even the ScriptManagerProxy. Now, the UpdatePanel itself is used to actually trigger AJAX request. So, the ScriptManager is responsible for managing the different AJAX controls on the page and that includes loading the necessary JavaScript files. But the UpdatePanel is actually, what you're wrap around content. That you'd like to AJAX-enable. This is what enables the Partial-Page PostBack. The Update Progress Control is useful when you'd like to provide some type of a visual as an AJAX call is started. Now, some AJAX calls might, only take a second or two. And in that case, you might not need to show the User, anything. But in other cases, they might take several seconds. Maybe, five seconds, as an example. And rather than, leaving the User to guess as to whether or not the call actually, started 'because keep in mind, we're only updating part of the page, not the whole page. So it's not visually, available. Then the UpdateProgress can be used to show some type of a graphic that lets them know that, hey, we're working here and we'll, we'll get back to you. The Timer control can be used to trigger HS calls on a time basis and then the ScriptManagerProxy is useful in cases where you might have a Script Manager. Inside of something like a master page. But, in a page that uses that master page, you might want to load separate scripts. And so the ScriptManagerProxy, allows you to do that when a Script Managers already been defined. In a master page. So, that's a quick look at the different controls that are available, and what we'll do in the next section is start with the ScriptManager, and then we'll move on to the UpdatePanel and UpdateProgress Controls and show you how those can be used.

Using the ScriptManager Control

The ScriptManager Control is the first type of control you'll need to add into a page if you want to add AJAX functionality and use ASP. NET AJAX Control. Such as, the UpdatePanel, the Timer and the others that we talked about in the last section,. Well, the ScriptManager is responsible for quite a bay. You can kind of think of it as the boss on a job. You have your different workers that perform their different functionality, but, ultimately, you need to coordinate that to make sure that the job gets done. Well, that's what the ScriptManager does. The ScriptManager is responsible for loading the necessary scripts. That are needed to make, AJAX calls and it can also be used to load even custom scripts. So, not only can you use it to loa, Microsoft's behind the scene scripts, that they use with these server controls. But you could also use it to load something like a validation script that you might of written. Now you can also load that of course. Using the standard script tag. But this is another option that's available. It also can be used to call back end services and generate proxy scripts. So, if you had, for instance, a ASP. NET Web Service that you wanted to call that returns JSON data. Then you could use the ScriptManager Control to generate some script, some JavaScript on the client side, that you could use to call the servers and you can do a few other things as well. It supports debug and release versions of scripts. So, when it loads either custom scripts or Microsoft scripts, it can load those as debug release versions. A debug version would typically be one. Where all the white space and tabs, and sa, and formatting have been left in the script. So, it's very easy to read and debug. A release version of a script, typically has all of the formatting stripped out, all the white space that isn't needed is gone and that conserves on the bandwidth and makes a script load faster. So, you can switch between those using the ScriptManager Control. And then finally, as mentioned it's really the boss. So it coordinates all the asynchronous operations. You might have multiple AJAX controls on a page such as the UpdatePanel. And you might have, Multiple Update Panels that make requests back to a server to get data. Well, the ScriptManager, coordinates all those different calls and makes sure everything works properly. Now you can drag a ScriptManager Control, onto your page. And we can either put it into a page, if only the one page needs AJAX functionality, or in situations where an entire site and many pages might need AJAX features, then we can drag a ScriptManager into the Master page. Now, dragging and dropping in's not going to gain, you a whole lot because you can see this is all you need to add at a minimum. So, we need to add ASP ScriptManager id equals whatever you want to call it, and then the standard runat server. So, it's just like a normal ASP. NET server control. By adding that though. It does quite a bit of work behind the scenes for you. To load the necessary AJAX scripts that might be used. Now, it has an interesting property on it. Normally, when you have a non AJAX-enabled page, and a button is clicked, you'll have a, a typical standard PostBack. And you could say in the code behind. And something like the page load, you can say if Page. IsPostBack then, let's do something special such as call a database. Well, sometimes you might want to know the difference between a regular PostBack. Because you could have a form on the page that does regular PostBacks. And you might even mix that with AJAX calls, Partial Page PostBacks, or Partial, Partial Page Updates. Well, you can see here, the ScriptManager, sm was the id, from the last section we just talked about. And you'll see that it has this IsInAsyncPostBack. IsInAsyncPostBack simply checks, are we in a AJAX callback type of scenario and then we can do so, something special if we wanted, based upon the fact that we're doing an AJAX call. And that would be useful in cases where you might have both. Potentially, a regular one and an AJAX one. Now, another thing you can do that's very I found it very useful for debugging purposes is with an AJAX call because the entire screen doesn't load if there's an error on the server side. Normally, with regular PostBacks ASP. NET, would show kind of yellowish red type screen with the error message and the stack trace and other information. Well that doesn't happen when you're doing AJAX because we're not reloading the entire page. So, what'll happen is you might see a little error. But you're not going to see the entire error page. In fact, oftentimes you won't see anything of use. So, what I'll do in those situations is, if something's not working as expected. I'll go in and first make sure that everything works with normal PostBacks and the way to do that is simply set EnablePartialRendering to false. And what that does is now, your Partial-page Post Specs are disabled. And it's as if, you're not using the AJAX controls and the AJAX scripts that get loaded by the ScriptManager. And so, in that particular case, now we can run the page, if we get an error. Net will actually show the real error message. Versus if you're doing AJAX it kind of hides that behind the scenes, and you'd have to use the other tools to view the HTTP traffic. So this is a really nice little trick you can do to easily AJAX enable. Or disable your ScriptManager. And little simple example shown here. When a page initializes. We might go in and have some custom method that we call. SupportsXMLHttp if it doesn't support. The XML HTTP object, maybe it's a really old browser or something, then we can simply say let's disable partial page rendering. We'll set that to false, and now it'll be as if there were no AJAX functionality. Built in the page. So that's a quick look at some of the properties, and the way that the ScriptManager is used. And you can see it's a very, very simple control just to get it started and going. So now let's take a look at some of the things that we can do with it.

Demo: The ScriptManager in Action

Let's take a look at how we can use the ASP. NET AJAX Script Manager Control. So to get started I am going to come in and create a new project and I am just going to create an empty ASP. NET web application and we'll call this AjaxEnabled. Site. Okay, so this'll just give us a web. config file. I'll right click and we'll add a new item. I'll select a web form, and we'll just call this Default. aspx. Now the first thing we'd have to do if we want to AJAX enable this page. Is we need to go in, and add a ScriptManager control. Now if, as mentioned earlier, you had multiple pages you wanted to AJAX enable, then the best place to put the ScriptManager control would actually be the master page. But in the simple example, we'll put it right here. So I'm going to switch to design view. We'll come over to the toolbox. I'm going to shrink up some of our. Standard tools here, and you'll notice we have this AJAX extension section, and here's the controls I mentioned earlier. So I'm just going to drag a ScriptManager on, and you'll see it's a non-visual control. We'll get something in the DesignTime viewer here, but. At run time you don't see anything. It's just a behind the scenes class that as I mentioned acts kind of like the boss. Runs, makes sure all the children on the page that are AJAX controls are doing the thing. So if we switch back to source view. I'm going to change the ID to just sm and you can see it didn't do much at this point. Now if we run it though. There's actually some script that's going to be added. Now we won't see anything in the paste, because we don't have anything yet. But if we right click and view source even at this point. You'll see first off that there's some, different script tags downloaded. Now, some of these relate to other things like handling post back operations but others are loaded by the ScriptManager. You'll also see a little fragment of JavaScript here that references sys. webforms page request manager. Page request manager is actually a client side JavaScript class. And the script manager automatically loads the script that handles and uses the page request manager, defines it. And then it adds this script to actually initialize it for the ScriptManager in the form. And then we can now work with it and use other controls like the UpdatePanel that we'll see coming up in the next sections. So that's what it's done by default. Now, this'll be a pretty boring demo if that's all I did. Did because that's really all you do to get started, but I mentioned in addition to loading its scripts, such as the one that has the page request manager, you can also use it to load your own scripts plus it adds a nice little function. To know about. So what I'm going to do is add in a custom script. So I'm going to make a folder. And we'll just call this Scripts. And then we'll add a JavaScript file into this. And I'll just call this, Custom. js. And let's say that we want to add a function called. Called show message, which takes a message. And then we want to call show message and pass something. All right, now for this demo, I'm just going to do an alert. Pretty basic. So what'll happen is, when this loads, it'll call show message. It'll pass in Hello AJAX and then we'll show that. In the alert. Now I could come up into the Head Section. And we could add a script with the source attribute. But I'm going to demonstrate how we can use the ScriptManager, so let me go back to here and show you some of the properties. All right, so a couple things to run through. You'll notice that up top, let me switch to alphabetical here. Yeah, we have a couple things related to errors. If there's an asynch post-back error what do we want that message to be? Post-back timeout. When it's doing the AJAX request. How long should it last? And then, moving on down. You'll notice that we can do Enable Partial. Rendering that I mentioned earlier. Some other, quite a few other things we can do as well. But the one I want to point out is the scripts. So scripts allow us to reference a custom script. And we can come in, I can add a new script reference, it's called. And then give it the path to our scripts. So, I'm going to say start at the root which is ~/ go to scripts, and then go to custom. js. Now, what's that just did is it's going to use the script manager to actually load our custom scripts. And the benefit. Is that it will load these in the proper order that you want, plus it will handle a particularly painful case which is knowing when has the page and the different controls in the page actually loaded and when are they available. Okay, so if we run this as is we shouldn't have any problems because we're just popping up a message box. So it should show us and there we go. We get Hello AJAX. Nothing big. But let's change that up. Let's come down and we'll add, say, a span tag with an Output Span id. And now, let's change this to go find that and we'll show Hello AJAX in that span. So we'll come in and first find the span. We'll use the standard document. getElementById, and then we'll say span. innerHTML = the message. Okay, now the challenge is when you're trying to reference a control that might not have been loaded yet. And are using custom scripts. You know it's very important of course that this span, output span, is all ready loaded with our page. Well, the script manager's not going to help you out much there because it's going to load, you'll see it's on top of this. Now we could move things around and make sure that this script is loaded last but, I'll show you a nice little addition, that just by adding the ScriptManager you can add this very easily and be cross-browser. 'Kay and that is if I ShowMessage right now you'll see we should get an error. So let me run this. And you'll see it says we're unable to set tha value of the property innerHTML, because the span object is null or undefined. And that's accurate because the script's running, before the span has actually been loaded. Well, the script manager loads some scripts I mention. And it has a nice little handy function that works cross-browser called page load. Kind of like the page load with code behind files. Now, this is available and called once everything in the page is ready to go. So we can simply say when the page is loaded, let's go ahead and show this Hello AJAX message, which now the span should be ready to go. So we'll come in here, we'll View in Browser, and you'll see now it works. So that's a nice thing that the script manager adds for. For us that if you're using custom scripts, and want to be cross browser and not worry about different unload events that fire across different browsers, this one works well in different browsers and it's part of, it's part of the ASP. net AJAX library. Now on the server side. If we were making a async post back, or what we call a partial page post back, then we can do as I showed earlier. We could say if our script manager is in async post back, then we might want to. Load some data. So let's pretend that we had this load data method, and that might update, let's say we had a data grid or something, that was triggered using asynch post-back. So now we could call Load Data. Now this would only get called if. If we're in an async post-back. And that's very different than the following. This is the standard ASP. NET is postback. That would be called if we're not in an AJAX type call. Versus the script managers is in async post back if a course gets called, if we are in an HS call. So, I'm not going to actually run that one in this case, but I do want to point out that that is available on the server side, if you need to know when an HS call is being made. Alright, so that is a quick look at how easy it is to get started with the script manager and And one of the things it can do, which is adding your own custom script references, as well as, how we can use the page load that the ScriptManager makes available to us.

Using the UpdatePanel Control

The next control that we'll talk about is the UpdatePanel. And I happen to personally think this is one of the more fun controls to play around with in ASP. NET. Because with just a minimal amount of code on your part, you can enable AJAX calls and you don't even have to know any javascript. So the UpdatePanel Control does require. Require the ScriptManager. If you recall from the previous section the ScriptManager loads required AJAX scripts, and it initializes a special client side JavaScript object called the page request manage. Well that page request manager. Is actually used by the UpdatePanel Control. So as long as you've added a ScriptManager into your page, or inder a ma, into a master page, then you can use the UpdatePanel. What does the UpdatePanel do? Well, it allows you to wrap portions of a page that need to be AJAX enabled. So that if a user clicks, for instance, a button. Button in that section of a page. The contents of that session can be updated asynchronously or through what we call a partial page post back, or you'll hear partial page update. This allows a really easy way to make it so the whole page doesn't have to reload. Now it can do some other things as well. You can actually cause buttons outside of the UpdatePanel, to be clicked and then cause a certain part of the page to update. And that's called an async post back trigger, which we'll see in a moment. You can nest UpdatePanels. So if you have a master parent update panel and want to nest a child inside of it. In more complex scenarios you can do that. So that only the children would be updated for instance. And then it provides a content template that you'll be using called content template. And the content template simply is the wrapper for the content you would like to AJAX-enable in a particular page. So here's kind of how it works. You have Default. aspx, and let's say that you have a simple form in there. And you would like the user, when they click that form, to not have to reload the whole page. You just want to show the results of the form, and let them keep going on that page. Well to do that, you can drag an UpdatePanel control onto the page, and that UpdatePanel control would have that content template. The content template, then, would have the contents. Of the form and in most cases even the button that would be clicked. When the button is clicked the XMLHttpRequest object you see here, would trigger an async post back to the server. The server then will process the data, and then it will send that data back. And the update panel will then update this portion of the page. Now, the rest of it though, if there was a header, a footer, maybe a menu over here, will be left alone, nothing happens. How does it work? Well, behind the scenes the UpdatePanel interacts with the page request manager. That I showed you earlier. And it intercepts in essence the normal post back operations that would occur, for instance, when a button is clicked. And converts those into asynchronous post backs or partial page post backs. And as I mention it uses the XMLHttpRequest Object that's available in every major browser out there nowadays. Now it does send information in the controls. So if you've filled out text boxes or something else that data will be posted back to the server with the XMLHttpRequest. And it also sends something called ViewState along for the ride as well. ViewState is generated as you use ASP. NET web server controls. Now in many cases that might not be a big deal but if you are not finding that your update panels aren't responding quite as quickly as you would think they should. You might want to go make sure that your ViewState is not too big. See how big that is and how much is being sent. Now when the response message comes back from the server, the update panel will automatically take the html that was sent back and update the appropriate part of the page. Absolutely no JavaScript is required out of the box. Now as you move past the update panel, you certainly can write JavaScript to integrate. More, sophisticated ways with the, page request manager, but it's not required out of the box as you will see as I do the demonstrations. So here's an example of what the UpdatePanel control looks like, and I'll run a demonstration, similar, similar to this coming up. In this case we have the UpdatePanel. We have the standard ID, and runat server attributes. And then here's out content start tag and end tag, ContentTemplate property. Now inside of here we have a Grid View, and we have a Label. And in this particular case we'd like any action that occurs within this ContentTemplate to be asynchronously. Updated, or in other words, an AJAX call. Now what'll happen then is the UpdatePanel will intercept, sorting on the columns, or clicking on a paging icon or link, and automatically send that back to the server in the background, and then update the HTML that comes. Back, so it'll be not only faster, but of course, the whole page doesn't have to reload. Now there's some other things you can do as well. In cases where you might have some controls that are outside of the UpdatePanels. Sometimes you might have a search form, or something that's not something you can wrap in the UpdatePanel, itself, within the content template. Then, you can actually tie the external control into the UpdatePanel. HS call, operation. And the way we do that is we add a triggers tag, and you'll notice I have this async post back trigger, and I give it a control ID. What we're saying is I would like button submit to participate in updating this update panel. So you'll notice down here, outside of the UpdatePanel content template, I have button submit, standard run at server. And then we say up here on the aysnc post back trigger, when the submit is clicked so you have to define the event name, then we would like the in this case the grid view to be updated with fresh data from the server, so although this control is outside of the UpdatePanel we can still interact with it and cause it to perform. An async post back for us. So let's take a look at how we can use the UpdatePanel, and it's content template.

