**Variáveis locais** são aquelas nas quais apenas a função onde ela está pode usá-la.

**Variáveis globais** são declaradas da mesma forma que uma variável local - primeiro o tipo, depois o nome da variável. Mas a diferença é que esta variável é declarada fora de qualquer função.

***Variáveis globais podem ser utilizadas por qualquer função***. E qualquer função pode alterar o valor, utilizá-la em um processo ou até mesmo atribuir o valor que quiser.

**XMLHttpRequest cria um objeto XMLHttpRequest**, logo ele é uma Classe. Porém JavaScript é Baseado em Objetos, por isso a todo momento é falado do OBJETO XMLHttpRequest.

No Ajax o Javascript faz uma solicitação ao servidores, no servidor nada muda com a utilização de Ajax, ele continua respondendo cada solicitação exatamente como fazia antigamente quando não se usava Ajax. Porém a resposta do servidor agora retornará apenas os dados que a página precisa sem qualquer marcação ou apresentação. Uma característica que será notada pelo usuário que está acessando a página que está utilizando Ajax é que grande parte da página não será alterada, mas sim apenas partes que necessitarão de atualização. Antigamente a página inteira era carregada, porém com Ajax esse paradigma muda.

Com isso, a transferência de informações entre o cliente e servidor diminui, consequentemente, a velocidade do site aumenta significativamente.

**Segue abaixo um exemplo de agrupamento de seletores que possuem as mesmas declarações:**

Se você tem elementos HTML com o mesmo estilo como no exemplo abaixo:

h1 {  
 text-align: center; color: red;  
}

h2 {  
 text-align: center; color: red;  
}

p {  
 text-align: center; color: red;  
}

Será melhor agrupar os seletores e minimizar a quantidade de código. Para agrupar os seletores, separe cada um com uma vírgula.  
No exemplo abaixo nós temos os seletores agrupados do código acima:

h1, h2, p {

text-align: center;

color: red;

}

Dados Escalares: estrutura formada por tipos de dados simples (tipos Escalares), sejam do mesmo tipo de dados.

A declaração **<! DOCTYPE>** deve ser a primeira coisa no seu documento HTML, antes da tag .

A declaração <! DOCTYPE> não é uma tag HTML; é uma instrução para o navegador da web sobre qual versão do HTML a página está escrita.

No HTML 4.01, a declaração <! DOCTYPE> refere-se a um DTD, porque o HTML 4.01 foi baseado no SGML. O DTD especifica as regras para a linguagem de marcação, para que os navegadores renderizem o conteúdo corretamente.

O HTML5 não é baseado em SGML e, portanto, não requer uma referência a um DTD.

**Dica**: Sempre adicione a declaração <! DOCTYPE> aos seus documentos HTML, para que o navegador saiba que tipo de documento esperar.

O texto alternativo pode ser apresentado de duas maneiras:

Dentro do elemento IMG através do atributo ALT;

No contexto de seu entorno na página.

Isso significa que o atributo ALT não é o único mecanismo que provê conteúdo e função a uma imagem. A informação pertinente a imagem pode ser fornecida por um texto adjacente. Em alguns casos, quando o conteúdo ou função da imagem não pode ser representado de forma sucinta pode ser fornecido um link ou o atributo LONGDESC, com referência a uma descrição mais detalhada.

A propriedade "**display**" é a mais importante no CSS para controlar layouts. Essa propriedade permite que você defina como determinado elemento HTML deve ser renderizado. "Display" pode ter vários valores possíveis, no entanto, os mais comuns são:

display:block ------------- Essa declaração faz com que o elemento HTML seja renderizado como bloco, tal como os parágrafos e os cabeçalhos o são. Um bloco contém um espaço em branco tanto em cima como embaixo e não permite outros elementos HTML ao lado, exceto quando tiver sido declarado ao contrário (por exemplo, declarar a propriedade float para o elemento próximo ao bloco).

display:inline ------------- A propriedade display: inline faz com que o elemento HTML seja renderizado inline, dentro do bloco na mesma linha. Quando o elemento encontra-se entre dois blocos ele forma o chamado 'bloco anônimo' e é renderizado com a menor largura possível.

Banco de Dados

**• Superchaves** – Associação de um ou mais atributos cujos valores, em conjunto, identificam univocamente cada tuplo.

**• Chave Candidata ou Alternativa ou Única** – Subconjunto dos atributos de uma superchave que, sendo ainda superchave, não pode ser reduzido sem perder essa qualidade.

