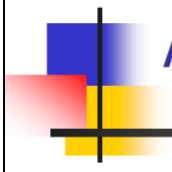


ASP.Net MVC

Sécurité

Principes, Fournisseurs, Validation par la décoration

What the f...un !

- Whaou ! Regardez comme c'est bien sécurisée !
  - La sécurité doit TOUJOURS être testée par le code

$£#%^!	Info
Usurpation (Spoofing)	Survient lorsque l'utilisateur se fait passer pour un autre. L'application doit vérifier l'identité de l'utilisateur à chaque fois qu'une opération sensible est tentée.
Falsification (Tampering)	Le phénomène apparaît lorsqu'un utilisateur arrive à altérer le contenu d'une page ou d'une base de données sans autorisation. le problème survient principalement de manière indirecte lorsque le serveur/le client exécute "par erreur" un script envoyé par un utilisateur. Il ne faut donc accorder aucune confiance aux données envoyées par l'utilisateur (ça ne vaut pas seulement pour la QueryString.)
Répudiation (Repudiation)	Le phénomène intervient lorsqu'une action malveillante ne laisse aucune trace, interdisant ainsi toute recherche du fautif. L'utilisation d'un journal (eventlog) par l'application est quasiment incontournable
Divulgation d'information (Information disclosure)	Le problème apparaît lorsqu'un utilisateur malveillant accède à des informations privées : des mots de passe, des fichiers ou des informations en base. Pour éviter cela, l'application ne doit pas stocker les mots de passe, stocker les informations sensibles sous forme cryptée (ou mieux : hachée+cryptée.)
Élévation de privilège (Elevation of privilege)	L'élévation de privilège survient lorsqu'un utilisateur obtient un niveau de confiance auquel il n'aurait jamais du accéder. Il est recommandé d'exécuter les pages ASP.Net avec l'identité ayant le minimum de privilèges pour effectuer leurs tâches normales. L'idéal, en intranet, est d'utiliser l'identité Windows de l'utilisateur.
Déni de service (Denial of service)	L'attaque tend à "écrouler" le serveur pour empêcher les utilisateurs "normaux" d'accéder au service. L'outil d'administration de IIS permet de refuser de traiter les requêtes provenant de certaines adresses IP et de limiter le nombre de clients servis en parallèle.

Principes

- Authentification
  - Permet au serveur IIS de connaître l'identité du client
  - Utilisation, en général, d'un couple (nom, mot_de_passe)
  - IIS peut également restreindre les accès en se basant sur l'adresse IP du client
  - L'identification peut être réalisée par toutes sortes d'autorités
    - À l'aide de la base des comptes du système
    - À l'aide d'une base de données dédiée
    - À l'aide de certificats
    - …

Principes

- Autorisation
  - L'authentification ne limite pas forcément les droits à Entrée autorisée / Entrée refusée
  - Un ensemble de droits différents peut être accordé en fonction de l'identité de l'utilisateur
  - Passez par des rôles vous simplifiera la vie :
    - Les rôles donnent des droits
    - On donne/supprime des rôles aux utilisateurs
    - On ajoute/supprime des droits à des rôles

Principes

- Personnification
  - Mécanisme permettant à IIS d'usurper l'identité d'un utilisateur
  - Le programme ASP.NET subit alors les droits d'accès définis au niveau des ressources système
    - Accès fichier
    - Quotas
    - Accès "Integrated security=true" de Sql Server
    - …

WebForms… Le bon vieux temps…

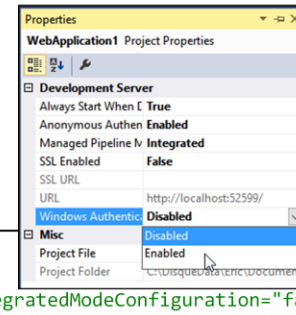
- C'est fini !
  - Contrôles serveur, ViewState haché et encrypté, <% @page validaterequest="true" %>, Event Validation : y a plus
- Quelques règles
  - Encoder les données "venant d'un utilisateur"
    - Champs, input hidden, QueryStrings, cookies navigateur, ajax, WS, …
  - Exiger l'accès authentifié doit être la règle, l'accès anonyme l'exception
  - Utiliser des cookies "HTTP-only"

```
<system.web> <httpCookies httpOnlyCookies="true"/>
```