Demo: The UpdatePanel in Action

Using the UpdatePanel's quite simple. We can start with a page that we'd like to AJAX enable, add in the functionality and then we're ready to. To go. So I have a page that we did earlier in the ScriptManager Demo that I am going to start with. And it had a reference to a custom JavaScript file. So I am actually going to just comment this particular code out right here, because well we really don't need it. So I am going to say Comment Selection. Now from there. What I'm going to do is, let's say that we would like to add a label. We'll keep it real simple to start. And we'll say this is our time label. And then under that, we'd also like to add a button. So, we'll do an ASP button,. And this will simply get the time. So when this button is clicked, let me go ahead and switch over to the design view. We want to go in and update our label, so we'll simply say time label. text equals and we'll just do date time. now. Too short time string. Okay, so very, very simple, pretty straight forward C-sharp code there. So, let's go ahead and build that. It looks good. Let's right click and view in browser, and we'll hit get time. And you'll notice it does work,. But, if you look up, you'll see the little e up here kind of flashes, and it's still fast because I'm, of course, hitting my local server. But, it's definitely not an AJAX call. So, to make this into an AJAX enabled page, it's very, very simple. We can come into the toolbox, I'm going to go ahead and pin that, we'll come down into our AJAX controls and because I already have the ScriptManager from the previous demo, I don't need to re-drag one of those, in fact, you only want one of those. But we can drag on our UpdatePanel, so I'm just going to drag that right in there, and then I'm going to grab these controls, and we're going to move those into the UpdatePanel, so I'm actually just going to grab em right here. And if you recall there's a content template. All right. So there we go. That's all we have to do to AJAX enable that part of the page so when that button is clicked, the label will get updated as appropriate. So let me view in browser now. And when you do get time you can kind of watch the e up here. And you'll see there's absolutely no flicker at all this time. It's so fast you really can't tell that anything's going on as far as hitting the server. Now it did actually hit the server though. How do we know that? Well one of my favorite tools to use is called. Fiddler. And you can go to Fiddler2. com to grab this. And if you just click on Get Fiddler you can install it. It's a very small download. So we're going to go ahead and run it directly. And go ahead and install it. Very fast to install as well. And we're ready to go. Now I normally keep Fiddler right down here on the bottom. And, so, I'm going to go ahead and that. We'll right click and say pin to task bar. Now what Fiddler is is an HTTP proxy tool. It intercepts messages that go from the browser to the server. And so. If I go ahead and start it, you'll see at, at first glance might look a little bit complex, but it's not bad at all once you get the hang of it, and it's very, very useful in AJAX scenarios, so right off the bat you can see Fiddler went out if. If we come over to our inspectors, and it made a request up to the Fiddler site to check the version, and you can see it returned some version information right here. I'm going to go ahead and clear that out by clicking the X and saying, Remove all. Now let's go ahead and rerun the page, and let's see what happens here. So I'm going to view in browser. We'll get time. Let's go look at fiddler. You'll notice that Default. aspx is first shown. But then it's shown again. And the reason for that, this is the first load. And this is everything that was loaded into the browser. So here's the ScriptManager scripts, here's the page request manager that the ScriptManager initializes, here's our UpdatePanel area with our span and our button. But let's go look at the second one. This should should be the AJAX call and here it is. So you'll notice that the data set up. If we pull this up in notepad and go a little bit bigger on the font. This is actually the data was sent up. So, it knows the. UpdatePanel that was being updated. Here's information about the button. Here's the view state. That's standard with ASP. NET controls. And some other information from the form. And now, you'll notice that you can tell this is an a synch post. A synch post is true. So, we could actually come in and, on the server side, we could have written some code to only update this if it's async post back but because the button is wrapped in the UpdatePanel, it always will be, so there's really no need there. But you'll notice that the data that came back. Includes the HTML that should be rendered in the update panel. Notice that we actually have the full on span, with the time label ID. This is our label control, as it's rendered to the browser. And then we have our button, and then we have some other data, such as the view state, and some other things we won't go into. Here. But these are actually really important because basically what happens is this all gets parsed word, and this information right there, is updated in the page, and that's what we see when we run it. And so if we, oh, we go back to here, in browser. That AJAX call you can see is not sending the entire HTML back from the server, only parts of it. Now, another reason this might be useful, the Fiddler tool, and using the update panel in conjunction with it is, if we go back, and let's just for. To demonstrate this, let's throw a new application exception. And we'll do my favorite, an error has occurred. Not the most fun error to get. And let's go head and, build that. All right, now it should load fine. But now let's hit this, and you'll notice that we do get an error. Just says an error's occurred, but we can't get to the stack tray, and we really have no idea. Other than the error message of what occurred. No there's some things you can set on the ScriptManager. Such as the async post back error to show. but, let's go look at what Fiddler has to offer here. So here is the actual initial load of the page. And here is the data that was actually sent. Back. And you'll see that it's not much better than what we had. But at least we can see the raw data that was sent. Let me clear all that. So what do we do then? You know, how do we know exactly what line number? All that information. Well, one of the tricks I mentioned earlier in the initial, discussion of the ScriptManager was we can come in and we can do it through typing, or it's actually just easier to go to the properties, and we can set them. Right there. Enable Partial Rendering to False. Now what that does is turns off the AJAX functionality. So we have a problem. We're not sure why. I really don't want to strip out the UpdatePanel here that I already have. You know I took a little bit of time to put that together, so let's run it now. We'll hit Get Time and now notice we get what we would be used to getting in ASP. NET. So that's a really nice technique you can use if fiddler doesn't have the information you need. Then we can jump into this. And take a look at the actual error simply by setting partial rendering to false on the ScriptManager. So let's go ahead and set that back to true, and we'll go ahead and we'll take out our error here. And now everything should be back to normal so. Let's run it one more time. Or each has calls made and now we're working so it's a really nice trick that if your working with the UpdatePanel, you're not exactly sure what's going on but you don't want to rip out the UpdatePanel, and the ScriptManager and all of that from your HTML source then that's what you can do to go in and enable it. Now, let's say that the button was actually stored outside of this. Let's say for whatever reason, maybe it was up top in another part of the page. Maybe it's up in the header, and for whatever reason we can't actually. Wrap the label and the button because they are in separate parts of the page with one UpdatePanel. Well if we do it as is. If we run the button, and run the page with that. You see that everything still works but you'll notice if I do it again, this will, that E here will. Will kind flicker. You'll see it's flickering there. And that's because we're not doing an async post back at this point. Now the reason for that of course is the button is outside of our UpdatePanel. So, to remedy that, let's go ahead. And we could either, A, move it inside, of course. But we're going to say that's not possible. So we're going to leverage the UpdatePanel's trigger functionality. So I'm going to come over to our UpdatePanel. Go ahead and highlight that, and go to the properties. And you'll notice we have this triggers collection, so I'm going to click on that. And there's two types of triggers you can add. You can do a regular PostBackTrigger which just does a standard Post back. Not something I really want in this case. It's already doing that, or an AsyncPostbackTrigger. So I can select that. We can go in and say, what's the control ID? Well now, we can come in and actually pick the button, and then what's the event name? On that button that triggers the async post back. Well, we want the click event. So we'll hit okay. So just by hooking up the triggers that you see here, we can now go in and easily tie in this button to this UpdatePanel, so that if the button gets clicked, we can automatically update the label. But we'll do it asynchronously using an AJAX call. Let's go ahead and run it now. kind of watch up top here. We'll do Get Time and notice no flicker at all. And if we go to Fiddler, let's go to the very last one, you'll see that it, it is indeed an update panel request and a response coming back. There's no, you know, this is the entire page. This is what we'd see if it was reloading the whole page. But instead we're seeing very small fragments of data sent back and forth. So that's an example of using the UpdatePanel and really just getting started, started with it. So in the next demonstration. I'm going to show you more uses of it, and we're going to tie it in to the time card demo, that was created in earlier modules.

Demo: AJAX-Enabling an Existing Application

In previous modules, I built a simple time-sheet viewer application, that allows employees to be selected, and then their time-cards to be viewed. And so, if we run this, we have a very simple master page. Has a header and a footer, and then we have the content in the middle, and we also have a printable master page where it strips off everything. So as we select different items, it is reloading the entire page. Can't really see that here because I'm running it locally. It's very fast. But if we watch up on the E again. You'll notice that it does reload the entire page. So what I'd like to do is AJAX enable this application. Well, the first thing I need to decide is where do I put the ScriptManager control, because we do have to have the ScriptManager. And so if it was only used in this page, I'd say put it just in that page. But I'm going to assume. That we might have other pages we'll add that need the ScriptManager control. So we're going to go ahead and run off to default master, and you'll see that I have a ScriptManager placed in there. And I didn't even do anything fancy at all. Literally added an ID, and the runat server on it. And we're ready to go. So that's already been drag and dropped on. And that's all we have to do to get going. Now from here we can come into the Timesheet viewer. And we can wrap an UpdatePanel around the content we would like to handle. Which would be these two grid view controls. And so the easiest way to get that going is we'll just drag an UpdatePanel right here. And in that UpdatePanel we can go ahead and put these guys. So I'm going to run back to the source and just copy those in. Keep it really simple, make sure we grab what we want. And we'll come down to the grid view, and you don't even have to put the object data source controls that I have, inside of here. So let me grab this one, and we'll put it outside just to demonstrate that. Okay. So now what we have is the two. Grid views, that drive showing employees and showing time sheets available inside. And let's go ahead and run it as is. Okay, and you can see everything's displayed. Now, let's watch up here on the E, we'll hit Select. And you'll notice that nothing is being reloaded up here, we're now doing AJAX. Now that seems great and at first glance that's about as simple as we can get. Let's go ahead and pop open Fiddler here. We'll clear out the existing calls that have been made. And let me just refresh the page, and then we'll select an item. All right, so if we go back you'll notice some scripts were loaded. But here's Time Sheet Viewer being loaded initially. And you can see there is quite a bit of data here that was loaded for, the first grid and things. And then here's the second call that was made. And so, right off the bat you'll notice that here's the data that was sent up, and there's a fair amount of view state in here, it goes all the way to about there, and that's because of the grid views. And then here's all the data that was returned, including the updated, ViewState. But you'll notice that we have two tables. Here's our bottom grid which is the time cards, and here's our top grid, which is the actual employees. Now while that works, the employees aren't changing. There is really no reason to put those inside of the UpdatePanel. Because really, all we want to update is this bottom table. So, one of the nice things we can do, is we can come back into our UpdatePanel. And what I'll do is grab this grid, well, actually the one right above it, that represents the employees. And we're going to put that outside because we don't want to load that except for when the page first loads. So we'll put that outside and I'll also move the BR tag and put it right below it. Okay, now the challenge though is if we run it. Let's watch again the loading up here that occurs. You'll notice it is posting back the entire page and if we go back to Fiddler, we can verify that. Here's the page and everything's being loaded. So not very efficient at this point in our application lifecycle because we might as well be doing a full page postback. So to fix this, all we have to do is leverage the trigger functionality that I demonstrated earlier. So, we can come in to our UpdatePanel, should be right down in here. And here's the GridView inside of it. Now, you'll notice I'm not able to get to it real easy. So I'm going to run off to the properties, and we're going to come into here and see if we can find it. So I'm just going to click, and there's my UpdatePanel right there, UpdatePanel1. So if you can't see it visually, you can always go to the Properties window and select it. Okay, so now what I'd like to do is go to the Triggers Collection. I want to add an AsynchPostbackTrigger to trigger an AJAX call. And the control I'd like to target is the first grid, which is Gridview1. Now the event is going to be when the selection changes. And you'll see we have a SelectedIndexChanged right here. And when that selected index changes we want to go ahead and trigger the AJAX call. And we'll hit OK. So now all that did in our source is added this trigger very similar to what I demoed earlier in this module, and we have an AsyncPostBackTrigger, it targets the grid, and the SelectedIndexChanged the event. Now we've tied in, this grid that's outside of the UpdatePanel, and we've tied that into the UpdatePanel itself, so now it will trigger an AsyncPostBack. Let's go ahead and Save and run it. Now, watch up here on the e again. We'll Select, notice, absolutely no flicker at all. And that's because, now this is only loaded once. Let's go prove that by looking inside of Fiddler, and notice that here is the actual HTML that was updated. It only has the time sheet data, you can see some of that data here for the hours. It didn't update the external grid or the employee grid in this case, so this is much more efficient in situations where you have a control that really only needs to load once, so you don't want to put it inside of an UpdatePanel. Because if you do, it's going to reload multiple times. So that's an example of how easy it is to use the UpdatePanel, but also integrate triggers to AJAX enable our time sheet viewer.

Using the UpdateProgress Control

The final control that we'll talk about in this module is the UpdateProgress Control. This control, like the UpdatePanel, is very, very simple to use and very effective for notifying your users as an AJAX call is starting and as it completes. What will happen is without an UpdateProgress Control, if you have an AJAX call that goes and the server is called and let's say it takes a few seconds. Some users get a little bit antsy and they start clicking around, I call them click happy users. And they'll start clicking which triggers another AJAX call, and they click again which triggers another, and so on and so forth. So the UpdateProgress Control, all it does is shows and hides text, images, even Silverlight, or Flash plugins if you'd like that give the user a visual indication that, hey something's go on, so chill out for a little bit. And so the UpdateProgress Control is very, very simple to use. It has a ProgressTemplate, you could see that here. And whatever you put in the ProgressTemplate will automatically be displayed when the UpdatePanel is performing the async call. Now what's nice about it is you can go in and control when it's displayed. So, you can go in and actually say hey, only display this after maybe 100 or 200 or 500 milliseconds have passed. Because if it's a really fast call, you don't want a little, animated GIF or a PNG image showing just really quick, kind of flashing to the screen and then going away. So you can control that as well. So, this is something you can drag on from the toolbox. You can either have it take up space in the page, DynamicLayout true would mean that it dynamically injects this as it needs to display. Or you can have it false, where if you put in an image that's maybe 50 pixels tall, it'll take up those 50 pixels, even if it's not showing. Now, if you put this inside of an UpdatePanel, it will automatically trigger and show whatever's in the ProgressTemplate here, in this case this progress. gif, when the UpdatePanel itself triggers an AJAX call. Now in some cases though, you might want to put this progress. gif in a different part of your page. Maybe you don't want it right where the UpdatePanel is. So what you can do is associate it with a specific UpdatePanel. So in this case, we're going to say the associated UpdatePanel ID that ultimately causes this to be shown, would be UpdatePanelCusts in this case. Now that's not needed, if you just put the UpdateProgress directly in the UpdatePanel, then it will be automatically be triggered by default. But this is very useful when you want more control over where the progress indicator actually shows to the end user. Now an example of using this is shown here. I put together a little sample app that calls out to Amazon. com and gets some album information. And when that album information comes back it'll display it. Well, I don't have any control over the Amazon. com service, so I never know, you know, what's in between me and the network, because they might serve the data fast but the network itself might be a little bit slow that day. So I added an UpdateProgress Control, as you can see right here. And this is just an animated gif, it doesn't actually give any real indication of when the call's going to come back. It's just a way to let the user know that something's going on. But this'll let them know when the page is triggering an async postback operation in an AJAX call. And then when the call and the data comes back, it'll automatically hide that particular indicator. So it's a very simple control to use and something I highly recommend using if you'll be using the UpdatePanel. So that your users know, as an operation's actually occurring. So, let's take a look at using the UpdateProgress Control.

