Aula05 – Forms Authentication

Disciplina: Programação Web
Prof. Allbert Velleniche de Aquino Almeida
E-mail: allbert.almeida@fatec.sp.gov.br
Site: http://www.allbert.com.br
/allbert.almeida

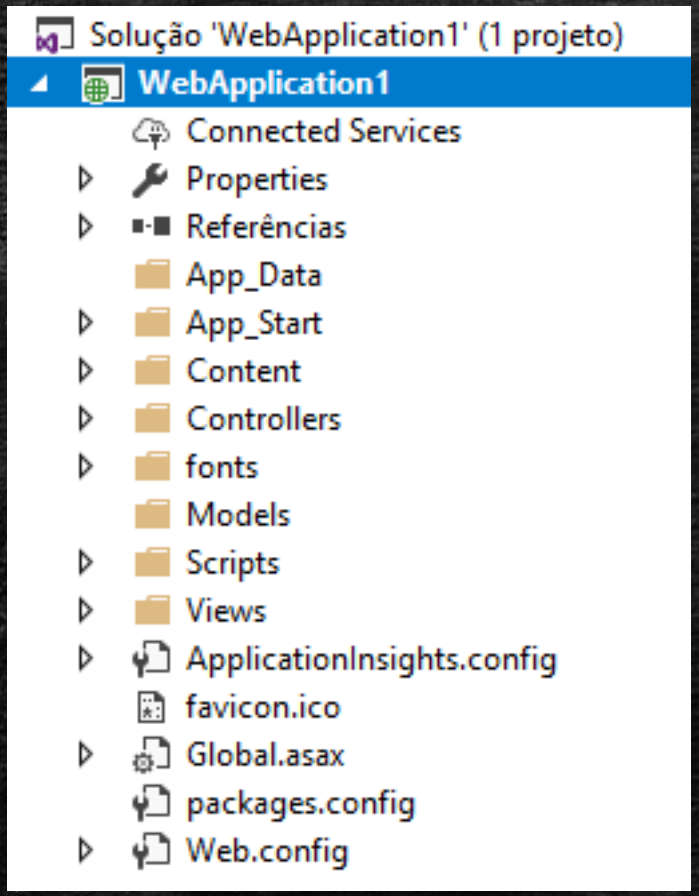
Objetivo

- O objetivo dessa aula é construir uma aplicação autenticando usuários com diferentes níveis de acesso usando Forms Authentication;