**• Chave Primária ou Chave Principa**l – É selecionada de entre as chaves candidatas. Abriga valores que individualizam cada **tupla**. **Esse campo não pode repetir-se em uma mesma tabela** e **não pode ser nulo**.

**• Chave Secundárias** – São das demais chaves.

**• Chave Estrangeira ou Chave Importada**– Atributos que são chave primária em outro esquema relacional, permite a implementação de relacionamento no BD relacional.

O **HAVING**é diferente do **WHERE**. O **WHERE**restringe os resultados obtidos **sempre**após o uso da cláusula **FROM**, ao passo que a cláusula **HAVING**filtra o retorno do agrupamento.

DotNet – Entity Framework

Data Annotations – classes:

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

Table = [Table("Alunos")]

Chave primária = Se tiver especificado Id, o framework já subentende que é chave primaria. Caso contrario, poderá ser anotada com [Key].

Not Null = [Required]

**DbContext** = classes ds entidades do banco de dados (using System.Data.Entity;)

Exemplo:

public class EscolaContext:DbContext

{

public EscolaContext() : base("Escola")

{

Database.CreateIfNotExists();

}

public DbSet<Aluno> Alunos { get; set; }

}

Arquivo Web.config = guarda a string de conexão. Exemplo:

<connectionStrings>

<add name="Escola"

connectionString="Data Source=Localhost\SQLExpress; Initial Catalog=Escola; Integrated Security=True;"

providerName="System.Data.SqlClient" />

</connectionStrings>

Em uma **aplicação ASP.net**, um Especialista em Tecnologia da Informação criou um arquivo chamado header.cshtml com um conteúdo que deseja utilizar como cabeçalho de muitas páginas do *site*. Para importar o conteúdo deste arquivo em uma página que está criando, este Especialista deve utilizar a instrução ***@RenderPage("header.cshtml")***

O ADO.NET é um conjunto de classes que expõem serviços de acesso de dados para desenvolvedores do .NET Framework. ADO.NET fornece um rico conjunto de componentes para a criação de aplicativos distribuídos e compartilhamento de dados. É parte integrante do .NET Framework, fornecendo acesso a relacionais, XML e dados de aplicativo. ADO.NET oferece suporte a uma variedade de necessidades de desenvolvimento, incluindo a criação de clientes de front-end de banco de dados e objetos comerciais de camada intermediária usados por aplicativos, ferramentas, linguagens ou navegadores da Internet.

ou seja, trata se de um conjunto de classes de acesso, muito utilizado para a criação de aplicativos distribuídos e compartilhamento de dados

A arquitetura dos recursos do AJAX no ASP.NET consiste de duas partes: as bibliotecas de script de cliente e componentes do servidor.

Essas partes são integradas para fornecer uma estrutura de desenvolvimento robusta.

A **Common Intermediate Language** (ou **CIL**) é uma linguagem de programação de baixo nível do ambiente de programação da Microsoft. O código de mais alto nível do ambiente .NET Framework é compilado em código CIL, que é assemblado em código chamado bytecode. CIL é um código orientado a objeto e executado por uma máquina virtual. A CIL tinha inicialmente o nome de *Microsoft Intermediate Language* (ou MSIL), na época das versões beta da linguagem .NET. Depois da standarização do C Sharp e da CLI, o bytecode foi oficialmente referenciado sob a designação de CIL. Os utilizadores mais antigos da tecnologia continuam no entanto a utilizar o termo MSIL.

O ASP.NET Identity foi lançado junto com o ASP.NET MVC 5 e o Visual Studio 2013 e logo foi muito bem aceito pela comunidade técnica, pois possui uma arquitetura bem aberta, limpa e modularizada de forma que proporciona grande facilidade de customização e testabilidade.

Historicamente o ASP.NET forneceu 4 componentes de Membership

(2002 – 2005) – Não possuía nenhum componente

(2005 – 2010) – Membership Provider

(2010 – 2012) – Simple Membership

(2012 – 2013) – Universal Providers

(2013 – Hoje)  – ASP.NET Identity

Características do ASP.NET Identity desde a primeira versão

Parte do ONE ASP.NET

Customização do perfil do usuário simplificado (escrito em Code First)

Controle de persistência de dados (EF ou outros)

Totalmente testável (Unity Tests)

Role Provider (separação de acessos por perfil)

Claims Based

Autenticação com redes sociais (FB, Twitter, Google+ e Microsoft Accounts)

Integraçao com Active Directory (On-Premisses e Azure)

Integração com OWIN (OWIN Middleware based)

Entregue via NuGet (Nuget Everywhere)

Alguns aspectos chamaram muito a atenção, como por exemplo ser baseado em Claims (Claims Based), ter sido projetado como um OWIN Middleware e possuir a capacidade de integrar facilmente com as redes sociais através do OAuth 2.0 e OpenID

É possível também integrar com **Yahoo e LinkedIn**utilizando este pacote adicional desenvolvido pela comunidade técnica. Para customizar a integração com seu próprio mecanismo OAuth, siga este artigo.