Demo: UpdateProgress in Action

Anytime you use an UpdatePanel Control, you might also want to consider using an UpdateProgress Control. Now, there's going to be situations where the data returns from an AJAX call so fast, that you really don't need to notify the user that an operation's currently in progress. But there might be other situations where it may take a few seconds and you need to let the user know. Well fortunately that's very, very simple to do. We can come in and either in the UpdatePanel or outside of the UpdatePanel, we can add in inside of the content template if we'd like, an UpdateProgress Control. So I am going to go ahead and do that right here. So we'll say asp:UpdateProgress. You can give it an ID, we'll just call it progress run at server. And then you can also do things like, do you want to display it after a certain number of milliseconds, maybe after, say, 300. If the data returns within that 0 to 300 range, we probably don't want it to show up because it would be so fast, it would just flicker on the screen and not really look that good, so we'll go ahead and say after 300. And then we'll come in and we can say, DynamicLayout, and I'm going to say True. And what this means is don't take up space for whatever I put in the Progress Indicator template, just make space dynamically. And you can see there's a few other controls, if we were going to put this outside of the UpdatePanel, which we could do to position it however we want. Then we can give the ID of the UpdatePanel whenever that UpdatePanel triggers an AJAX request, it would automatically cause the UpdateProgress Control to show its template. But we'll go ahead and go with this particular, set of properties. Now inside of here we just have a single template called ProgressTemplate and I'm for now just going to say Data Loading, please wait. All right now, if we run this because it's local, it's going to be so fast that we probably won't see a whole lot. So let's see what we get. And you'll notice it returned faster than the 300 milliseconds, so it didn't show anything at all. So, let's go ahead and, when that's clicked, we're going to be loading some data behind the scenes. And, so what I'm going to do, if we go to the code behind there won't be much there, because we're using an object data source. But I'm going to simulate that this operation maybe took a little bit of time to perform. So I'm going to come into our, again employee time cards here, and I'm going to do something one these don't try this at home things but, it'll be good for demonstration. And that'll slow it down on purpose. Now again, normally you don't want to add that. But for demonstration purposes it works pretty well. All right, so we'll let it just in time compile. There we go, we'll hit Select. You'll notice data loading, please wait. And then it instantly went away. Click again, data loading, please wait and then it reloaded the fresh time card. So very simple because I didn't have to do any JavaScript, there's really nothing at all I had to do. Now what's even nicer than maybe adding text is we could add a visual image, but what do you add? Well, typically an animated GIF would work, or you could even work with PNGs in some cases. But, there's a really nice site out there called www. ajaxload. info, and it just has all types of different graphics that you can use and put them in your applications. So we could do, for instance, an indicator, if I hit Generate It. There's our indicator. Now, again it doesn't actually monitor how long it's really taking, this is just a visual indication of what's going on. We could put it on a transparent background and make it so it works on different things, so we could do bouncing balls. And there's just a lot of different settings you can do here. So here's kind of a clock. That looks pretty nice so now I can right-click, and we can Save Picture As, and then I'm going to navigate to where my project is. And we'll come in to my Time Tracker. Let's make an images folder in here. And I'm just going to call it, progress. gif. So now what I'll do is in Visual Studio, I can hit Refresh but you'll notice because this is a web application project I can't see my images folder. So if I hit this Show All Files, there it is, we can right-click and we can say Include In Project and there we go, there's my progress. gif. Alright, so now, I can simply come in, we can drag that into the ProgressTemplate and now we'll run it again. All right, we'll click on Select. You'll notice now it's showing our image. And then it goes away and you'll notice it's dynamically adding space for it, and then taking it out. Now, if I wanted that space to be kind of hard coded, the size of the image, we could set DynamicLayout to false, and now it'll actually embed some space for the size of that image, and it's going to leave it there you'll notice. See that gap here, so in most cases I kind of like the DynamicLayout true. But there are a few cases where it might add a little bit of a weird expanding, nature to your page. So it's something you just have to play around with, and find what you like. So that's how easy it is to add an UpdateProgress Control, define a ProgressTemplate and then set a few properties such as Display After and DynamicLayout into an UpdatePanel. And as mentioned if I wanted, I could even move this outside of the UpdatePanel and then we could use the associated UpdatePanel ID to reference this UpdatePanel and the same thing would occur. It would just allow more flexibility on where that data, or where the Progress Indicator actually goes. So, very nice control, something I'd highly recommend you add because that way your users at least know that something's going on in the application.

Summary

AJAX is one of the most popular web technologies out there and for good reason. People like fast and efficient user interfaces. In this module, I've walked you through the fundamentals of what AJAX is and discussed some of the technologies that actually compose and make AJAX possible. So, we've talked a little bit how JavaScript, CSS, the Document Object Model. And most important, the XMLACTP Request Object can be used to perform partial page updates. And this allows of course just a part of the page to be updated rather than the whole page. Throughout the module, I've explained some of the different ASP. NET web forms controls that you can use to AJAX enable parts of your page without having to be a JavaScript expert. And we went through controls like the ScriptManager, which is responsible for managing different AJAX controls on a page, such as multiple UpdatePanels. And it's also responsible for managing the calls that get made to the actual server in an async postback scenario. It's required, of course, so you do have to put a ScriptManager even though it's a nonvisual control. But once you have it, we can then use controls like the UpdatePanel. We talked about how it exposes a ContentTemplate Property. You can then wrap your different content you'd like to AJAX enable with that, and it'll automatically handle all the heavy lifting of triggering the AJAX calls back to the server and then updating the HTML that comes back. And then finally, we talked about the UpdateProgress Control. Which provides a very simple way to show users what's going on. So they know an operation is occurring and don't try to click and get all click happy on us and trigger multiple AJAX calls. So that's a wrap up of AJAX in ASP. NET. Now keep in mind this is really only scratching the surface of what you can do with AJAX. Some other technologies that you can go off and research would include, exposing data through web services. Using other libraries like jQuery to make AJAX calls. And there's even other technologies such as REST API's that could be discussed. But now you have a good foundation on how to get started. And I'll let you take it from there. So, that'll be the end of this module on AJAX technologies in ASP. NET Web Forms.

Security

Introduction

Hi, this is Dan Wahlin with Pluralsight. In this module, we're going to talk about ASP. NET security and the process of securing folders or individual pages within an ASP. NET website. So we'll start off by talking about some of the key terms and technologies that you need to know about. And we'll talk about two key modes that ASP. NET web form supports out of the box for securing internet or intranet based applications. From there, we'll talk about a special website that Microsoft provides called the ASP. NET configuration website. This is a website you can access directly through Visual Studio or Visual Web Developer Express, and it allows you to configure security for an ASP. NET website without having to write any XML code in web. config. And it'll even create a database to store user credentials. Once a database is available you can even create users. You can create roles. And even define access rules for who or what role is allowed to get to a particular folder or page. A really easy tool to use. And I'll walk you through the process of getting started with it. From there I'll introduce a tool called aspnet\_reqsql. And if you already have a sql server, database, then you can use that for security, but Microsoft provides one that you can use out of the box, and this database automatically can store users, roles, passwords, all that stuff you would need for security. And this aspnet\_reqsql tool. Can be used to install this custom Microsoft database, onto your own SQL server instance. So I'll walk you through that process as well. And then finally, we'll talk about one of the more fun aspects, I think, of security with ASP. NET is, and that is the security server controls. These controls will. Very easily allow you to do things like log in the user, recover a users password, register a user, and much more. And so, I'll walk you through the process of using these server controls. And show you just how easy it is to get security into an application without having to write hardly any code on your part. So let's go ahead and jump right in to an introduction to security in ASP. NET web applications.

Introduction to ASP.NET Security

In this section we'll talk about a few key security terms, as well as two modes ASP. NET. Supports for security, that can be used to secure internet and intranet applications. Lets start off by talking about a couple of terms that you might run across. So, the first is authentication. You've logged into a website, and you typically supply user name and password. And you're proving who you are. So, that would be authentication. Authorization. Well, just because you get in to a website and log in successfully doesn't mean you'll see all aspects of that web application. There may be an administrator side, an editor area, or approval area that you have to be a member of a specific group or role. As a user in order to be able to see those so that would be authorization, what rights do you have and what actions are you allowed to perform as a user. Then the final listed here is impersonation this will be where you'll log in as you, but then behind the scenes ASP. NET might actually run as a different account. Maybe because that specific account has rights to hit a database, versus your account that doesn't. So, sometimes you might impersonate and run under a different account, once a user's logged in, so you can get to some back end processes. You can even use the, use this to elevate privileges and perform very specific custom actions that the default user can't do on their own. So there's a lot of things you can do with that, and then typically once your done impersonating, you'll do something called unimpersonate and resort back to the standard user account. Now ASP. NET supports two types of security modes that can be used for authentication authorization or even impersonation, so the first is forms authentication the second is Windows Authentication. Now forms authentication is more of an internet based type of authentication. Versus Windows Authentication is typically used for intranet applications. Now it doesn't have to be though. You could certainly use forms authentication internally within a company if you'd like. So let's talk about each of those, and explain the differences in how they work. So Forms Authentication, it's typically used with a custom database that stores users, and maybe even roles. So this would be anything with custom tables that you want to query, so that when a user goes to a web page to log in, you'll write a query against the database, and then get back results saying whether or not the username and password is found. That's the standard way. But it doesn't have to be a database. It could be a custom XML file, or some other type of structure that you want to store users in. So the way it works is this is more internet based authentication, and you can even do authorization with this as well. So what'll happen is if a user goes to a page they don't have rights to, they'll automatically be redirected to a login screen. Now that'll happen as a result from some configuration code you can add. And a web. config your websites config file. Now once they've logged in if it's successful, they'll be issued something called an authentication token, now this is basically a nice little cryptographically secure token that'll be stored in a cookie, or it could be stored up in the url in case they have cookies turned off. And what this does is now every time they hit another page, that cookie or the information up in the url, will be passed to that page. ASP. NET can then verify the authentication token, make sure it's valid, make sure it hasn't been tampered with. And then let the user in or block them. It's a very secure technique that's actually been in place for nearly 10 years now. Very, very secure and works very well. Now, behind the scenes it actually relies on a class in the system. web. secure. Security named space called forms authentication, and it has a method on it called set cookie, and you can call that if you actually want to see this authentication token. And that'd be used in cases where you want to write some custom code to authenticate a user, maybe against your own custom database as an example. And as mentioned, all of this is configured in web. config. Now, I mentioned that FormsAuthentication is typically used in internet scenarios, but it's very possible you might mix and match pieces from both worlds, Windows and FormsAuthentication. And for instance, you might log in a user through Windows Authentication, nut then add roles to that user using forms techniques. So, we'll talk about Windows Authentication here next. So Windows Authentication is actually the easiest to get in place. There's nothing you have to do in web. config out of the box, because it's the default authentication mechanism. And Windows Authentication's very easy and, and the reason is that the operating system itself, on the server, once it's passed the user credentials, it will take care of validation. So, you have to write no programming code out of the box to authenticate a user. So, when the user hits a web page one of two things generally happens, if it's Windows Authentication. If the browser is set up to use pass through authentication, then there are Windows credentials they logged into Windows with, or any other operating system they have, will automatically be passed through assuming this supports one of the Windows authentication technologies. The different technologies are listed here. You can see Basic and NTLM, Digest and Kerberos. Basic is what it says, it's the most basic and what happens is the user name and the password are passed in clear text over the network. So often times when basic authentication is used, you'll see https up on the url, because it uses SSL behind the scenes to encrypt the traffic over the network. Now what'll happen is administrator typically will simply go and right-click on a folder or file. They'll go to properties and then go to the security tab to add in the users or the groups or roles that have access to that folder or page. And so what'll happen is the client makes request if pass throughs on, their credentials are passed through, and if their valid IIS, internet information services, the web server will forward that request onto ASP. NET. If they're not valid, ASP. NET doesn't actually have to see it. Your code will never get hit itself. If pass through is not enabled then they'll be prompted with that grey dialog that you've likely seen if you work internally in a company, and they'll have to enter user name and password or possibly domain, user name and password. And then the same process applies. IAS then takes that, authenticates again something like active directory or another L app. Directory services, store. And if it's valid your code would then be invoked. And then you can get to the user name. You can get to the user's roles or groups they're in. And all that type of stuff. So that is what Windows authentication is. And that's why it's typically used for intranet scenarios. Because we don't really want to add people outside of the company into our active directory. Although in some cases you might for extranet type scenarios. But that's the difference between Windows authentication, which is more intranet. Versus forms authentication which is more web based. Uses standard web technologies to secure.

Demo: ASP.NET Security Modes

Let's take a look at the difference between forms authentication versus Windows Authentication from the end user perspective. So the first thing I'm going to do is create a new project. And we'll create an ASP. NET web application. And we'll simply call this Forms Authentication Demo. And I'll go ahead and save this to a security folder. All right. So out of the box, you'll notice that if I click on View Browser, we have a log in area, and we can go in and register and do all that type of thing, and that's the standard way with forms authentication that you would actually go to log in. So we'd have to go in and register. This'll take a second to generate because behind the scenes, this will actually create a database for us. Not too bad. And you'll notice that we're now logged in and now we can get to any site that's been secured. So that would be what forms authentication looks like. If we log out, then of course, we'll have to go back to our login page to get to any secured sites. So that's pretty standard. What most people are used to out on the internet. Now we can compare that with Windows Authentication. So on this one I'm just going to create an empty site. And we'll say, Windows Authentication demo. And you notice in web config, first off there's nothing specific to the security mode. Now by default though it's actually using Windows Authentication. And I'll show you how we can change that later in this module. So to start, I'm going to to add a page. Let's call it default. aspx. And we'll say, you made it. Now if I run it as is, we haven't turned on any authentication at the server level. So you'll notice I can get right to the page. But, if I right click on my project, come down to Properties, and come over to Web. This'll allow me to get into the port settings. Plus, I can say, I'd like to use Windows NTLM Authentication. Now, if I save this, and we'll try to go back to the page. You'll notice that I'm now prompted to inner credentials. If I hit Cancel, I can't even get into the page at all so my code is never touched at this point. Let's go ahead and refresh, I'll go ahead and login. And now you notice I made it end of the page. So this will be useful in situation where you have an actor directory set up, and you're users throughout your company already stored in that, because now I can go in and I could actually secure this page if I wanted. So, we could come in to the application itself. We'll pretend that this is where it lives on a real server. We can come into default aspx. We can right click and go to Properties, go to Security, and now I can actually come in to here and we can edit which users or groups are allowed to get in. Now you'll notice I'm already allowed to at this point. But if we hit Add we can certainly add other users, or other groups in our active directory. And what's nice about this is an administrator if you have that in your company can automatically handle all this for you as long as they know about the appropriate security and the groups or users that should get to the page. So that would be the difference between using forms authentication, which you can see is definitely more internet looking type authentication, versus Windows authentication in an ASP. NET application.

Using the ASP.NET Configuration Website

ASP. NET provides a special type of website that allows you to configure security and other aspects of an ASP. NET web application. So in this section, I'm going to walk through how we can use this ASP. NET configuration website. Now, the particular website we're going to talk about you can access directly through the project menu in, in Visual Studio or Visual Web Developer Express. And you'll select ASP. NET Configuration. Once you do that this web application shown here will be displayed, and you could do several different things. You can set up smtp servers, you can set up app settings. You can set up different types of provides your application may use but we're going to focus on the security aspect. So you'll notice from the screen shot here that we can control users, we could set up different roles and we can actually hook users into those roles, and we can also set up access roles. And these access rules will control who can get to which files or which folders in your application. And what's really nice about this is number one it supports several different auth, authentication types, but number two, you can just click on the different items you need. And it's going to write the XML configuration code for Web. config for you for that given web application. So it's very simple to get started with and to use. So let me show you an example of using the ASP. NET configuration website.