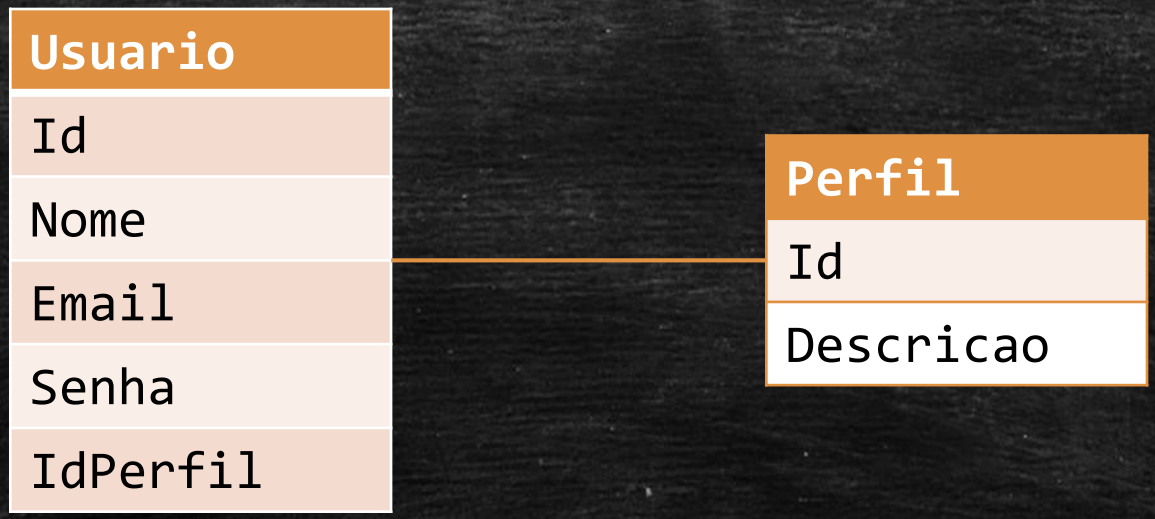
Criando uma nova aplicação

- Vamos abrir o Visual Studio;
- File >> New >> Project >> Aplicativo Web;
- Tipo de aplicação MVC;
- Change Authentication >> No Authentication

Exemplo de uma aplicação



Modelo



Criar uma classe Perfil

```
public class Perfil
{
    public int Id { get; set; }
    [Required]
    [DisplayName("Descrição")]
    [MaxLength(255)]
    public string Descricao { get; set; }
}
```

Criar uma classe Usuario

```
public class Usuario {
        [Key]
        public int Id { get; set; }
        [MaxLength(255)]
        public string Nome { get; set; }
        [MaxLength(100)]
        [EmailAddress]
        public string Email { get; set; }
        [Required]
        [DataType(DataType.Password)]
        public string Senha { get; set; }
        [Required]
        [DataType(DataType.Password)]
        [Compare("Senha")]
        public string ConfirmaSenha { get; set; }
        public int PerfilId { get; set; }
        public virtual Perfil Perfil { get; set; }
}
```

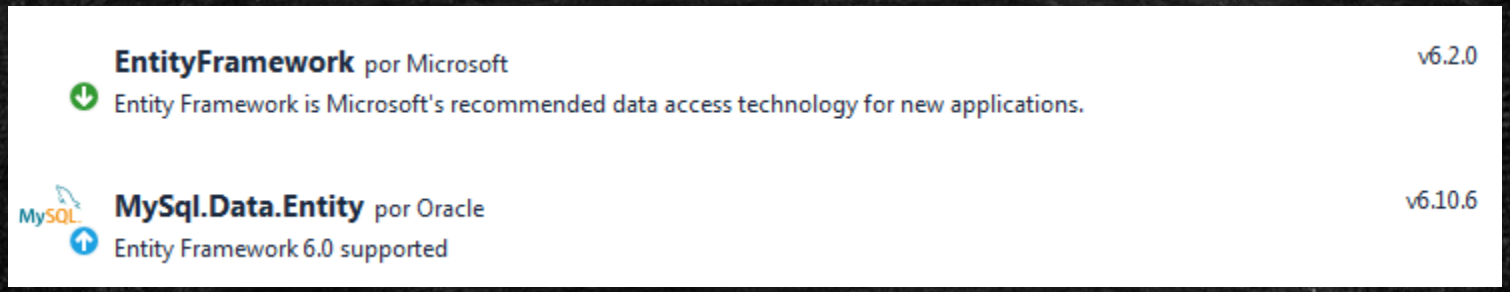
Criar uma classe Contexto

```
public class Contexto:DbContext
    {
        public Contexto() : base(nameOrConnectionString:
"StringConexao") { }

      public DbSet<Usuario> Usuario { get; set; }
      public DbSet<Perfil> Perfil { get; set; }
}
```