  - Utiliser l'AntiXSS @Html.AntiForgeryToken()
  - Ne pas essayer de ruser face aux utilisateurs : vous avez perdu d'avance

Web.config

- Identification par windows
  - Idéale pour une application intranet
  - À activer sur le serveur Web




```
<system.web>
  <authentication mode="Windows" /> ← SSI activé sur IIS
  <identity impersonate="true" />   ← SSI mode != Pipeline intégré ou validateIntegratedModeConfiguration="false"
  […]
<system.web>
<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />      ← Ou pas mode "Pipeline intégré" mais "Classic"
</system.webServer>
```

⚠ Lancer IIS Express en tant qu'Administrateur !

- Identification applicative
  - Nécessaire pour une application internet

```
<system.web>
  <authentication mode="Forms" />
  […]
<system.web>
```

Restrictions d'accès

- `AuthenticateAttribute`
  - Décore actions et contrôleurs
  - Options
    - Users : noms d'utilisateurs séparés par des ,
    - Roles : noms de rôles/groupes séparés par des ,
  - Peut être appliqué au niveau global

/App_Start/ FilterConfig.cs

```
public static void RegisterGlobalFilters( GlobalFilterCollection filters ) {
  filters.Add( new AuthorizeAttribute() );
  filters.Add( new HandleErrorAttribute() );
}
```

- `AllowAnonymousAttribute`
  - Permet d'autoriser l'accès anonyme ponctuellement lorsque le contrôle d'identité est la règle du niveau supérieur

Accès aux identités

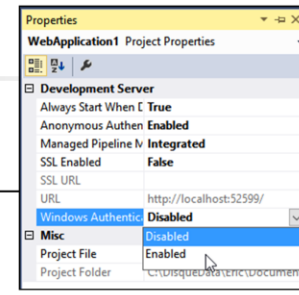
- Identité logique Asp.Net

Membre	Info
`bool Controller.User.IsInRole( string )`	Tests les rôles logiques comme AuthenticateAttribute
`string Controller.User.Identity.Name { get; }`	Nom sur lequel se base AuthenticateAttribute pour les vérifications
`string Controller.User.Identity.AuthenticationType { get; }`	Qui a authentifié (ex : "Negotiate" pour une identité vérifiée par le système)
`bool Controller.User.IsAuthenticated { get; }`	La requête a-t-elle été authentifiée ?

- Identité du thread courant

Membre	Info
`WindowsIdentity WindowsIdentity.GetCurrent()`	Identité Windows du thread courant
`string WindowsIdentity.Name { get; }`	Nom d'utilisateur qualifié
`IdentityReferenceCollection WindowsIdentity.Groups { get; }`	SID des groupes de l'utilisateur système