Demo: Using the ASP.NET Configuration Website

To use the ASP. NET configuration website, we first need a project to work with. So, I'm going to go to File > New Project. We're going to create an ASP. NET web application. Now, depending on the type of project you create, you get different default security. If we do an ASP. NET web application, that out-of-the-box forms authentication will be enabled. If we do an empty Web Application, then nothing will be defined in the Web. config file. And so you'll default to a Windows authentication project. So that's just something to be aware of. But we're going to go with the Forms Authentication one to start. And we're going to work with this configuration website. So let's call this ConfigWebsiteDemo. And we'll create our project. Now, before we go up to Project > ASP. NET Configuration, let's run into Web. config and let's take a quick look at what's been added for us by this default project type. So the first thing you'll see is a connection strings area. And the connection string is called application services, and it points to the data directory to a database called aspnetdb. Now the data directory is a key word that is used to represent our app data folder that you'll see right here. But you'll notice that if I go to Show All Files, there's nothing in there right now. That's because that database will be created for us on the fly. So I'll come back and revisit that in just a moment. Now, moving on down, you'll notice our authentication mode is set to forms, and inside of that we have a login URL that points to our account. And we have a login page. And then you'll also notice we have registration as well as change password pages automatically added for us. So you can see right off the bat we're doing forms authentication. Now, the next important piece is, we have what's called a membership provider configured. Now, a membership provider basically says we would like to use Microsoft's default database, and we're going to use a SqlMembershipProvider class to talk to that database. Now, that's the database that currently does exist but it will in a moment. And the important part here is not only we're defining that we're going to hit Microsoft's database using this. But we're also defining the connection string that would be used so that this class knows, hey, here's how to talk to the database. So if you recall, that ApplicationServices is the same name that we have up here for our connection string. Now, moving on down we can also store personalization data. That's what a profile is. And then we can also do roles. Now, you'll notice, currently, roles are not enabled. So they're disabled by default, but in a moment I'm going to go ahead and enable those. But we won't do it through web config. I can certainly come in and type true there, but I'm going to show the ASP. NET configuration website. All right, so the first thing to show you is, let's go to that website, and this will pull up, and it's going to actually reference in the query string the path to our application we're working with. So you'll notice up in the query string, we have config website demo, that's the folder. And then we get into the actual project itself. Now, from here, we can go in and configure app settings, we have a SMTP settings if you want to configure an email server for sending email. We can take the application offline if we're going to do some work on it, and we can even configure debugging, tracing and error pages. We're going to focus, though, on the security section. But you'll notice that when I click on Security, it says cannot load type ConfigWebsiteDemo. Global. Now, the error's a little bit misleading, and what it really means is, we don't have a database yet. It's trying to hit it, but as I showed you earlier, if we go to App\_Data and click Show All Files, there's no database there. Let's fix that real quick. So I'm going to right click on default ASPX, view it in the browser, and we're going to try to log in. Now, I don't even have a database so obviously I don't have a user yet, but what this will do is as soon as I try to log in here, it's actually going to create the database for us on the fly. So we'll let it sit here. There we go. Said we were unsuccessful. We don't have a user, so that makes sense. We'll come back, and now I'm going to right-click and say, let's include this database in our project. So now we have a default database that can store users and roles. And we haven't had to do anything at this point, so pretty nice. Now, I'm going to go ahead and show or hide all the files there. And let's go back into our Data Explorer. You'll notice there's our new database. Let's just take a really quick look at some of the tables. Now, we're not going to need to know much about these, but it's nice to know what it stores. So the first thing it does is, we have an applications table, and this stores all the different applications. So this database could be shared across multiple applications, and I'll talk about that in an upcoming section in this module. We can also store membership information. We can store personalization information, roles, users, and even associate users with roles. And so this default database that ASP. NET provides out of the box stores all that data for you, and passwords are even stored very securely. They use things called salts and hashes to make it very secure. Okay, so now that we have a database, we can come up to Project, go back to ASP. NET Configuration. Let's go back to Security. You'll notice that we get a different screen now. Now that we have a database, we'll notice that we have no users. Roles are not enabled. It read that from Web. config. And then we can even create access rules. So I'm going to do is, let's say that in our application, we'd like to come in and add a new folder, and we'll just call it Secured. And in Secured, I'll add a new web form. And we'll call this super secure web form. And add that in. I'm just going to put a message. You made it past security. All right, so this could be an admin screen, we'll say. Now, right now, if I run it in the browser, we can get directly to it, and that would be expected, of course, because I haven't secured it. So let's go ahead and use the ASP. NET configuration tool to automatically secure any file that's in this Secured folder. So, we'll come back into the tool, and first thing I need to do is I need to add some users. So, let's create a user. And you'll notice in this case we get a little error. Occasionally, if you ever get that, go ahead and close it, run back to ASP. net Configuration, and every now and then it will time out. So we'll go back in. We'll do, jdoe. Let's do a password. And an e-mail address. You'll notice Roles are not enabled yet. Yet, but we're going to come back and add a role, so we'll say Create User. 'Kay, hit OK, and now we can add more users if we'd like, so it's pretty convenient, very easy to get users into this database. Now we have one user, but we don't have any roles, so let's go ahead and try to enable roles. That's now enabled. That just updated Web. config for us. Now we can say Create or Manage roles. Let's add a new role called admin. Okay, so now we have a role. Now we can say add or remove users. Let's click on Manage, and we'll go and search for jdoe. There it is. And we want to say that the user is in the admin role, which you can see the role right up top, there. And we'll hit Back, and that's going to take us back to the screen we're at, and then we could add other users if we want. But if we go to Security now, go to Manage users. You'll notice that jdoe if we come over, let's go to Edit roles, is now in the admin role. We didn't have to do anything because we've already hooked that up. So it makes it very, very easy without having to write any custom code to integrate with this database that Microsoft provides to add your users and your roles. Now the final step is we need to create some access rules. We added this Secured folder. And I could just secure the file, but I'm going to secure the entire folder and say, anything in there you have to be in the admin role to get to those files. So let's come back in. We'll create an access rule. And the first thing I'm going to do is, on the left, it's going to show you your project. And so if we scroll on down, there's my Secured folder. So I'm going to go ahead and click on that. And right now, we're going to say that the role Admin is allowed. So we'll hit OK. Now we're going to come back into Manage Access Rules. We're going to go ahead and click on Secured. And right now, you'll notice, this did work, so we are allowing admin, but all, all users are currently also allowed. Now, that's not going to work so good. So if I try to go to this page, I can still get to it even though I haven't logged in yet. Now the reason for that is we need to go in and actually add an access rule that says don't allow all users. So we'll add a new access rule, come back down, make sure we click on our secured here just to make sure we're on the right one. We're going to say all users, deny, OK. So now let's go back, we'll try to go directly to this super secure, and let's see what happens. All right, perfect, so we can't get into it now, but jdoe should be in the admin role. Let's try to log in, it now redirects me back to super secure, and now we're in. So we're good to go. So that's how easy it is to use the ASP. NET configuration site to add users, add roles, and even set up access rules. And let me just show you what Web. config, how things were modified in it. So, connection string was already there. We already had our forms and some of this other information, but you'll notice that role manager is enabled and set to true. And so now, everything is configured. The database has all the data we want inside of it. And again, if I try to go back to super secure, we shouldn't be able to get to it, we have to log in. Now, you may wonder, how did ASP. NET actually know that if we hit the Secured folder, that it should redirect to the login page. Now, of course, we added those access rules using the website configuration tool. But, where was that stored. Well, let's go back. And if we go to the main Web. config, we've already scrolled through this a little, bit and you'll see we have the standard stuff. We have our connection strings, we have our authentication for forms, we have our membership provider definition, profiles and roles. And that's it. There's nothing regarding the admin rule. Well, when we set up the access rules using the tool, it actually added a Web. config file into the Secured folder. Now, to show that we can do Show all Files, I'm going to right-click and say Include in Project, and now you can see it shows up, it's not grayed out. And let's open it up. So you can see that right off the bat, it's pretty obvious what it's doing. So when we added the access rules, it added this authorization tag. It said we want to allow the admin role and deny everyone else, and that's what this is saying here. That's how it actually knows when you try to hit the SuperSecure. aspx. Because this folder has this web config, any file in this folder would only be accessible if you're already logged in and you're in the admin role. And that's what's going on behind the scenes. So that's an example of how we can use the ASP. NET configuration website tool to create users, define roles, put users into those roles, and even define access rules. But we didn't have to write any custom code to interact with the database or even set up the custom Web. config file that was added into the Secured folder. So it's a great tool to help you get started securing your ASP. NET applications.

Using aspnet\_regsql

ASP. NET makes it easy to secure websites. And up to this point, we've seen how the ASP. NET configuration website tool can be used to define access rules as well as users and roles. Now, during that process, we saw that behind the scenes, ASP. NET can create a database by default for us called ASP. net. db. Well, ASP. net. db by default, runs under SQL Server Express. And while SQL Server Express works very well, you'll certainly want to move that up to a more enterprise level server at some point, especially if your website gets a lot of traffic. So in this section, we're going to to talk about a tool called aspnet\_reqsql that can be used to install the membership tables, store procedures, and other database entities that you need to store users' roles and other things. This tool, aspnet\_reqsql is very, very simple to run, provides a wizard, and you can get this up and running in literally a matter of seconds. So up to this point, we've seen the ASPNETDB. MDF file. And, every example I've shown has put it in the App\_Data folder. That means it's not shared across multiple web applications. We're only using it for this one. So, if you had 50 web applications that all use forms authentication, then by default, you'd have 50 instances of this database, and obviously that might not be the best way to go. So using this tool, aspnet\_reqsql, we can install the necessary tables in a master server such as SQL Server Enterprise, and then all web applications can share that same database. The database fortunately supports all the tables and necessary items to make that possible. So when you run the tool, it'll ask you if you'd like to configure SQL Server for application services, or if you'd like to remove application services from the existing server that already has the database. We'll be doing the configuration part in this particular demonstration that I'll show you in a moment. From there, we can go in and we can select the server name for where we'd like to install the membership database. It's going to prompt you, do you want to use Windows authentication or custom user name and password? And then you can either type the name of the database. You can select Default. In that case it'll call it aspnetdb. Or you can even select an existing database you might already have and have all the membership tables, stored procedures, and other related items. Installed into that existing database if you'd like. So let me show you an example of how easy it is to run through this wizard and what it does for us.

Demo: The aspnet\_regsql Wizard

Earlier in the module I showed how we could use the asp. net configuration website tool to configure users and rolls, and when I logged into the site initially, it created an ASP net db database for us and put in the app data folder. Well we're going to assume that either based on traffic or just the fact that we want to share the database across multiple ASP. NET web applications, that we want to move this into a full blown SQL server instance. Such as SQL Server Enterprise. Well we could actually attach this database right here, because it's a SQL Express database, into SQL Server and that would work. But let's assume that maybe we don't have it or we want to create it from scratch. Now, to do that you can either use a Visual Studio command prompt, if you have Visual Studio installed, or, if you're using Visual Web Developer Express, you need to know how to get directly to the ASP net red sequel tool, and that's what I'm going to demonstrate here. So, I'm going to run off to Windows Explorer. We're going to go to C drive. We're going to come into Windows, and we're going to scroll down to Microsoft. net. Now once you get to there, you basically have to go to the target framework, and since we're on. net four here, we'll go to framework in v four. Now we can simply scroll down. We'll find aspnet\_regsql. Which is right there, and we can run it. So you can run it just by clicking on it and that starts up a wizard. Or if you prefer to use the command prompt you can do that as well. Now if you're using Visual Studio you can actually go to the Visual Studio command prompt, and simply type aspnet regsql and this wizard will pop up. So that's both ways of doing it. So, we're going to hit Next. This is going to take us to Configure SQL Server for application services, and we basically want to install the database that has our membership, our users, our rules and, and that type of thing in a custom database that we have. So from here we can simply click on Next, and now it's going to prompt us for the server where we want to put this database. So this'll be your sql server name. Now, WallingDVPC is currently the actual server I'm running on. But let's say that this is on a separate machine, a separate box, and I have one called WallingDI7 and this is a sql server and it sits on a separate machine that I'm going to get to. Now from here, I'm going to use Windows authentication, and I'm going to select the default option. But, if I wanted to install this in an existing database. Then I could certainly click on the database and now it'll put those membership tables in that database. Instead of it, in its own unique database. So it kind of depends on what you want to do. Now me personally I like to have a standalone database so I know it's specifically for membership, but it really depends on what you're doing. We could come in and type a name here if we want a custom name, or I can just leave default. And what that'll do is leave it as a ASP. NET DB named database. So we're going to go ahead and do that option. So we're going to hit Next. It's going to tell us about what it's about to do, so we can see we're going to go to Wally DI7, we're going to name it ASP. NET DB, we're going to hit Next. And now it's going to go out to the database. And perform the action. And now it says it's being created. We'll go ahead and finish. So that's how we can get to the tool. There's the path again. Right up top. And once we run that tool we simply run through the wizard. Now we have that membership database available. Now what do we do from here though? We're going back to our application, I'm going to go ahead and leave that database in, but we're not going to use it now. And I'm going to go to web. config, and now what we can do is we can change our connection string. Now the membership provider. That's defined right here, points to this connection string. So the only place we'll have to make a change is right here at the top in the connection string area. So I'm going to come up and we're going to set the data source. Instead of SQL Express, it's going to be Walleen DI7. We're going to use integrated security, which means use my Windows Credentials, and now I need to take out all of this information here. I'm going to come back, and we're going to leave that. I'm going to take out the user instance, and I'm going to say that the database is aspnet. db, that we want to hit. So now we have our data source. We're going to use Windows security, my credentials that I login with. And the database we're going to hit is here. And you'll also see something called initial catalog. Now there's different ways you can do connection strings, so we'll go ahead and leave that since I've shown both. But we now have a data source, which is our server. And we have the database we want to hit. And that's all you have to do. Now the membership provider and, if we scroll on down, the role provider as well, will also use this same connection string. So now we're going to be hitting our full SQL server and hitting that ASP net database. Let's go ahead and save that. Now, if we run, let's right click on Default. aspx and View in Browser, and earlier. When I ran the ASP. NET configuration demonstration, I showed a J Doe account I created. Well that account's gone now, because we're hitting a different database. So let's try to log in and you'll see that it was very quick and it didn't do it obviously. So we can go ahead and try to register. Let's go ahead. And this time I'm going to do dwaleen. I'll do test@test. com and a password. We'll create a user, and it looks like it worked. We're now logged in. We're now hitting a separate database entirely. Than the one that was local. So now we can reuse that database across multiple ASP. net applications. Let's go ahead and make sure that that user actually showed up. So I'm going to create a data connection. I'm going to right click on our data connections here. We're going to go to add connection, and we're going to change this to hit a SQL server now. Not a SQL express file. We'll hit OK and I'll type in wall in d I7. We're going to use Windows authentication again and now we should be able to get, there's my ASP net DB. So let's go ahead and we'll test the connection, it looks good and now we can actually hit that SQL. SQL Server. You can see all the membership tables that I showed earlier in this module are available, and let's just run off to Users and we'll say, Show Table Data by right-clicking on it. And you'll notice there's my user. So you can see everything's now working perfectly. So that's an example of how we can use the Aspnet regsql. Tool to create a ASP. NET DV type database, a membership database, to store users and rules, but instead of using it locally, we want to share it, and we can actually create it in a full blown SQL server instance.