A plataforma .Net possui um conceito semelhante ao conceito de pacote em Java. Este conceito se conhece como: **Namespace**

Na plataforma .NET, o componente responsável pela execução do código é chamado de ***Common Language Runtime –* CLR**

.NET (.NET*framework*) possui apenas dois componentes principais: *Common Language Runtime* (CLR) e a biblioteca de classes.

O **tempo de execução de linguagem comum**, em inglês **Common Language Runtime** (**CLR**), é o componente de máquina virtual da plataforma .NET da Microsoft que gerencia a execução de programas .NET. Um processo conhecido como compilação just-in-time converte o código compilado em linguagem de máquina que a CPU do computador executa.[1] A CLR fornece serviços adicionais, incluindo gerenciamento de memória, segurança de tipagem, tratamento de exceção, garbage collection, segurança e gerenciamento de thread. Todos os programas escritos para o framework .NET, independentemente da linguagem de programação, são executados pela CLR. Todas as versões do framework .NET incluem a CLR.

Quais são os três modelos de desenvolvimento diferentes suportados por ASP.Net: **Web Pages, MVC e Web Forms.**

A classe **BaseValidator** fornece a implementação principal para todos os controles de validação. Controles de validação são usados para validar a entrada do usuário em um controle de entrada associado.

O .NET *Framework*é um ambiente de execução gerenciado que consiste de dois componentes principais: o ***Common Language Runtime*(CLR) e a .NET *Framework Class Library*.** Sobre o .NET*Framework*, analise:   
  
I. Em muitas linguagens de programação, os programadores são responsáveis por alocar e liberar memória e por manipular o tempo de vida do objeto. Em aplicativos do .NET *Framework*, o CLR fornece esses serviços.   
  
II. Em muitas linguagens de programação tradicionais, os tipos básicos são definidos pelo compilador, o que complica a interoperabilidade entre linguagens. No .NET *Framework*, os tipos básicos são definidos pelo .NET *Framework Type System*e são comuns a todas as linguagens que o utilizam.   
  
III. O .NET *Framework* inclui bibliotecas para áreas específicas de desenvolvimento de aplicativos, como o ASP.NET para aplicativos da web, o ADO.NET para acesso a dados e o *Windows Communication Foundation*para aplicativos orientados a serviços.   
  
IV. Compiladores de linguagens direcionadas ao .NET *Framework* geram um código intermediário chamado de *Common Intermediate Language*(CIL), que, por sua vez, é compilado em tempo de execução pelo CLR. Com esse recurso, as ro- tinas escritas em uma linguagem tornam-se acessíveis a outras linguagens da plataforma .NET.

Olá pessoal, neste artigo veremos os conceitos e exemplos das palavras-chave **Virtual** e **Override** e veremos também sobre o modificador**Sealed**. Acompanhem:

**Virtual e Override –**Com o uso da palavra-chave **virtual** (destinada a métodos, propriedades, indexadores e eventos), determinamos que um membro pode ser sobrescrito em uma classe filha. Por sua vez, usando a palavra-chave **override** determinamos que na classe derivada, um membro virtual da classe base pode ser sobrescrito.

 Devemos ter em mente que ambas as palavras-chave completam uma à outra. Importante salientar também que a propagação da palavra-chave **virtual** ocorre para descendentes. Um método virtual pode ser sobrescrito em descendentes e, ainda, em uma **classe derivada**.

 Veja um exemplo com o uso das palavras-chave:

public class ClasseBase

        {

            //Declarando um método que pode ser sobrescrito

            public virtual void ExibeTexto()

            {

                string texto = "Texto da classe Base";

                Console.WriteLine(texto);

            }

        }

public class Derivada : ClasseBase

        {

            //Sobrescrevendo o método da classe Base

            public override void ExibeTexto()

            {

                base.ExibeTexto();

                Console.WriteLine("\nTexto da classe Derivada");

            }

        }

 E no método **Main**:

DadosPessoa.Derivada objDerivada = new DadosPessoa.Derivada();

            objDerivada.ExibeTexto();

            Console.ReadKey();

Obs: o uso do **virtual** em um método indica que ele pode ser sobreposto em uma classe derivada, opcionalmente. Já se declararmos um método como **abstract** deve, obrigatoriamente, ser sobreposto.