Identification intranet



- Exemple

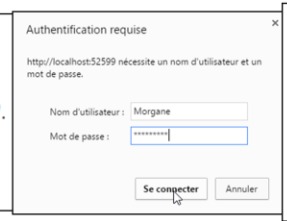
```
<system.web>
  <authentication mode="Windows" /> <identity impersonate="true" /> […]
<system.web>
<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />      ← Ou pas mode "Pipeline intégré" mais "Classic"
</system.webServer>
<connectionStrings>
  <add name="NorthwindEntities" connectionString="[…];integrated security=True;[…]"
       providerName="System.Data.EntityClient" />
</connectionStrings>
```

```
public class ExempleSecuriteController : Controller {
  public ActionResult Index() {
    ViewBag.Utilisateur = User.Identity.Name;
    ViewBag.IdentiteIIS = System.Security.Principal.WindowsIdentity.GetCurrent().Name;
    using( var dc = new NorthwindEntities() ) {
      return View( "Index", dc.Products.Where( p => p.CategoryID == 1 )
                               .Select( p => new { p.ProductID, p.ProductName } ).ToList()
                               .Select( p => new ProductPourListeViewModel() {
                                         ProductID = p.ProductID, ProductName = p.ProductName } ).ToList() );    }
  }
  [HttpPost, Authorize(Users= @"IronPC\Morgane" )]
  public ActionResult Supprimer( int productID ) {
    ViewBag.Message = $"On dit que {productID} est supprimé...";
    return Index();
  }
}
```

Identification intranet

- Exemple

```
@using WebApplication1.Models
@model List<ProductPourListeViewModel>
@{ViewBag.Title = "Du ménage dans les brevages";
  var lesBrevages = Model; string message = ViewBag.Message;
  string identiteIIS = ViewBag.IdentiteIIS;
  string utilisateurApplicatif = ViewBag.Utilisateur == string.Empty ? "un inconnu" : ViewBag.Utilisateur;
}
<h2>@ViewBag.Title</h2>
<h4>Vous êtes @(utilisateurApplicatif).</h4>
<h4>Votre code et votre SQL sont exécutés par "@(identiteIIS)".</h4>
<p>@message</p>
@foreach( var b in lesBrevages ) {
  using( Html.BeginForm( "Supprimer", "ExempleSecuriteSysteme" ) ) {
    <p>
      <input type="submit" value="Supprimer" /> @b.ProductName
      <input type="hidden" name="productID" value="@b.ProductID"/>
    </p>
  }
}
```



Identification applicative

- One ASP : `FormAuthenticationModule`
  - Petite configuration dans Web.config
  - L'application teste l'identité
  - Un cookie crypté et signé est fabriqué
  - Asp.net vérifie le cookie et renseigne `HttpContext.Current.User`
- En fonction du framework
  - WebForm : autorisations basées sur les URLS
  - MVC : autorisations basées sur les actions

Identification applicative

- Web.config, rubrique `configuration/system.web/authentication`

```
<forms name="name" loginUrl="~/Login.aspx" timeout="30" path="path" requireSSL="[true|false]"
  slidingExpiration="[true|false]">
  <credentials passwordFormat="Clear|MD5|SHA1">
    <user name="Toto" password="Toto aussi..."/>
  </credentials>
</forms>
```

- `System.Web.Security.FormsAuthentication`

Membre (de classe)	Détails
bool Authenticate(string name, string password)	Pour petits tests : lit les credentials du Web.config
string Encrypt(FormsAuthenticationTicket ticket)	Permet d'ajouter des informations complémentaires dans le cookie
FormsAuthenticationTicket Decrypt(string encryptedTicket)	Permet de récupérer les informations complémentaires
void RedirectFromLoginPage(string userName, bool createPersistentCookie)	Pas vraiment dans la philosophie MVC : envoie directement un entête de redirection vers la page précisée dans QueryString["ReturnUrl"] au navigateur
void SetAuthCookie(string userName, bool createPersistentCookie)	Utilise le nom de cookie précisé dans la configuration et utilisé implicitement par Asp.Net pour fixer le User
void SignOut()	Efface le cookie, pour la PROCHAINE requête

Identification applicative

- Identification minimum
  - Web.config

```
<authentication mode="Forms">
  <forms loginUrl="~/Securite/Index" />
</authentication>
```

  - Contrôleur de test

```
public class TestController : Controller {
  [Authorize()]
  public ActionResult Index() {
    return View();
  }
}
```