Adicionar 2 pacotes para solução

- EntityFramework
- MySql Data Entity 6.9.11



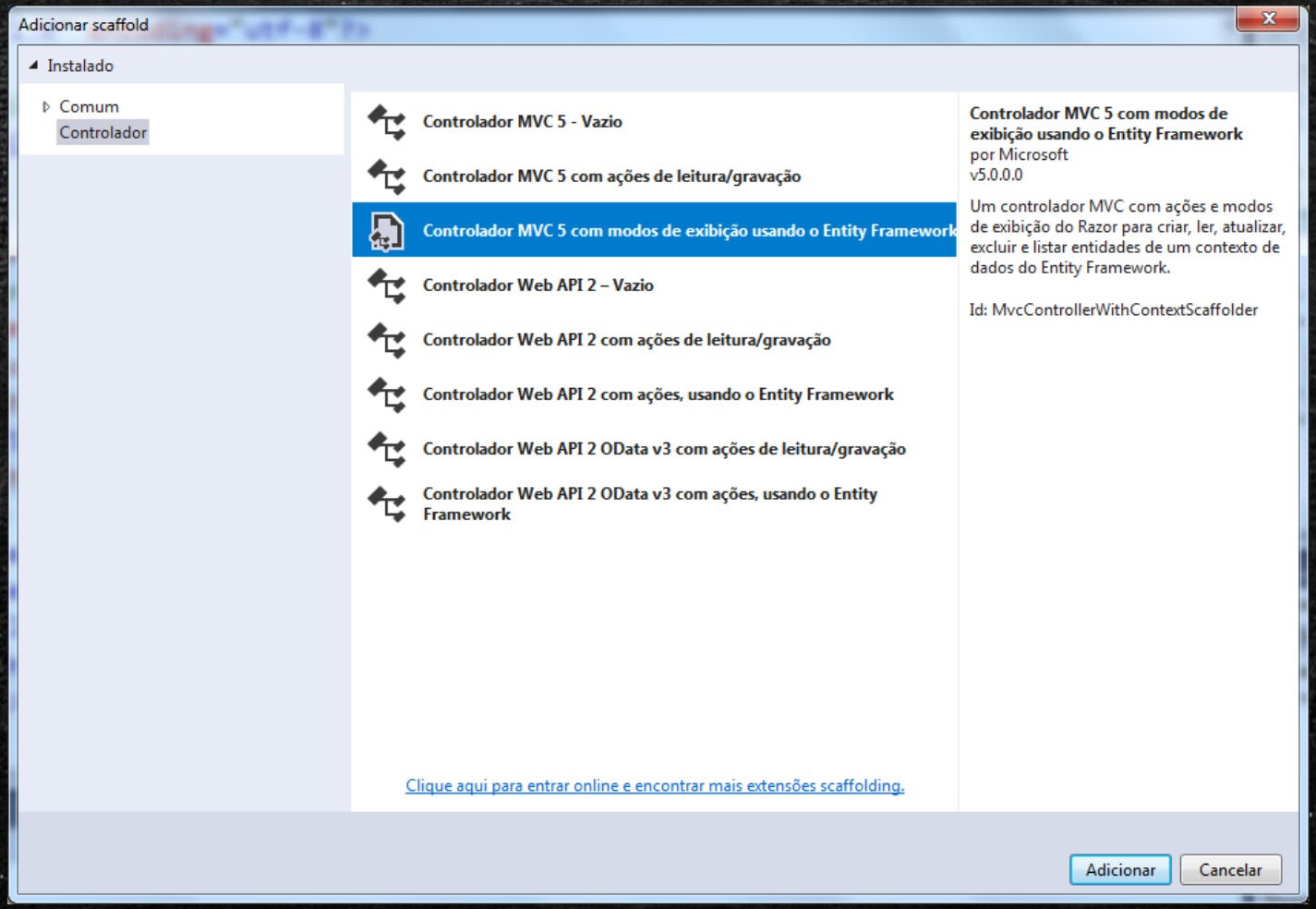
Adicionar a String de Conexão no WebConfig

```
<connectionStrings>
    <add name="StringConexao"
providerName="MySql.Data.MySqlClient"
connectionString="server=localhost;dat
abase=teste;uid=root;pwd='root'" />
  </connectionStrings>
```