Security Server Controls

ASP. NETprovides several different web server controls that you can add into your website to secure different parts of it. So, we can log in, we can. We can deal with passwords and much more. So in this section, I'm going to walk through a few of these controls, and then I'll demonstrate how you can use them in your web applications. So here's a list of the different controls that are available. Logging the users in and out. Typically we have to drag on labels, text boxes, a button. And that type of thing to get the user credentials. And then we'd have to write code to call in to the membership provider, hit that ASP. NET database, and all that. Well if you use the login control, it does all that for you. You can literally drag it on, gives you username and password functionality, and when you hit enter it automatically has the code built in to hit the database and check to see if that user has entered a correct username password. Login view provides a customized way to template a control and provide different views of data if a user is logged in or logged out. PasswordRecovery is pretty obvious. That one is a very simple control to use. You can simply hook it up and make it so if the user's password is encrypted or stored in clear text then you can recover it and email it to them. LoginStatus is a very simple control. It just shows either the login link or the logout link, based on the status of the current user. LoginName will actually show the user's name. And then, at some point you might need to register users. And that would be the case typically on internet sites, where. Before a user can log in of course they have to register. I've done that a couple times throughout this module actually. So we'll look at the create user wizard tool and control, and then finally if you need to change your password there's even a control for that, so you can see all the major stuff. Is already covered. Literally you can drag this controls on to your web forms, and you can get going and then you can change the appropriate properties. So, let me just take a really quick look and walk you through what these look like. So, the login control looks just like a standard web server control, you can see we have ASP. P login. We give it an ID and we have runat server. Now aside from those standard kind of staple attributes or properties that you'll add, you'll notice that we've also added in this example, create user URL. That way, if they go to the log in screen, but they haven't registered. Yet. You can give them the URL, and it will automatically display it for you. You can also go in and give the text for the URL, if you like. And that's the create user text. So we're going to say, in this case, credit user account. So very, very simple. And you can even customize some different templates for this. And the CreateUserWizard control is the registration control. So this automatically will capture things like the username they want, their email address, and their password, and all we have to do is drag it on. It puts all the key stuff, and then, we can also customize through templates and steps in provides. The different data we want to capture. Now one of the things, once they've registered, that you might want to do is set where do they go once they actually click the button to register. And you can see that there's this continue destination page url. In this case we'd go to default. aspx. So very, very simple. And you can even configure all this of course. Course visually in Visual Web Developer Express or Visual Studio through the properties window. The next one is the logging view control. And I really like this one because the logging control just shows links. It just shows log in, log out. Based on their status. Well, sometimes you actually want to show more specific messages, more than just a, a login type of link. So in this example, we're using two of the templates that are provided, AnonymousTemplate and LoggedInTemplate. And they serve the very obvious purpose of, if the user's anonymous, they haven't logged in yet. We're going to show that text, and if we didn't have the log-in page right here. In this case it assumes that the log-in control, of course, is on the same page below. Then we could put links, we could put whatever we want inside of this template. If they're already logged in, we're going to go ahead and say, welcome, and then we're going to to use the LoginName control to automatically write out their user name, and that will take care of that automatically. Just by dragging the LoginName control into this LoggedInTemplate. So you can see it's pretty easy to get started with, and what's really nice about what I'll show you in a moment. Is you don't even have to write a lot of code to do this. This is showing you the code just so you understand what happens. But most of this, you can just drag and drop onto your web forms. So let's take a look at how we can use a few of these different controls.

Demo: Login and Registration Controls

In this demonstration I'm going to show how we can use some of the different asp. net security control so I have the time tracker application that we seen a couple times across the modules in this course and what we are going to do is simply secure the time sheet viewer. But I'm going to build the login, the registration, and some other related screens dynamically so that you can see everything and how it fits together. So right now, in the secure folder, you'll see that I have a web config, and I've already secured the application to only allow the admin role. This is very similar to what we did earlier in the module. So now if you try to right click and go directly to that file, you'll see that we're going to be taken to a login page, but we don't have one yet, but you'll notice up top we've got login. ASPX. Now, how did it know to go there? Well if we go to the root web dot config you'll see I've already enabled. Forms authentication, but I didn't specify a log in page. Well by default asp. net will automatically use login. aspx as the default name, so you can certainly change that, we could come in, and we could add a forms tag. And then there's a login URL that we could give a different name if we'd like. But I'm going to go ahead and just leave the default. That makes sense to me. And we'll go ahead and add our login page. So now we need a way to log in. Now, as a point of review for the module, I've already gone in. To the ASP. NET configuration site. And under security I have a user called jdoe and jdoe is in the admin group for roles. So we're all set up there. All right. So let's go ahead and get started then by right clicking on the project. And we'll do add new item. We'll do a web form, and we'll call it login. aspx. Now, I'm going to put this at the root of the site, you definitely don't want to put this in the secured folder, of course because you'd never be able to get to it, so we're going to put it right at the root of the site, and from here I could go to the toolbox. We could switch do Design View. And I could start dragging on my text boxes and buttons and all that stuff. But obviously, the reason for this demonstration is to show some of these security controls. So, you'll see that in the toolbox we have a login section. And that login section defines the different controls that we can use. So we're going to go ahead and drag a logging control and now I'm going to click on auto format and there's some built in templates and of course with CSS. And different colors and things you can certainly customize this even more, but these are some defaults for you. And we'll hit OK. Now from here, I'm ready to go. This control literally knows how to call into the App Data folder and get to the ASPNETDV file, which is our SQL Express database. So, I'm ready to go. However, if they come to this screen, if they tried to go directly to the Secure file, they're not going to be taken to this screen, but if they haven't registered we probably need a registration screen. So, we need to be able to link from here to a registration screen. So let's go ahead and add that screen first. We'll do Add New Item and I'll just call this Registration. aspx. And in here, we're going to go back into, to Design view. We'll go to the Toolbox and now I'm going to drag on a CreateUserWizard control. Now, this actually uses a control that's built into ASP. NET called the Wizard Control. It allows multiple step processes to be defined. So, if you want to break a large form into pieces you could use the Wizard Control. Well, this is based on that and you'll see that I can actually come in and do multiple steps. So, right now we only have two steps and we're going to go with the default for this demonstration. But I could certainly go in, and we can add and remove different Wizard steps just by clicking on the different button here. And that would allow me to actually capture additional information if I want more than just username, password, email, and some security info. And I can break it up into multiple screens, which is pretty nice. So, we're going to go to Auto Format, we'll select the same color scheme as I already did for the logging control. But, we do need to define, once they've filled this in and they hit Create User, as shown by the steps, what'll happen next is they'll go to the complete screen. And then there'll be a button they can click to get out of that. Well, how do we set that? Where do they go once they click it? Well, if we go to the Properties, we can scroll on down into our C's area. Let me make this just a little bit wider. And you'll see that we have a ContinueDestination page. Now from here, I'm going to assume that once they've logged in through this registration screen, which will happen by default, that they want to go directly to the Time Tracker, or at least try to. Now if they can't make it, they'll be redirected back to the Log In. So, you could certainly change this up a little bit, but we're going to go ahead and click on Secure. Click on TimesheetViewer and hit OK. Now, when I hit the Continue button after filling out this information, they'll automatically be taken to this TimesheetViewer screen. And again, you might not want to do that. It depends on your application. Okay, so we'll save that and close it. We're done with the registration screen. You can see how easy that is and if we go to the source, you'll see that it actually has two Wizard steps here. So we have a CreateUserWizardStep and a CompleteWizardStep. And then we can actually go in and make custom wizard steps if we'd like. And we have some custom styles, based on that general yellow-ish type theme that I picked. Okay, so, on the Log In screen, we need to build a link to the Registration screen because if obviously they haven't registered, and they want to as long as they're in the admin role, then they'd be able to get in, if not they won't. So, the easy way to do that is we'll come to the Properties. And we can actually go in and say, what is the page they would go to for the CreateUserText? And so, you can see CreateUserText. And CreateUserUrl which is by user text we mean the Registration screen. So, CreateUserUrl would be our Registration aspx, and I'm just going to say, Register. You'll notice that automatically puts the link. Now we certainly could've done that on our own, you know, above or below this control, but that sure makes it easy to work with and it does it automatically for us. That's just built in to the control. Now, likewise, this control is just consi con, consists of different tags and you could see what we just did is added as a property. Okay, so at this point, now we have a Log In screen. So if we try to go to TimeSheetViewer directly, let's view it in the browser, we're now taken to the Log In screen, and let's go ahead and see what we have at this point. So, we have, we're going to do jdoe and the password I entered. We'll log in. And, you'll notice I'm now in to the TimeTracker, so that worked very nicely. We might want to show some other information though. Maybe on this particular screen up here, we'd like to below this put their Log In status so they can log out. We might even want to customize the view of the Log In. So, let's take a look at how we can do that.

Demo: Login Views

There's two main ways to let the user know their Log In view status. And if we come over to the Toolbox, you'll see those. We could do the LoginStatus, and that simply displays a login or logout link by default, or we could do the LoginView control. So, let me show you both of those. So, what I'm going to do is come over to our master page. I'll switch over to Design view and right up here in the header, I'm going to drag on a LoginStatus control. Now this particular control has our Logged In, Logged Out views and what it's going to do is, if you're logged in, it'll say Logout. If you're logged out, it'll say Login as you see right here. So, if we run this as is, let's go to our home page, you'll see I'm already logged in actually. So we'll log out, and we'll log back in, takes me to the Log In page, so a very, very simple control to use. Now, if you want more control over what displays, then you can actually use templates that are provided by the LoginView control. So, the LoginStatus just shows the links by default, the LoginView provides an anonymous template and a logged in template that allow you to customize what's shown in those two different modes. So we can drag that on, and what this allows you to do is use these two templates that you see here. So we have both of those that are available. And you can simply drag and drop into here, or you can come into the source. And since those aren't in there yet, let's just go ahead and add them to show that. And we can do our anonymous template, and we can do our logged in template. And it makes it very easy to now add different things into these templates. So I could say please log in and we could just put some basic HTML in here if we'd like. And I'll say Login. And then in the logged in template, we could say Welcome User. And let's just do that for now and let's see if we can get this working first. So, I'm going to run off to Default aspx. We'll view in browser. And you'll notice it says, Please Login now. So I have control, let's go ahead and Log In. And you'll notice now it says, Welcome User. Well it'd kind of be nice if we could personalize it a little bit more and say, welcome name of the user account, J Doe, or whatever it may be for their username. And we can do that as well. We can actually come in to our master page. Come back in here instead of Welcome User, there's another control, in fact we can just drag this in to save a little typing called LoginName. And the LoginName control, we'll simply write out Welcome in this case, J Doe, because that's who's I'm logged in as. So, let's go ahead and try that out. All right, see where it says Please Login? We'll go ahead and Log In now. Do J Doe, and we'll Log In and you can see now it says Welcome J Doe. So, very nice because I've yet to write any C sharp or VB code. This is all just built into the security controls. Makes it very easy to work with. So, that's the main controls I wanted to show at this stage. And let's go ahead and now put it all together. Let's go in, we'll create a new account, we'll register, we'll then log in, and then we'll hopefully get to the TimeSheetViewer and I'll show you what's going to happen though, security wise. So, first thing I'm going to do is just try to go to TimeSheetViewer directly and if we're not logged in then nothing will happen there. We'll go to the Log In screen, like we'd expect. But I don't want to Log In in as J Doe. Let's say in this case that I am G Doe and I'm going to give it a password Give it an email, and we'll kind of fake the security question. We'll hit Create User. Okay, so it just created G Doe, and when I hit Continue here, it will now take me to wherever I told it to in the continue URL. Now, recall that earlier, when we did the registration, I set the Continue URL to actually go to the TimeSheetViewer. And we saw that in the different properties that we have here and we have ContinueDestinationPage URL. And that ContinueDestinationPage went to here, so why didn't we get in? Because we did login successfully but it took us to Log In. Well, the answer is that we are a user now, but recall that only admin users can get into this and this user is not an admin user, so you could see the security worked. So, I'm going to come back up to our configuration. Let's run into that user real quick. And we'll manage, let's go to gdoe, we'll edit roles, check Admin, now we should be able to get directly into that role. Let's go ahead and try to go into TimeSheetViewer. Or go to the Log In, we'll do gdoe, and we'll do our password. We'll Log In and now we're in where we'd like. So that's an example of using several of the controls. Now, when you create a brand new ASP. NET web application, you'll also get the Log In and the Registration created for you. I wanted to show how to go through them from scratch so you saw a few of the properties. But, let's go ahead and close this to wrap up and I am going to create a New Project from scratch and we'll do a web application and we'll hit OK here. And you'll notice that, right off the bat, there's this Account folder and they already have the Login, the Register and even a Change Password screen, which uses our ChangePassword control. You can see that control right here, where they can change it. And that's all done for you. And there's other controls you can use as well, but they'll help get you started out of the box with the default web application template. Really helps out in kind of jump starting the process, but now you know how everything fits together. And some of the other controls that you can use as your adding security into your web application.

Summary

In this module we've seen the different types of security that ASP. NET supports including Forms and Windows Authentication. We've also seen how different security settings can be added directly into web. config, and how we can manage that using the ASP. NET configuration website. And using that makes it easy to create users, roles, add those in. Plus we can even visually define our access rules that ultimately get added as XML into the web. config file. Now, through this process we saw that ASP. NET does provide a membership provider and a role manager. And the membership provider knows how to talk to the ASPNETDB database. And the role manager does as well. And I demonstrated how those two providers use the connection string to find in web. config to determine which database to hit. Now, a lot of the demos as I showed them we use the built in ASPNETDB and it was just added into the App Data folder. But, I explained how, well you might want to move that out into a more enterprise SQL Server instance. And we can do that quite easily using the aspnet\_reqsql. exe tool. And that was the Wizard I walked through, which is very, very simple to get going. And then finally, we ended with a discussion of some of the different server controls, such as the Log In, the Registration, and the Log In View controls that can be used to easily secure your site, allow the users to log in or register, and do much more. So security, obviously, is a very important part of applications and ASP. NET provides the necessary tools you need to implement the proper security.

Using ASP.NET Dynamic Data

Introduction

Hi this is Dan Wahlin with Pluralsight. In this module, we'll talk about a form of ASP. net webforms called dynamic data. This is a feature that's been around since ASP. net 3. 5 was released. And I'm always surprised to hear how many developers have either never heard of it. Or never taken the time to research it and see what it offers them. So in this module, I'll walk you through what is dynamic data and what are some of the features it offers us. And you'll see it can really offer a huge productivity boost, from there we'll jump into what are the different project types you can use in visual web developer express or visual studio. To create a Dynamic Data Project. We'll talk about how to create a Data Model that exposes data to a Dynamic Data Project. How to register that Data Model and then we'll end with, how do we modify Dynamic Data websites, so there's a lot of functionality. That dynamic data websites can offer you, especially if you have very data driven applications, where you have Inserts, Updates, Deletes. Maybe you need to do filtering operations, and those types of things. So let's jump into what exactly is ASP. Net Dynamic Data.

What Is ASP.NET Dynamic Data?

Let's discuss what ASP. NET Dynmaic Data is, and what it can offer us as developers. And really what we're after. Here is productivity games. And you'll see that it really delivers on that for certain types of applications. So, if you ask the question, what is ASP. NET Dynamic Data? Really what it is, is a way to create a data driven application but without having to write really. Any code at all to get started. Now you can certainly go in and tweak things, and you probably will. But out of the box you can get a website up and running that hits a database with many different types of tables in it. And have full support for Inserts, Updates, Deletes, of course selecting data, and even filtering data. It's fully customizable, so, once this website is generated from a template, you can go in and change things to make it show fields, show differently, for instance, you might have a date field, and you'd like to show a date picker. That's possible. So, you can use link to Sequel or Entity Framework as your data model, to communicate. Between the website and the actual database. Now if you've ever needed to create a database maintenance screen, this is a huge boom to productivity. Often times going back to the old days I would have to create some screens for different clients. To allow them to update look up tables update the main tables the whole sum of their core data and perform other types of data activities. We didn't want them going in to Sequel Management Studio or earlier it was called Enterprise Manager so what dynamic data allows. Is you can create a website that will automatically have full support for CRUD Operations, Create, Read, Update, Delete. An example of that's shown here. Here's an example of displaying the products table. And you'll notice that I can actually drill down and filter by product category. By product model. And then once I get those I have built in support for paging. I can enter new items. And of course I can edit an item. Delete it. Or even drill down and get more details. About that particular role of data. So it provides a really simple approach to generating automatically data driven websites. Now, some of the key features are. The validation that you might have to find in a data base. Such as, certain fields are required. They have a certain length. Things of that nature can be constrained. And, whatever the data model that you use, whether it's framework or link to Sequel, has, that's what the dynamic data website will use as well for validation. So it's very easy to get it set up. It's also easy to go in and change the markup for. Different fields that might appear. So, you may have, as I mentioned earlier, a Date, Time field. And instead of just giving the user a text box, you can go in and modify a particular file; which I'll show you at the end of this module. To, for instance, show a date picker. Makes it much easier and less susceptible to errors. It does have support for many-to-many relationships, if your database and your entity model supports that. And entity framework does that quite well. And it does support field templates, as I mentioned, such as date-time. And custom attributes and validation attributes. So you could say this field is required, this has a string length. You could even use regular expressions to match on patterns if you'd like. And then finally, it does support data filtering, which makes it really nice so that if there are a lot of rows. A user can easily drill down and find the data that they'd like to look in to. Now, all this works based on a concept called scaffolding. And so, ASP. NET Dynamic Data will generate pages. That can automatically read the meta data from a given table in the database, and it does this again through link to Sequel or Energy Frame Work. And then automatically generate all the different pieces you would need to perform the edits of that data. And as I mentioned that includes the CRUD operations, Create, Read, Update and Delete. So, that you don't have to drag on a grid view control or a details view control to actually perform the edits. That's all built into the scaffolding process. And you'll see that in the next section where we actually go in and create a dynamic data project. Now, scaffolding really is just a bunch of templates. So you have page templates energy specific templates for objects field level templates and even templates that are specific to filtering. So that's the type of thing dynamic data offers and where for me it's extremely productive. Is, we work with a lot of clients, and the different databases we use need admin screens. So one of the more simple uses of dynamic data websites are what I mentioned earlier. We can very easily create maintenance screens for our databases, which is very nice. Now you can even do much more than that, though. But we'll focus on that type of activity in this particular module.