  - Vue de test

```
<h1>Hello @User.Identity.Name</h1>
```

Identification applicative

- Login minimum
  - Contrôleur de login

```
public class SecuriteController : Controller {
  public ActionResult Index( string returnUrl ) {          ← Sera reçu dans l'URL
    ViewBag.ReturnUrl = returnUrl;                         ← Pour que la vue puisse transmettre à l'action "Login"
    return View();
  }
  [HttpPost(), ActionName("Login")]
  public ActionResult Login( string nom, string returnUrl ) { ← Devra être transmis
    FormsAuthentication.SetAuthCookie( nom, false );
    return Redirect( returnUrl );
  }
}
```

  - Vue

```
@using( Html.BeginForm( "Login", "Securite" ) ) {
  <text>Nom : </text> @Html.TextBox( "nom" )
  @Html.Hidden( "ReturnUrl" )
  <input type="submit" value="Entrer" />
}
```

Membership

- API Asp.Net standard
  - Mise en œuvre par différents fournisseurs
    - SqlMembershipProvider, OracleMembershipProvider, LeVotreMembershipProvider
    - Idem : SqlRoleProvider, LeMienRoleProvider
  - Choix et configuration du provider dans Machine.config
    - C:\Windows\Microsoft.NET\Framework\version_du_framework\Config
    - **Adapter et/ou contredire dans le Web.config** de l'application web
  - Création simplifiée des tables et procédures stockées pour Sql Server : Aspnet_regSql.exe

```
aspnet_regsql -S localhost -d BdMembersip -A mr -E
```

    - -S Serveur, -d base de données, -A comme Add, m = membership (compte), r = rôles, -E pour utiliser l'identité système, -U et –P si non

Membership

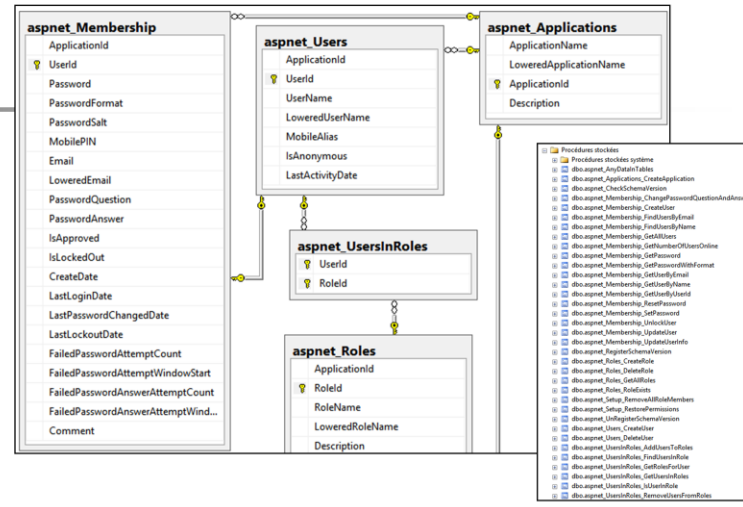
- Extraits du Machine.config (<system.web>)

```
<membership defaultProvider="AspNetSqlMembershipProvider">
  <providers>
    <add name="AspNetSqlMembershipProvider"
         type="System.Web.Security.SqlMembershipProvider, System.Web, Version=4.0.0.0, …"
         connectionStringName="LocalSqlServer"                ← Changer la base de données est souvent suffisant
         enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="true"
         applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed" maxInvalidPasswordAttempts="5"
         minRequiredPasswordLength="7" minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
         passwordStrengthRegularExpression=""/>
</membership>
```

```
<roleManager>
  <providers>
    <add name="AspNetSqlRoleProvider" connectionStringName="LocalSqlServer" applicationName="/"
         type="System.Web.Security.SqlRoleProvider, System.Web, . . . />
    <add name="AspNetWindowsTokenRoleProvider" applicationName="/"
         type="System.Web.Security.WindowsTokenRoleProvider, System.Web, . . ./>
  </providers>
</roleManager>
```