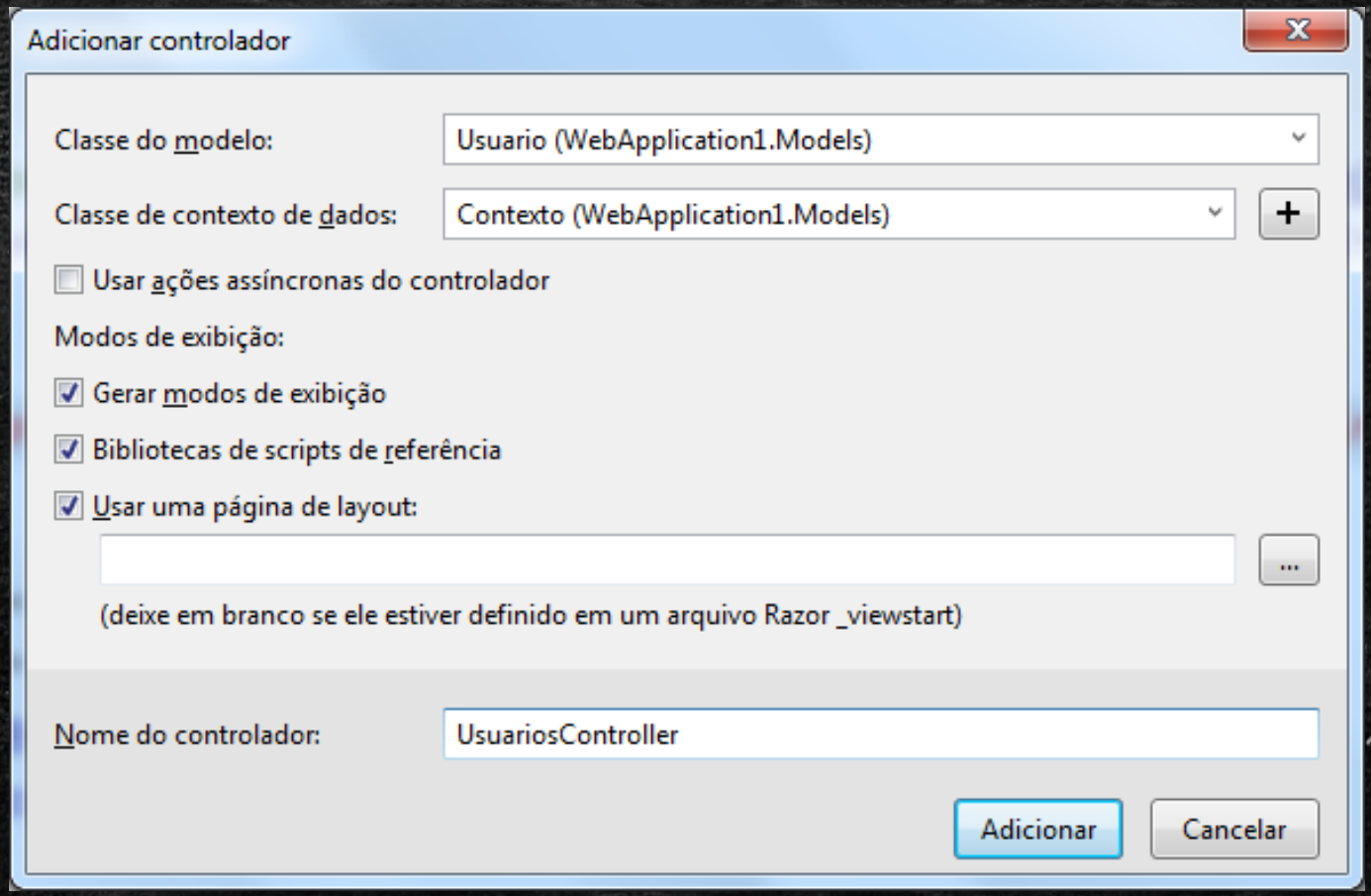
Compilar a solução

- Compilar a solução para que os modelos sejam validados e seja possível criar o "CRUD" da classe Usuario; Perfil; UsuarioPerfil