Creating a Dynamic Data Website

Editors such as Visual Web Developer Express of Visual Studio make it. Really easy to create and work with dynamic data projects. Whether you create a web application style project or a website project, you can select from built in templates. So this example shows how we could go in. Create a new website and then we can select a dynamic data template. If we'd like to use Entity Framework for our model, that will communicate between the database and the website. Then we can select the ASP. net dynamic data entities website. If we'd like to use Link to Sequel, we can of course select this highlighted one right here. Now regardless of which template you select, Visual Studio will then apply that template, and create a specific type of web application project or web site. And you'll see this DynamicData folder. This is where all they key pages that are used in this dynamic data website are contained. And this is actually where you can go to customize things if you'd like. So let me show you an example of how we can use these templates to create a dynamic data website. And what that website offers us out of the box.

Demo: Using Dynamic Data Templates

It's really simple to get started creating an ASP. net dynamic data website. And we have two options actually, we can do websites or we can do web applications. So if I go up to file New Project, we can create. A web application project. And you'll notice that we have ASP. NET, Dynamic Data entities web application and one for link to Sequel web application. Or, if I just wanted a website project, we can do File, New Website. And we have the same type of thing, but you'll notice it's now. A data entity's website or linked to sequel website. So in this example, I'm going to do an asp dot net dynamic data entities website. And let's go ahead and call this, Adventure Works Light Admin, because that's the database we're going to work against. And we're going to make some admin screens throughout this module. So I go ahead and hit OK. And its going to take us to a file called global. asax and give us some information about what we need to do. Now we'll get to that in the next section of this module but for now just note that we have some nice comments here on what we need to do and ultimately we need to go in and register. Our Entity Framework model so we're going to put in the name of our model right there. So we'll come back to that in a later section. Now let's take a look at what we get out of the box though. First off you have a site master so we have a master. We can go in and certainly tweak things. Right now, Dynamic Data Site is our Kind of header if you will. But we have some built-in style sheets that you can go in and tweak those as well. So we have a site. css and this will allow you to change the different styling of the different pages that you'll see, that will be generated. As I do those later in this module. And then we have this DynamicData folder. Now this has the different templates I mentioned earlier. Entity templates can be defined here so we have our Edit our Insert our Selects. Different content such as Images can actually be defined here. You can see that we have some paging images. That are available here, a little arrow. We can also come in and you'll see that we have in addition to those field level templates. So if you wanted to go in and modify how date time fields are edited,. You'll notice there's a text box here and this text box is going to be used for editing date time type fields. Well we can go in and tweak that. Or maybe we want to tweak the way just text in general is handled. URLs, Booleans, and you can see there's several other things here. We also have a folder that handles filtering our data, and then we have our actual page templates for our Inserts, our Updates, our Edits, or Selects, and drilling down into details about what'll happen as they click, click on a particular row. So all of this code here would obviously take quite a bit of time to make on our own. This is what we're given out of the box. This is our scaffolding. This helps us once we make a model very easily generate the insert update and delete operations that we're after. So that's all we have to do though to create a Dynamic Data website. Now the next process is we need to create a model. And that's what we'll look at in the next section.

Creating a Model

In the previous section, you saw how easy it was to create an ASP. NET Dynamic Data Project. Simply select the appropriate item, and it it'll do most of the heavy lifting for you. In this section we're going to talk about how we can create a Data Model that can act as the go between, between the database and our website that has all the scaffolding to enable Inserts, Updates, Deletes, and filtering operations. So you have two options when you create model. We can go in and do schema first or model first type of design. And I'm going to go ahead and just base our model on an existing Adventure Works Light database that I'm going to add to the project once I get to the demonstration. Now you could also choose link to sequel though. That is another model option that's available, but because I did entity framework for the dynamic data project created earlier we're going to go ahead and go that route. So to do that I'm going to walk through the process of adding an ADO. net, an entity data model into our project. Walk through the wizard to create a database. That will create a connection string for us automatically, so it's very, very simple. And then we'll simply check the check boxes next to the appropriate tables that we'd like to expose. Now once we do that, an Entity Framework diagram will be created for us and it will have the. The different tables that we'd like to expose. So, in this case, you can see we have Address, Sales Order, Header and some others. And whatever tables you select that's what will be scaffolded out or rendered to enable our CRUD operations, Create, Read, Update and Delete, using dynamic data projects. So it's actually very, very simple to get started with that. And let's go ahead and jump in. And add in a database and then an entity data model into our dynamic data website.

Demo: Creating an Entity Framework Model

Before we can add a data model into our website we of course need a database to base it on. Now, if we were doing a model first, it would be possible to actually generate the database. And I encourage you, if you're interested in that, to view the Entity Framework module in this video series. But what I'm going to do is add an AdventureWorksLight database into our app data folder. Now if you already have a database either at work or a host that's provided then you can certainly use that database as well. But we'll go ahead and add one, so I'm going to say Add Existing Item. I'll select our database and that's a sequel expressed database that we can now use. Now from here I need to be able add in our Entity Framework model. If we're in a web application project, I could create a sub folder or even put the model right in the root of the web application project. But because this is a website project and you'll know that. By just simply looking at the name here, notice it starts with a drive path. Then we need to add in an app code folder and that's where our model's going to go. So to do that, we'll right click on our project. Go to Add ASP. NET folder and select app code. Now from here I'll right click on App Code and we're going to Add New Item. Now the item we'd like to add is this ADO. net entity data model. And again, that will model our database and be used by the dynamic data scaffolding to automatically generate all the CRUD operations and pages for us. So, we'll go ahead and name this after the database AdventualWorksLight. edmx. Let's go ahead and add that in, and now it's going to ask us to either start with an Empty Model, or to base it on a database. Now, since I added the database, we'll go with that route. You can see it picked it up automatically, it made the appropriate connection strength for an Entity Framework a connection and model. We'll hit Next, and now it's going to go out, query the database. And we can simply select the tables that we'd like to expose to be edited. Now, if I'd like all tables, I can just check the check box and that would make all tables available. The sysDiagrams one, I probably don't want though. And everything else looks pretty good. I think build version, we can probably do without as well. Okay, so we'll go ahead and select all the others though and hit Finish. Now, this will generate the diagram that I showed earlier. And whatever's on this diagram can now be used to actually go in and work with dynamic data. So it actually makes it very simple. So there, we have our model. You can see all the different tables that were selected. And if we needed to update it for the database changes, we can right-click and say Update Model From Database. And run through the wizard again which makes it very easy to work with. So that's how we can add a model into a website type of project. Now again, if you're doing a web application project, you don't have to create the app code folder. You can just add this Edmx file anywhere you'd like in the project. So the next step is, now that we have a model, we need to register that model with this dynamic data website. So let's take a look at how we can do that.

Registering the Data Model

After we've created a data model, we need to register it with our dynamic data website so it knows about and can make it possible to do CRUD operations against our different tables. So we'll do that using the Global. asax file. Now, if you're in a website project, you'll just see global. asax. If you're in a web application project, you'll see that global. asax and then it can have a code behind such as. cs or a. vb file. Regardless, we are going to need to go into that file and because we created a dynamic data website, it'll add in some comments for us showing us specifically what we need to do. In fact, when I created the dynamic data website shown earlier in this module. It automatically opened up the Global. asax file and I pointed out those comments and mentioned that we'll be talking about those later. So what we'll be doing is we're going to find where it says, YourDataContextType and we're going to need to rename that to our custom Data Model type. So whatever you call that, whether it's a link to SQL or entity framework, you'll have to do this process. Now in addition to going in and setting the actual context type that you would like to use, you can also specify if we'd like to scaffold all tables. And if you set that to true, what that will mean, is every table in your data model will automatically have CRUD operations available. Inserts, updates, deletes, we can filter, we can do all those types of things. Now, if you're doing administration type screens and you've added your tables to your data model, then you may want to set that to true. But in other cases, maybe you don't want to scaffold all the tables. You don't want to make all those available. Well fortunately in those cases, we can use attributes to control which tables have scaffolding and which don't so that they either show up or they don't appear. So let's take a look now at how we can go in the register our adventure works light data context that I created earlier. And make that work with the dynamic data website and then we'll see the dynamic data website in action at that point. So let's jump on in.

Demo: How to Register a Data Model

To register a data model, we need to open up the global. asax code file and then add our data model into it. So, I'm going to double-click on global. asax and we're going to scroll down. You'll notice that several comments have been added for us here. About what we need to do for our data model registration, and these were added right when I created the dynamic data website earlier in this module. So very simple. So if we read through this, it tells us we need to uncomment the line below which is this guy right here. We can set scaffolding all tables to true. Only if we want to make sure that everything that's in our data model has the appropriate insert, update, delete, and filtering capabilities. It even tells us that if we want to mark certain tables as either being scaffolded or not, we can add this attribute into a partial class to turn that feature on and off. So we could set scaffold table to true above a table class, and I'll show you how to do that at the end of this demo here. So the first thing I'm going to do is uncomment this line. So we're going to say, DefaultModel. RegisterContext. And we need to register our AdventureWorksLT. Context. Now how do you know the name of your context though? Well the easiest way is whether you're doing link to SQL or entity framework. If you come in, there's a code file that'll be generated, in this case it's the designer file. And up here at the top, we can come down and see all the different tables. Now, all these are pretty obvious. They are the different tables. And here is our data context, called data entities. So I'll come down to that, I'm going to copy that into the clipboard, and we'll just paste that in. Now you'll notice it's still red because it doesn't know the namespace. So, we can either click this guy right here and import the namespace. Or we can right-click and we could say, Resolve > DataEntities. And that'll take care of it for us and it puts the namespace on front. And here's the name of our data context. So over here to the right, you'll notice that ScaffoldTables is false. I'm going to go ahead and bump that down so it's a little bit easier to see. We're going to go ahead and set that to true. Now at this point, after saving the file, that's all I have to do to make dynamic data aware of what's going on. And it'll now go in and automatically scaffold all the tables that we have available. Very, very nice to work with. So let's go ahead and test out our dynamic data website and see what we get simply by adding the data model and then registering it with our DefaultModel. RegisterContext call. So, I'm going to right click on the Project. And just to make sure everything's in good order, I'm going to right click and build it. And looks like everything built. So now we'll go to default. aspx, we'll right click on that and do the standard view in browser. Now, this will pull up the data context editor. So now, what's going to happen is all the scaffolding's going to be done for us. So you'll notice that I can now get to all my tables. We'll go to let's say customers. It's now going to go to the customers table and you'll see that I have instant paging between the different records. Looks like we have around 10 at a time. I can insert a new item if I'd like and that'll take me to an insert screen just to cancel that. We can even come in and filter and we can do edits. So we can edit this row and automatically even drill down into other related tables that are associated with this and edit those. We can also come in and do deletes. We can do the details, which makes it in more of a vertical view of the actual data. Now of course you can go into this because all the code's available. As shown earlier, there's a master page for this. We can change the themes, we can change the CSS. Everything can be swapped out to make this easy to work with and to edit. So, just by doing that one simple step we automatically have admin screens so that we can go in and edit or delete our different records. So, definitely a huge time saver. If you've ever had to build this from scratch, you'll definitely know that this is a good way to go. And at least it gives you some starting code, and if I don't like what I have, I can go into the dynamic data folder and we can start to customize things including the pages, the fields, and even the controls that display the data within this particular web site. Before we move on and talk about how we can do some customizations, let's review scaffolding a little bit. So I'm going to go back to Global. asax and recall that we added true for ScaffoldAllTables. I'm going to set that to false, and let's rerun the application and see what happens. So you'll notice, we now get an error because every table that we had in our model, every entity, has now been disabled from being scaffold. So it says there's no accessible tables at this point. Make sure that we registered our model and that scaffolding is enabled. Well we, of course, disabled it. So, we could come in to our model, and we could do what they say right here. We could add ScaffoldTable(true). The problem is I don't want to do that right here in designer. So, let' s say we want do enable the customer. And if I put this in, the problem is it'll eventually be wiped out. We can right-click and resolve this and things but what' s going to happen is if we rerun the wizard, some new table show up, we want to add into our interview data model, then this might get wiped out. So we don't want to put it in here. This is a code file we don't want to edit. Instead, what I'm going to do is right-click on app code, and we're going to add a new item, which is a class, and just call it customer. We're going to make a partial class. So I'm going to go ahead and take out all of this. We're going to come back into here and I'm just going to copy that portion. And then we need to put this in a name space. Now what name space? Well it needs to match, so whatever name space this class is in, our partial customer I'm making now must be in and there's our name space right there. So, we'll go ahead and say it's in that name space. And I'm going to take off the inheritance, since that's already done for us. Now, if we come in, we can say ScaffoldTable(true). We're going to have to resolve that. So, I'll right-click on it and say, Resolve the Name Space > System. ComponentModelDataAnnotation. Now we have a partial class that will be merged with the class that you saw here in the designer. Let's go back down to that one. Here's our partial class. And that attribute will now also be applied to our customer. So ScaffoldTable(true) is now going to be applied and we should hopefully see that we can at least get one table displayed when we run the web application. So, let's try to run this, or build it first. And it looks like we built, let's go ahead and run this now and we'll say, View in Browser. And you'll now notice that Customers shows up. So we can come in to Customers and we can edit our Customers, perform our edits, our deletes, our details. All that fun type of stuff. So that's how you can actually go in and have a little more control over which tables in your model are going to get scaffolded. So this is good for security. You may have some Admin screens where you only want very specific tables to be scaffolded. So if you want to go that route, like the one I just did, we can set this to false. We can then go find the entity that we want to actually enable. We don't want to touch the code file, of course, for the model. But we can create this partial class that will get merged at compile time with the partial class that you see right here, and that will automatically add in our entity and merge in this attribute to scaffold to table. So it works out very well if you'd like more control over that. So now that you've seen how you can control scaffolding, I'm going to go ahead and just going to rename this to. txt so it's not actually used. That'll take it out of the compile process and we're going to come back to Global. asax and we're going to set this back to true for now. Now in the next section, in the final section of this module, I'm going to run through the process of how we can do some field customization. So let's take a look at that.