Membership

- Base de données standard



- API standard `System.Web.Security.Membership`

Membres (de classe)	Détails
`MembershipUser CreateUser( string username, string password, string email, string passwordQuestion, string passwordAnswer, bool isApproved, out MembershipCreateStatus status );`	Plusieurs surcharges
`bool DeleteUser( string username, bool deleteAllRelatedData );`	
`MembershipUserCollection FindUsersByEmail( string emailToMatch )`	
`MembershipUser GetUser( string username )`	
`void UpdateUser( MembershipUser user )`	
`bool ValidateUser( string username, string password )`	

Membership

- API standard `MembershipUser`

Membres (d'instance)	Détails
`DateTime CreationDate { get; }`	
`string Email { get; set; }`	
`bool IsApproved { get; set; }`	
`bool IsOnline { get; }`	
`DateTime LastLoginDate { get; set; }`	
`string UserName { get; }`	
`bool RequiresQuestionAndAnswer { get; }`	
`string GetPassword( string passwordAnswer )`	
`string GetPassword()`	
`string ResetPassword( string passwordAnswer )`	
`string ResetPassword()`	

Membership

- System.Web.Security.Roles

Membres (de classe)	Détails
`void AddUserToRole( string username, string roleName )`	
`void CreateRole( string roleName )`	
`void DeleteCookie()`	
`void CreateRole( string role )`	
`bool DeleteRole( string role, bool exceptionSiPasVide )`	
`string[] GetRolesForUser( string user )`	
`string[] GetRolesForUser()`	
`bool IsUserInRole( string user, string role )`	
`bool IsUserInRole( string role )`	

Gestionnaire de rôles maison - exemple

- Gestionnaire

```
public class RoleProviderMaison : RoleProvider {
  public const string Grouillot = "Grouillot";
  public const string Chef = "Chef";
  public override string[] GetAllRoles() { return new string[] { Grouillot, Chef };  }
  public override string[] GetRolesForUser( string username ) {
    if( username.ToUpper().Contains( Chef.ToUpper() ) )
      return new string[] { Chef };
    else
      return new string[] { Grouillot };
  }
  public override bool IsUserInRole( string username, string roleName ) {
    if( string.Compare( roleName, Chef, true ) == 0 )
      return username.ToUpper().Contains( Chef.ToUpper() );
    if( string.Compare( roleName, Grouillot, true ) == 0 ) {
      return true;
    return false;
  }
  public override string ApplicationName { get; set; }
  public override void AddUsersToRoles( string[] usernames, string[] roleNames ) {
    throw new NotImplementedException();
  }
  #region Pas mis en oeuvre non plus...
}
```

Gestionnaire de rôles maison - exemple

- Enregistrer le gestionnaire : Web.config

```
<roleManager enabled="true" cacheRolesInCookie="true" defaultProvider="RoleProviderMaison">
  <providers>
    <add name="RoleProviderMaison" type="SecuMaison.Classes.RoleProviderMaison" />
  </providers>
</roleManager>
```

- Conserver le contrôleur de login
- Utiliser dans un contrôleur sécurisé

```
public class TestController : Controller {
  [Authorize(Roles = SecuMaison.Classes.RoleProviderMaison.Chef)]
  public ActionResult Index() {
    return View();
  }
}
```