Adicionar Controller



Adicionar Controller



Executar a solução

- Na URL entre com o caminho da aplicação + /Perfils/
- Na URL entre com o caminho da aplicação + /Usuarios/
- Cadastrar perfil e usuário;

Forms Authentication

- Adicionar no system.web do web.config:

```
<authentication mode="Forms">
<forms loginUrl="~/Home/Login"
timeout="2880" />
</authentication>
```

Adicionar autorização

- Na PerfilController vamos adicionar autorização. Somente usuários autorizados terão acesso a esse controller;
- Importante a autorização pode ser definida no controller inteiro ou somente nas actions;

```
[Authorize]
```

HomeController

- Vamos adicionar na HomeController a action Login;

```
public ActionResult Login(string email, string senha, string ReturnUrl) {
       Usuarios usu = db.Usuarios.Where(t => t.Email == email && t.Senha == senha).ToList().FirstOrDefault();
       if usu != null) {
           FormsAuthentication.SetAuthCookie(usu.Nome, false);
           FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(1, usu.Email,  DateTime.Now,
DateTime.Now.AddMinutes(30), false, usu.Perfil.Descricao);
           string hash = FormsAuthentication.Encrypt(ticket);
           HttpCookie cookie = new HttpCookie(FormsAuthentication.FormsCookieName, hash);
           if (ticket.IsPersistent)
               cookie.Expires = ticket.Expiration;
           Response.Cookies.Add(cookie);
           if (String.IsNullOrEmpty(ReturnUrl))
              return RedirectToAction("Index", "Perfils");
           else  {
              var decodedUrl = Server.UrlDecode(ReturnUrl);
             if (Url.IsLocalUrl(decodedUrl))
                 return Redirect(decodedUrl);
             else
                 return RedirectToAction("Index", "Perfils");
             }
       }
}
```

View Login.cshtml

- Vamos adicionar a view login;

```
@Html.AntiForgeryToken()
<div class="form-group rx-form-group">
<div class="col-md-12 col-sm-12 form-group">
<i class="fa fa-envelope select"></i>
<input type="email" placeholder="E-mail" required="required" class="form-
control" id="email" value="" name="email"><br />
</div>
<div class="col-md-12 col-sm-12 form-group">
<i class="fa fa-unlock-alt"></i>
<input type="password" placeholder="Senha" required="required" class="form-
control" id="senha" value="" name="senha"><br />
</div>
</div>
<div class="col-md-12 col-xs-12 form-group">
<button class="btn btn-default form-control" type="submit">Acessar</button>
</div>
```

Compilar a solução novamente

- Compilar a solução novamente e verificar se conseguem acessar perfil;

Validar o Perfil

- Já vimos que somente usuários autorizados podem entrar na página, agora vamos ver como controlar o perfil;

Global.asax

- Vamos adicionar o método;

```
protected void Application_PostAuthenticateRequest(Object sender, EventArgs e)
{
var authCookie =
HttpContext.Current.Request.Cookies[FormsAuthentication.FormsCookieName];
if (authCookie != null) {
   FormsAuthenticationTicket authTicket =
FormsAuthentication.Decrypt(authCookie.Value);
   if (authTicket != null && !authTicket.Expired) {
      var roles = authTicket.UserData.Split(',');
      HttpContext.Current.User = new
System.Security.Principal.GenericPrincipal(new FormsIdentity(authTicket),
roles);
   }
}
}
```

Adicionar perfil na autorização

- Na PerfilController vamos adicionar autorização. Somente usuários autorizados com roles definido terão acesso a esse controller;
- Importante a autorização pode ser definida no controller inteiro ou somente nas actions;

```
[Authorize(Roles = "Admin")]
```

Compilar a solução novamente

- Compilar a solução novamente e verificar se conseguem acessar perfil;