**Modificador Sealed –** O modificador **sealed**(também conhecido como **selado**) define que uma classe não pode ser uma classe base, ou seja, não pode ter herdeiras. Uma classe **sealed** está pronta para uso e deve ser empregada no programa. Devido especialmente às suas características, as classes **sealed** podem ser consideradas o **oposto** das classes abstratas.

 Além das explicações mencionadas referentes às classes **sealed**, devemos considerar alguns outros, conforme podemos ver abaixo:

 - Os membros **sealed** não podem ser sobrescritos, mas devem sobrescrever um membro **virtual** ou **virtual deduzido**(por exemplo, um membro abstrato);

 - O modificador **sealed** deve ser combinado com a palavra-chave **override**. Apesar de não poder ser aplicado a membros estáticos, ele pode ser aplicado a **propriedades**, **métodos**, **indexadores** e **eventos**;

 - A Common Language Runtime (**CLR**) está apta a realizar otimizações nos membros **sealed**.

 Agora veja o exemplo a seguir, que mostra o uso do modificador **sealed**:

public class ClasseBase

        {

            public virtual void Pessoa()

            {

                string mensagem = "Mensagem padrão da classe Base";

                Console.WriteLine(mensagem);

            }

        }

        //Classe extensível

        public sealed class ClasseDerivada : ClasseBase

        {

            //Sobrescrita do método

            public override void Pessoa()

            {

                Console.WriteLine("Sobrescrita do método!");

            }

        }

 E no método **Main**:

            ClasseDerivada objClasseDerivada = new ClasseDerivada();

            objClasseDerivada.Pessoa();

            Console.ReadKey();

Na arquitetura ADO.NET, a classe do namespace System.data que contém as funcionalidades necessárias para gerenciar dados armazenados na memória, através de um cache de dados desconectado, é chamada de: **DataSet.**

Responsabilidade da ViewModel: Disponibilizar para a View uma lógica de apresentação.

A arquitetura dos recursos do AJAX no ASP.NET consiste de duas partes: as bibliotecas de script de cliente e componentes do servidor.

Essas partes são integradas para fornecer uma estrutura de desenvolvimento robusta.

**O QUE É MODEL BINDING?**

Model Binding é o processo de criar objetos .Net a partir de dados enviados pelo browser. Mesmo sem saber, estamos usando Model Binding toda vez que implementamos actions que recebem parâmetros.

Toda vez que uma requisição do tipo “**/SeuController/SuaAction/121**” é recebida, o framework MVC precisa tratar essa requisição de forma que possa passar valores apropriados nos parâmetros das actions.

De forma sucinta, esse tratamento inicia-se com o componente **action invoker**, que, como o próprio nome diz, é o responsável por invocar action methods. Antes de chamar a action, o action invoker verifica cada parâmetro e encontra o **model binder** correspondente para cada tipo de parâmetro.

**What is Razor?**

Razor is not a programming language. It's a server side markup language.

Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages.

Server-based code can create dynamic web content on the fly, while a web page is written to the browser. When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser. By running on the server, the code can perform complex tasks, like accessing databases.

Razor is based on ASP.NET, and designed for creating web applications. It has the power of traditional ASP.NET markup, but it is easier to use, and easier to learn.

**Razor Syntax**

Razor uses a syntax very similar to PHP and Classic ASP.

A classe **BaseValidator** fornece a implementação principal para todos os controles de validação. Controles de validação são usados para validar a entrada do usuário em um controle de entrada associado.

As ASP (Active Server Pages - Páginas de Servidor Ativas) são um **ambiente** para **programação por scripts no servidor**, que você pode usar para criar páginas dinâmicas, interativas e de alta performance. Como as **páginas ASP, os scripts rodam no servidor** e não no cliente.

O Asp apresenta como vantagens:

**Independência do browser:** ASP pode rodar páginas complexas no servidor e enviar somente os resultados para o cliente.

·  **Páginas x Bancos de Dados:** Permite visualizar, atualizar e adicionar informações nos servidores SQL

·  **Segurança do código fonte:** Como o Servidor retorna somente o resultado html, o código fonte (lógica) fica preservada.

·  **Linguagens:** O ASP pode utilizar de comandos em VBScript, JavaScript e Html.

Neste caso ele garante o encapsulamento dos dados de conexão junto ao cliente por encapsular o resultado requisitado ao servidor em html. preservando - se desta forma o código fonte.