Modifying Dynamic Data Sites

After you've created an ASP. NET Dynamic Data website and registered your model, you're more than likely going to want to go in and make a few changes to that site. What they give you out of the box is nice, but it definitely opens up a lot of room for enhancements and modifications. So in this section, I'm going to talk through how we can do some different modifications. Now there's a lot of things you could do. You could go in and just modify the master page. You could go in and tweak the CSS to change the fonts or the colors. You could go in and actually change how the different edit or list pages render their data. We can even go in and change how the URLs displayed up in the browser and change what's called the route. But, what I'm going to focus on is what I think is definitely more useful, and certainly real world. Something you'll find yourself doing fairly frequently with these types of sites and that is customizing and modifying fields, and the data that's displayed in those fields. Now, modifying fields is actually very simple to do. I'll show you how to get into the Dynamic Data folder and locate those templates that it provides. And then I'll show you how we can make some pretty cool modifications. So the one I'm going to demonstrate in the demonstration coming up here is we're going to talk about how we can have date time fields. And by default the time will be written out, and sometimes you don't want that and it just writes out a text box. Well, we're going to change that to look like what you see here. When the user clicks in the text box, we're going to make it so a jQuery date picker pops up. That way, they don't have to type in the date plus it'll block them from typing invalid characters that really shouldn't be in a date time field anyway. So I'll show you how to do that process and how to make this much easier to work with, from an end user perspective. Plus, it'll keep the data cleaner of course and you can certainly add validation logic on top of this as well if you'd like. So let me show you an example of how to get started with modifying dynamic data fields.

Demo: Modifying Dynamic Data Fields

ASP. NET dynamic data websites can easily be customized in a variety of ways. In this demonstration, I'm going to show how we can do field level modifications and customizations. But, at a higher level, we can always come in and customize site. master for layout and the overall look. We can come into site. css and change fonts, colors and padding and borders and all that type of stuff. But anytime we want to change how the page is rendering data or go down as low as the field level changes, we need to go in to our Dynamic Data folder that's automatically added when we create a Dynamic Data website. So you could see that we can come in and change the page templates, how things are filtered. We can even come in to the fields and that's what I'm going to do in this demonstration. So date times are currently being rendered using a date time edit. And let's look at that in action. So let's come back, we'll run our default aspx. Come in to down into SalesOrderHeaders. And we'll edit it, and you'll see that Order Date, Due Date, and Ship Date, all are date time items, but it's not too good of an experience. Number one the time is being written out, we probably don't want that, it's pretty irrelevant for these types of dates. And we also have the date, but although it displays it fine, the user can type in whatever they want. They'd have to go look up the date if they're trying to adjust the ship date for instance. And so it's not a very good user experience, we'd want to make it more like most of the airlines and hotels out there do on their website, where you get some type of a date picker. So to do that, we can come in to our dynamic folder field templates, and we can edit our date time edit ASCX. So this is a user control and this is actually what's being used any time a date, time field is rendered. You'll see right now a simple text box is being used. Well, I want to go in and change that of course to be some type of a date pic or calendar. Now one of my favorite ways to do that, is to use a script library called jQuery. So let me run off to jQuery. com. And at jQuery. com, you can get the latest version of this script library. Now a full discussion of jQuery's well outside the bounds of this module's topic, but if you're interested, up on the Pluralsight. com website there is a course on jQuery, if you'd like to get more into this. I'm going to show you the fundamentals though of how to get started with it, and how we can use jQuery Date Picker. So, the first thing I'm going to do is download jQuery 1. 61, and I'm going to go ahead and save this into our website. So, I have Adventure Works Light Admin. We're going to go to the Scripts folder, and we're going to save our script right into there. So that's the first thing I need. Now the next thing I want to do is I need to get the Date Picker functionality. Now that's not included out of the box with the jQuer script, but it's easy to get. If I click up on UI, I can come in, and we can see all the cool stuff that jQuery UI offers. So if I click Browse All Effects and Widgets, we can come down and click on Datepicker. Now Datepicker is just a JavaScript. It's a client side control that runs in the browser, that allows the user to come in, and do like we're pretty used to on different websites like I mentioned. Airlines, hotels, and things like that. That's what we're after. That's what we want to make. So to do that, I'm going to click up on Download. We need to download the theme for this, the CSS. And the JavaScript that drives this Datepicker control. So when we click on Download, I can come to our jQuery UI download screen. Now there's a lot of themes t you can pick, we're going to go with the default one. We're going to go with the latest version, and you'll see in addition to the date picker, there's a lot of other items you can add, accordions, buttons, custom pop-up dialogues, sliders, tabs, a lot of good stuff. So I'm just going to download all of it into the script, but we could just select Date Picker if we want. All right, so let me go ahead and open that. It's going to give us a zip file. And there's two things I need to grab outta here. I need to grab this folder in the CSS called UILightness. We're going to copy that. I'm going to run off to the desktop where my website is. And, let me go ahead and add a new folder into here, and we'll call it Styles. Let's paste that into there. And the last thing I need to do is we need to come into the zip, come into the JavaScript area and grab this UI 18 custom, this contains all the controls that we're going to work with. And I'm going to put that in my scripts folder. Okay, so that's kind of how you get started going out to jQuery. com, grabbing the jQuery script and then grabbing the appropriate files for jQuery ui. So now let's hit Refresh back on our website. And you'll see that in our scripts we now have jQuery 161 which is the latest version, as well as our custom jQuery ui, I'm going to go ahead and delete the older versions that were automatically added for us with dynamic data. So go ahead and take those out. Okay, so the next thing I need is to put these scripts and the custom CSS that I have, here, into my website. Now I, I certainly could put it just in this file, but, I probably could use this across multiple pages potentially in my dynamic data website. So we're going to come in and just modify our master page. So let's drag in the CSS. And then let's also drag in our jQuery file first. And then we'll add in jQuery's UI. Now I'm going to adjust this just a little bit to tell the website to go up one level. There's a couple ways we could do this but this will work fine. And we'll go ahead and save it. So now, the scripts are going to be available. I also need to adjust that one to make it similar to what we have there. These scripts are going to be available to every, page in the website. So in the page templates, there's several for details and editing and inserting and all. All that good stuff. So since we put this in the master, it'll, it'll be available everywhere. So the next thing I need to do now is come back into our template, and when this loads, we would like to attach some client side JavaScript functionality, that will automatically add that date picker. So to do that, I'm going to come in, add a script block. And we'll do text equals JavaScript. And then in here, I need to know when this particular user control is loaded. I need to know on the client side in the browser when this text box is available. Well in jQuery, here is how you do that. You say I'd like to know when the document is ready. And when it is ready I'm going to call some code, a little function, like that. So, when the document's ready, invoke this anonymous function, and that anonymous function is now going to find this Text Box, and then attach the date picker functionality. All right so the first thing we need to do is find this Text Box. And the way we do that in jQuery is we do what's called an id selector. That pound or the hash means let's go find the ID called text box 1. And then I'd like to add date picker functionality. Now that date picker function is in our jQuery UI script that we downloaded. Okay, so now we're kind of ready to go. So let's run it and see what happens. You'll see it's not going to work quite yet. There's another little trick I need to show you. So we'll come in to SalesOrderHeaders, we'll hit Edit and you'll see that doesn't work. And we're actually going to have a script there that's going on, I'll show you why. If we go to View Source here, let's go search for, say, ship date. All right, you'll notice that we have right here an input, type equals text. Here's the value that was written out. And then you'll notice that the id is not Textbox1. And we are writing JavaScript, after all, so we need the client side ID, not the server side ID. So, I could come in and cut and paste that id right there. In here, but that would only work then for that one, so that's not what we want, because that would only work for the one ship date. What I need to do is find the client id that's generated, and a way we can this is let ASP. net figure it out, so I'm going to come in. And do a little server-side tag to say, ASP. net, please go find TextBox1 and write out the client id value into this script area. So now, this pound will say to jQuery, go find the id, the client side id. This percent equals tells ASP. net please write out text box one client id which will be that long string it just solved. But now it'll work with any of the text boxes. Okay, so now that we have the date picker hooked up. We have jQuery go find the text box, now it should find the appropriate one. Let's see what happens, so let's run it one more time. Go to Edit, and now you'll see we get a nice little datepicker, so I can come in and I could pick some different dates in here, if we'd like. Works out very, very nicely. The user can switch months. Makes it very easy to work with. And definitely something most users are accustomed to. Now, the only problem I have with this still is when it loads, you'll notice that although the date picker's nice, we really don't want to show the time here. To fix that, we can come back into our user control, and you'll notice that we have the field value string. What I'm going to do is add a little bit of code in here to say if it's not an empty string, let's go ahead and format it as a specific type of date time object, but only show the date. So we can do that by coming in and we'll do a little if statement here. So I'll say if the field value string does not equal empty strings, then, we want to go head and wrap a string. format call around this, and we're going to do a special format code. So, we're going to take the value, that's going to represent the zero, once it runs at run time, and we're going to convert that to month, day, year. And then we're going to pass in Conver. ToDateTime our fieldValueEditString. So we'll go ahead and close that. And then we'll close the string. format that we had. Okay? Otherwise or else, we're just going to put empty strings into the text box. So again as a review, it's a little bit of a long string but we are going to say if the fieldValueEditString that dynamic data passes us isn't empty, then, that would be the then, we're going to go ahead and say let's format that value as a month, day, year format code. That'll give us a date time object to use the format code on. And then we can say, otherwise empty string. So there's multiple ways we could do this part right here. We could even do daytime. parse if we wanted, and it'd actually be even a little less code. But this will work. So let's go ahead and run it. We'll go to SalesOrderHeader. Go to Edit. And that looks much better, we don't have the time on there, we still have the date picker functionality though. Makes it very easy for the user now to come in, and select what they'd like and it's a lot cleaner I think approach. And one of the nice things I didn't mention earlier is that if I take this date out, I'm going to try to type some alphabetic characters. You'll notice it doesn't allow those. So it only allows them, if they are going to type, to enter specific dates, and you can even tie in some validation to that. So that's an example of one of the different things we can do, as far as modifications and customizations go with an ASP. net Dynamic Data Site. Makes it very easy to go in to these different fields and customize things and work with those. Now there's a lot of other things we could do. But this'll wrap up the module, and give you a good jump start on getting started with dynamic data.

Summary

In this module, you've seen how we can work with the dynamic data feature of ASP. net, and how much productivity can really offer. Whether you're just making simple admin screens for a database, or whether you're building a fully data driven web application. So we start off by talking about the different templates that are available, and you can either choose Website or Web application templates. And that will automatically generate the Website or Web application project for you, and get everything going out of the box. The only thing you have to do is choose your data model. We talked about, there's two types you can choose from out-of-the-box. We can choose LINQ to SQL, or we can choose Entity Framework. Once you've picked your model and built it, we need to go into Global ASAX and we register that model. And then we can simply run the application, and out-of-the-box will give insert, update, and delete functionality, and even some filtering capabilities provided for us. We ended talking about how we can go in and customize dynamic data websites, whether it's customizing master pages to actually fields. And I showed a demonstration of how we can go in and customize date, time type of fields to add a jQuery date picker. So, this is a really great technology that you can use to build a variety of different data driven applications, and it's something that I know for me personally, I use a lot with clients that need admin screens for their database tables. So it's a great technology, something that's very customizable, and something that you can use right out of the box.

Web Deployment

Introduction

Hi, this is Fritz Onion with Pluralsight. In this module, we're going to be looking at the deployment of our ASP. NET for web applications. Until now, we focused on working within the development environment and testing against the development server. The final step is, of course, to push your application out to a live server and make it accessible to the world. We're going to start with a look at what is actually involved with deploying your application and we're going to do it by hand to give you a feel for what really has to happen as you push an application from your development machine or perhaps a staging machine to a live server. And then we'll take a look at tools to help you with this process. The tool of preference today is something called Web Deploy. It's built on top of the MSBuild tool and it takes into account all of the things you need to think about as you're deploying your application and it manages most of those details for you. So it's a real welcome addition to the deployment model. We'll also take a look at configuration files. This is one of the things that you have to think about when you deploy your applications. You very often change the compilation mode or the database connection strings that you have stored in your configuration file. And we'll take a look at a technology called XDT, which lets you transform a configuration file as you are doing the deployment of your application.

Installing IIS

So, the first step in deploying your application is, of course, to prepare the web server that you plan to deploy to. Now, the web server you may be deploying to could be already set up. It could be an internet service provider. But it's also useful for you to understand what is involved with setting up the server so you can test it locally before you actually try and push it out to a live server. Until now, we've been using either IIS Express or the Visual Studio Development Server and that's just a little application that runs locally, and it runs under your identity. And you'll often find that some things that you do in the development server or IIS Express will not propagate correctly to IIS because of security issues. The IIS identity is always going to run under a distinct identity that is probably not your own, so you may run into a couple of surprises when you first try to deploying to IIS. So, before you even consider deploying to a live site, I'd highly recommend that you install IIS on your local machine and try deploying your application just to your local IIS instance to make sure that you can flush out any details that should be taken care of before you push it to a live site. Now, IIS is available in most versions of Windows. It's off by default. So, if you want to test locally on your development machine, you will have to go in and enable IIS in the Windows Features panel, and make sure you turn on ASP. NET as well. So, we'll take a look at doing that next.

Configuring IIS locally

So here we are in a virtual PC with a fresh installation of Visual Studio Web Express. And what I'm going to do is enable IIS so we can test locally in a deployment scenario. Now, you'll find the installation for IIS is just one of the Window's features available under Programs. So if you go into Turn Windows features on and off in your Control Panel, you should see a check box called Internet Information Services. And the important thing you want to check here, you want to verify, is that under Worldwide Web Services, ASP. NET or Application Development Features ASP. NET has been checked. If you do that, it will bring in a bunch of dependencies necessary for that to work properly and you should be good to go. So I'm already configured here and we should be ready to test. One thing I do want to point out though, is that if you enable IIS after you've installed ASP. NET, which you probably installed when you installed Visual Studio Web Express, you want to make sure that ASP. NET is properly integrated with IIS. And to do that, bring up a command shell with administrative privileges. I'm going to use PowerShell just because it's my favorite command shell environment. And navigate to the Windows directory and then navigate into Microsoft. NET Framework Version 4. And in here, if you look for everything that begins with asp and ends with exe, you'll find the utility called ASP. NET, or aspnet\_regiis, and that's the utility that will install IIS, install ASP. NET into IIS and make sure that all of the modules and handlers are properly configured. So let's go aspnet\_regiis. exe and do a -i to install, like so. And that will just, like I said, make sure that ASP. NET is properly configured in your IIS installation. Okay, at this point, we should have IIS ready to go. And you can either find it just by typing internet in your search programs or it's also under the Administrative Tools section. And if we bring that up, you knows, we can drill into Application pools and Sites and some other things. So, we'll take a look at that in more detail in a minute, but I just wanted to show you how to get IIS configured on your local machine so you can prepare for deployment testing.

Demo: deploying by hand

Now that we have IIS properly configured in our machine, let's go ahead and try deploying a web application to our local IIS server. Now, I have one, I have a little application that I've put together here. It looks very much like one we saw earlier in the course using the membership features of ASP. NET to add login capabilities. So, let me just go ahead and run this. And we just have some content on the front page here. We have an About page and we have the ability to log in. And I've created a user account called Bob with a password of p@ssw0rd if you're following along. And notice now that we are logged in as Bob, so what I'd like to do is try to get this deployed to our local IIS server and see if we can access it through there. So currently, we're going through the ASP. NET development server. Let's switch that out. So, let's go take a look at what IIS is doing. So the root directory where IIS is installed is under inetpub > wwwroot and in fact, if you go to a browser and just browse to localhost after you install IIS, you should see the IIS 7 banner that is dropped in there by default for you. So, what I'm going to do is just add an application below the root URL for the IIS server. And to do that, I'm just going to create a directory within here and populate it with the contents of my application. So here's my application. This is what Visual Studio created for us and let's just go ahead and try the really simple way of copying it over. So I'm going to hold the Control key down and drag and drop it into here. And there we go. Now we have a test app sitting up in the root, ready to go. So, let's go try configuring this with IIS now. So if we bring up IIS and navigate into the default website, there's our directory, but what we need to do now is actually create an application on top of that directory so that we can surface it through the URL. So I'm going to add an application, call it TestApp. Notice we have an option to select an application pool, which we'll talk about in a minute. And the physical path is just going to be that one we dropped it into. So that was inetpub > wwwroot > TestApp. Now, you could place this in another location on your disk, you're not required to put it under wwwroot. However, the security settings are already configured to grant read access and such to IIS under this directory. So, if you do it outside of this directory, you may have to tweak the security settings in the directory structure. All right, and let's go ahead and try browsing through it now that we have it configured. Let's just verify that we got the right directory. Looks good. Browse to the site. And that will cause ASP. NET to compile it for the first time. And notice now, the URL is pointing to localhost/TestApp and if this machine were exposed on the Internet, this would be going over port 80 and this would be my machine name or IP address, and that would be available to external clients as well. Let's try logging in as Bob like we did before. Okay, so we're running into our first problem with the deploying to IIS and in this case, it's having trouble with that local database, the one that we auto-generated inside of the app data directory, which is the default model used by the Membership API in ASP. NET. So next, we're going to take a look at things you often have to do after you deploy to a local IIS installation.

Understanding AppPools

Now, the reason we encountered an error when we tried to access the log in database with our newly deployed site to IIS is that we're no longer running under our identity. And the way you can find out what identity your application is running under is by finding out which application pool it's running within. And you can find that just going to your application within IIS, looking at the basic settings, and it will describe right here what application pool it's running in. And application pools are a mapping onto processes, worker processes within IIS, that determine the identity and security rights of a given application. Here are all the different app pools that are available. Now, in our case, the issue was the user profile was not loaded for the account and that's something that SQL Server Express depends on for locating its database instances. So, if we go and look at the identity, the application pool we are running under, DefaultAppPool, and go to Basic Settings, you'll see the name and the framework version. But if you go to Advanced Settings, we'll find a setting in here called Load User Profile, which defaults to False. So we can fix this particular error in our scenario by switching the Load User Profile to True and that should make SQL Express happy. Now I would like to point out here that if you're deploying to a live server, it's very likely that you'll have a full version of SQL Server available to you there, if it's a real web server. And this would not be an issue in that scenario. But if you do happen to be trying to deploy locally for testing within IIS, this is a nice little trick to remember that you need to switch the identity or change this attribute in the application pool. One other important thing to note is that the identity under which this process runs, notice it's set to ApplicationPoolIdentity. That's an identity that is uniquely generated for each application pool and the way you refer to it is with IIS AppPool and then the name of the app pool. So DefaultAppPool is what this one would be. So, that's the security identity that you could use to grant access or grant privileges to the process that was hosting your application. Okay, so let's just try going back and launching our site again and let's verify that we can log in and we have access to the database now. So Bob, with our password, and now Bob is successfully logged in. So, that was a deployment to the local IIS installation and all we did to deploy was copy the files, right, and add it simplest. That's really what it's about. We did encounter a minor issue, which was, how is the SQL Server going to be configured on the deployed machine? And that's something we do need to think about. Two things that come up very often are, you know, the database deployment, the configuration file deployment, as well as security issues with access to files in the machine. So, those three things are issues that often crop up in deployment that you need to be ready to deal with.

Introducing WebDeploy

As you have probably already guessed, there are better ways to deploy your web application than the technique I just showed you, which was manually setting up the IIS application and then physically coping the files to the target directory. You want to prefer to use a tool like this because it will handle all of the deployment issues for you. It will choose only the set of files necessary to deploy the application. We actually deployed everything and that was overkill, right? We didn't, we shouldn't have to bother deploying the source code files, any extra configuration files that aren't used at run time, etc. And so the model here is use this tool, Web Deploy. That will take care of packaging up your components. It will also take care of. Modifying IIS, if necessary, to change settings within the IIS deployment. You can specify the name of the application you want to create. It also supports the installation of the SQL database, that we'll take a look at later. And there's a feature within IIS that let you actually import. Or you can simply deploy to the local file system or you could push it out to an ISP. Any number of options are available. So, the preferred deployment model is to use the Web Deploy Tool within Visual Studio.

Demo: using WebDeploy

So, let's take a look at how to use Web Deploy to push our application to an IIS installation. If you go to the Properties of your web project, you'll notice a tab over here on the left called Package/Publish Web, and this is going to let you configure how the Web Deploy utility packages up your application for deployment. So a couple of options just to point out here, which files do you want to include as part of the deployment, typically leaving it to the defaulted only files needed to run this application is appropriate. And that will not include things like source code files which we kind of blindly included before. Whether you want it to include the App\_Data folder directory, our folder. And this depends on whether you're actually going to be using SQL Express as we have defaulted here. Or, if you want to take the data from your application, from your database and install it in a local SQL Server on the target installation. For now let's go ahead and include it. But we'll come back and revisit that when we deploy to an Internet service provider. There's also an option for including databases that are configured in the Package/Publish SQL tab. We'll come back to that in a minute as well. You can ask it to propagate the settings, you have defined in IIS Express. So if you've configured some settings locally that you're doing testing against and you want those propagated, it will install them in the, in the IIS, version that you deploy to. And then the output for this whole process is a, is going to be a zip file that contains the package of your application, all the contents zipped up ready to go. And you can then use an import facility within IIS to actually deploy it. So let's go ahead and save that, and if we just build our application. So if you look under the object directory, package sub-directory, this is where it's expanding all the files in preparation for deployment, and will have generated a zip file for you to import if you want to. However, the easiest way to use this MSDeploy feature is to use this Publish tab up here. And what the Publish toolbar lets you do, is configure one or more deployment scenarios, so it's really useful to configure MSDeploy the way you need to configure it. And then set up one or more deployment targets so that you can test in those targets. And this might be where you configure to a local machine for testing, and then maybe you have another option here to, to deploy to a live server once you've gone through your tests. So if you look at the editing, the properties on this local publication and I, I just set this up as a local deploy so I could try out the deployment model on my machine here. You specify the Publish method, which can either be Web Deploy, FTP, File System, or Front Page Server extensions. Generally you want to prefer Web Deploy where possible, unless the target doesn't support it. We're going to map onto local hosts. And we're going to specify what application we want to be set up in IIS. Now what I've done is gone through and actually removed the application we created by hand earlier from IIS, and I've gotten rid of the directory that was there. So we're starting from scratch again and we're going to use this new or this Web Deploy feature to do that. So, we request the settings to to deploy this as a IIS application and not to delete all the files, when we push it there. So, let's go ahead and try publishing the site and see what happens. The first thing you'll notice is that you do need to be running under administrative mode to perform this deployment because we are pushing to IIS. So, let me go ahead and close this and re-open Visual Studio with administrator privileges which you can do with a short cut if you do a, a Ctrl+Shift+Click on the application bar that will launch with admin privileges. And let's go ahead and open up our application again. And try using our deploy which was local. We're going to go to local deploy. And let's publish. Okay, and you can see, this is going through the process of generating the installation package and then actually executing it. Again, this is something that's very easy to do from the command line or by hand, if you want to. There's a command file that is generated for you as a part of this process. But within Visual Studio this single-click Publish is, is very convenient for testing as well. So, let's go ahead and see what happened over here in IIS, let's do a Refresh. And notice it created a test application for us and if we look at the properties of that notice it dropped into Inetpub wwwroot TestApp that was the same location that I had tried it out earlier. And, all the contents should be there. Now, let's just go look at the directory structure because I want to point out a couple of things that were different from how we did it before. If you recall, before I manually created that application and then we just copied the entire directory over. You'll notice here, Web Deploy was much more selective in what it copied over. So there are no source code files here. There's just the root Web. config file the Bin directory contains our compiled assemblies. The App\_Data directory contains our database that we requested that it deploy and everything else is just the bare essentials right only what is strictly necessary to deploy your application. And if we had to do that by hand we'd have to very carefully select only the files necessary. So let's go ahead and trying browsing to the site. All right, so the, the application is deployed. Let's try logging in as Bob again. Verify that the database was installed properly as well in that local directory. And all looks good. Okay, so that is definitely the preferred deployment model even if you're doing local testing. That, that gives you a nice reproducible deployment process which you can then customize depending on the target that you're shooting for.

SQL Deploy and Web config transforms

I'd like to finish by showing you how to deploy your application complete with its database to an Internet service provider. Just to give you the complete picture in a very common scenario for application development within Visual Web Developer Express. So, there are two things that we need to take care of make sure we have support for before we try publishing to a service provider. One is the database that I was using was the default SQL Express database. And that works fine when I had control on the, of the machine and we were just pushing a physical file to the directory structure and I knew that SQL Express was installed. However, if you're deploying to a machine that has a different configuration like most Internet service providers will have their own SQL database installation that you'll have to inte, integrate into we'll need to figure out how to get our database schema, and perhaps data into the target database. So the good news is the Web Deploy packaging facility has a Publish SQL capability as well. And you can specify where it should pull the data from, typically your local database. Whether it should pull the schema of the database only or the database schema and the data and then you specify the target connection string. How should it actually take the data and publish it to the target server. So, we're going to take a look at doing that and then the one other piece we need to take care of is the configuration file. If we're deploying to a new server, that new server is going to have a different database location. And we're going to have to change our database connection string within the configuration file to match it. And there may be other things you want to change in your configuration file as well. So, the good news is, is there's a generic configuration file deployment feature that lets you customize web. config during deployment. And I'm going to walk you through this. There's a very sophisticated syntax called XDT, X, XML Data Transform, that lets you generate many different configurations of the file depending on your target scenario. I'm just going to show you a couple of simple ones that are the most common, but if you want to dig into this further there are many more features you can try out. So, let's go try configuring our database for deployment and our web configuration file to match the target server.

Demo: deploying to an ISP

Okay, so our final goal here is to publish this application to an Internet service provider. Now, your first step, of course, is to set up an account with an Internet service provider and get it ready to go. If you don't have one already there's actually a nice little directory you can browse. If you go to the Start Page in Visual Studio you'll see an entry for Finding Affordable Web Hosting. And there's a little gallery here, Web Hosting Gallery that you can navigate to and it will show you a list of service providers that you can use to deploy your applications to. Now all of these, huh, all of these service providers that are shown here, do support web deployment using the Web Deploy facility that we just looked at and that makes it really simple to push your applications up into an ISP. Now for demonstration purposes, I already have an account set up with a shared hosting provider called Cytanium, the one you see here. I'm not endorsing them necessarily, it's just one that I happened to try out, so please do your due diligence, and research the provider that makes sense for you appropriately. Once you've set up your account with your service provider you should receive an email with detailed instructions on how to use Web Deploy Microsoft Web Deploy. And from my experience so far they, they really walk you step by step to configure it. So, let me just show you that what we're going to do. First of all, in order in, in preparation for deploying to a service provider, I'm going to include the database. So that was the one step that we had overlooked before because we knew that it was just a local copy. So I want to use the SQL Package Publish feature to actually deploy the database that contains our log in information. And then I'm going to go to the Package Publish SQL tab. And you'll notice that it's got a database here that it inferred from our application. If you open up our configuration file, you'll notice we have an ApplicationServices connection string that's pointing to that SQL Express database that we're using by default. And that's where our user account information is stored. This is the default application services database that ASP. NET membership will store its data in. So all I want to do is take the data in that database and propagate it to my target server. And what I've done is down in Database Entry Details, I had to specify the connection string for the destination database. And this is where you need to find the information from your service provider. This particular service provider that I'm using gave me a single database instance. And so what I need to do is actually install my data into the database instance that was provided to me. So, here's the information that my provider handed to me, and I just pass in my credentials so that when I run this Web Deploy against the internet service providers machine, it will take the data from my database, and install it into this particular database instance. Now, if you have a higher level account, you actually might get a separate database application or instance for each of your applications in which case you would specify that locally here as well. Now, the other thing that we need to take care of and so finishing up the database here, we want to pull data and or schema from our database, and here's the connection string that I just copied from my configuration file so it knows where to pull the data from. And in my particular scenario, I want to go ahead and copy both Schema and Data because I pre-populated it with a couple of user accounts that I like to use for testing. You could just choose just could choose Schema if you wanted to start with a blank database. And then finally down here, we're just going to have it include the Auto-generated Schema and Data script that is created for us. Notice the ability to add additional scripts however. So if you have your own database configuration script that you want to use, that's very easy to integrate with here as well. Okay, now before this is going to work, before I'm going to be able to actually deploy this to the service provider, I need to make sure that the connection string is, is appropriate in the deployed server. Here's the connection string that I need to be installed in the server. And what I'm going to have to use is this configuration file transformation so that when it does deploy to the server, it generates this new database connection string instead of using the default one that's in here. So let's go look at how that works. If I open up the configuration file, web. config, here is my connection string that's already defined. And that's the one that's going to work locally on my local development machine. Now, what I've done is I've used the web config transformations here. Now, when you generate a new asp. net project, it will create these web. config files below your main web. config file that describe transformations that should occur when you publish in debug mode and when you publish in release mode. So, you can have one of these for each target compilation that you support. And you can add as many of those as you want to. And within here if you open up one of these, the debug for example. There will be a couple of sample transformations already to find. And the good news is that they included two of the most common ones you'll tend to use. The first of which is the connectionString. So this initially will commented out, but you'll see it has a connectionString ready for you to substitute. And the way it works is it, it will take the existing configuration file, and then it will apply this transformation to it. And the transformation is going to use this locator. It's going to find out, figure out which element it should match in your config file, in this case, it's going to match the name of the connectionString. And then, what we're going to tell it to do is to set the attributes of that element. And the attribute we want to change is the connectionString from the one that we have defaulted here to the target one that we need for deployment. So again a very common thing to do changed your connection string on deployment and it's already stubbed out for you. If you open this up, it's very obvious how to change it. The one other thing that you'll tend to want to change is when you publish and release mode, in addition to the connectionString, you're probably going to want to remove the debug attribute from the compilation element. And that's something they also have already in place for you. So if you don't really want to get into all the details of how this XDT syntax works, the two examples they provide to you are, the most common examples that you'll tend to want to use anyway so hopefully you'll find that a very easy transformation. Okay so I have my configuration file transformation ready to go. Let's go now configure a new one click publish. And we're going to target my hosting provider. Now I've already created this, but what you would do is create a new publishing profile and name it after your hosting provider. Specify web deploy as the publication. Specify the service URL, and this should be information they provide to you as part of your sign up to the service provider. Designate the site application. So, this is the URL that I have configured with the service provider Pluralsight-test. And I've specified an application of test app. I've asked it to mark it as an IS application like we did before. And I've said let's just leave the files in the server when you deploy, and then finally we specify our username and password to actually do the publication. So let's go ahead and try publishing to the service provider. Now this is actually reaching out to an endpoint your service provider has defined that is listening for this particular type of deployment. This ms deploy model. And, once it's deployed, I should be able to actually go to my public URL, which in this case is Pluralsight-test. com/testapp. And, if our web deployment worked properly, we'll see our site. Which we do. And let's check the database now, let's go ahead and try logging in, and our friend Bob was the one with the credentials here, so password, login. And we're good. So if you've set up your site with an MS deploy, deployment you can tap into any number of service providers that support that model deployment and as long as you think a little bit about your database deployment. How that's going to get propagated, and your configuration file, what changes need to happen when you deploy your configuration file. It's relatively straightforward to push your application up to the site. And then, the, of course, the advantage to that is when you want to do testing you can go back to your various single-click, Publish, configurations that you have. And you can deploy locally or you can deploy to your isp. And get things up and running.

Summary

And that concludes our look at web deployment in ASP. NET 4. Hopefully you saw how straightforward it was to set up a web deployment using MSDeploy. How you can configure it to deploy to a local testing environment, or perhaps to a local server, and also how you can configure it to point to an internet service provider. This new web deployment set up tools make it very straight forward to target all these different environments and tweak the things you need to tweak to make that deploy. And we saw how the xdt configuration file transformation made it possible to change the configuration file as we deployed to accommodate things like connectionString changes or debug changes and that's a very extensible system that you can use to configure whatever you'd like.