Hello Super-méga-chef

Vous êtes un chef : True

Vous êtes un Grouillot : False

- Vue

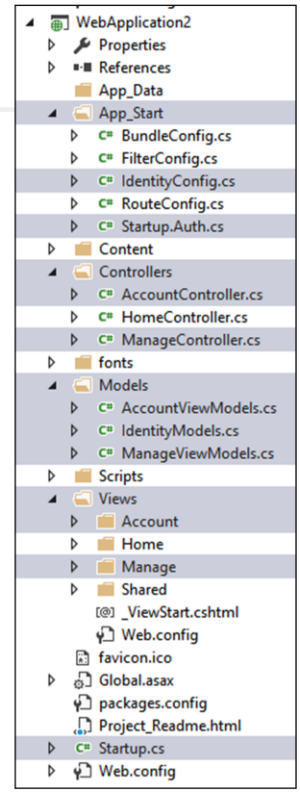
```
<h1>Hello @User.Identity.Name</h1>
<h2>Vous êtes un chef : @User.IsInRole( SecuMaison.Classes.RoleProviderMaison.Chef ) </h2>
<h2>Vous êtes un Grouillot : @User.IsInRole( SecuMaison.Classes.RoleProviderMaison.Grouillot ) </h2>
```

ASP.NET Identity

- ASP.NET Identity : modifier le template standard
  - Si vous êtes partis d'une application vide
    - Packages NuGet à référencer
      - Microsoft.AspNet.Identity.EntityFrameWork + .fr
      - Microsoft.AspNet.Identity.Core.fr
      - Microsoft.AspNet.Identity.Owin + fr
      - Microsoft.Owin.Host.SystemWeb
    - + Si vous voulez utiliser des identités externes
      - Microsoft.Owin.Security.Facebook
      - Microsoft.Owin.Security.Google
      - ...
    - Copier les fichiers d'une application Asp.Net MVC utilisant l'identification par défaut et y faire le ménage
      - ⚠ Attention aux version des packages NuGet des projets source/destination !
      - ⚠ Attention à l'espace de nom des projets source/destination !

ASP.NET Identity

- Fichiers impliqués →
  - À copier depuis une appli Asp.Net MVC créée non-vide
  - Models/IdentityModels.cs
    - Utilise EF Code First
    - Modifier pour personnaliser les utilisateurs
  - Views/Account/Login.cshtml
    - Utilise _ExternalLoginsListPartial.cshtml
    - Désactiver éventuellement
- Web.config
  - IdentityModels utilise une connexion nommée "DefaultConnection"…
    - Gérer ça d'une façon ou d'une autre



ASP.NET Identity - Exemple

- Avant tout
  - Ajouter les packages… Copier les fichiers…
  - Désactiver le login tiers : Login.cshtml + Register.cshtml
- Choix de la base de données

Web.config

```
<connectionStrings>
  <add name="WebApplication1Membership"
       connectionString="Data Source=.; Database=WebApplication1Membership; Integrated Security=True;
MultipleActiveResultSets=True;"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

/Models/IdentityModels.cs

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser> {
  public ApplicationDbContext()
      : base( "WebApplication1Membership", throwIfV1Schema: false ) {
  }
  public static ApplicationDbContext Create() {
    return new ApplicationDbContext();
  }
}
```

ASP.NET Identity - Exemple

- Sécuriser une action

```
public class ExempleSecuriteApplicativeController : Controller {
  public ActionResult Index() {
    ViewBag.Utilisateur = User.Identity.Name;
    ViewBag.UserIdApplicatif = User.Identity.GetUserId();                    ← SSI using Microsoft.AspNet.Identity
    ViewBag.IdentiteIIS = System.Security.Principal.WindowsIdentity.GetCurrent().Name;
    using( var dc = new NorthwindEntities() ) {
      return View( "Index", dc.Products.Where( p => p.CategoryID == 1 )
                              .Select( p => new { p.ProductID, p.ProductName } ).ToList()
                              .Select( p => new ProductPourListeViewModel() {
                                ProductID = p.ProductID,
                                ProductName = p.ProductName
                               } ).ToList() );
    }
  }
  [ActionName("Supprimer"), Authorize( Users = @"toto@toto.com" )]     ← Si seulement Post, pb sur retour de login
  public ActionResult Supprimer( int? productID ) {
    if( productID == null )                                              ← Survient sur une redirection après login
      return RedirectToAction( "Index" );
    ViewBag.Message = $"On dit que {productID} est supprimé...";
    return Index();
  }
}
```

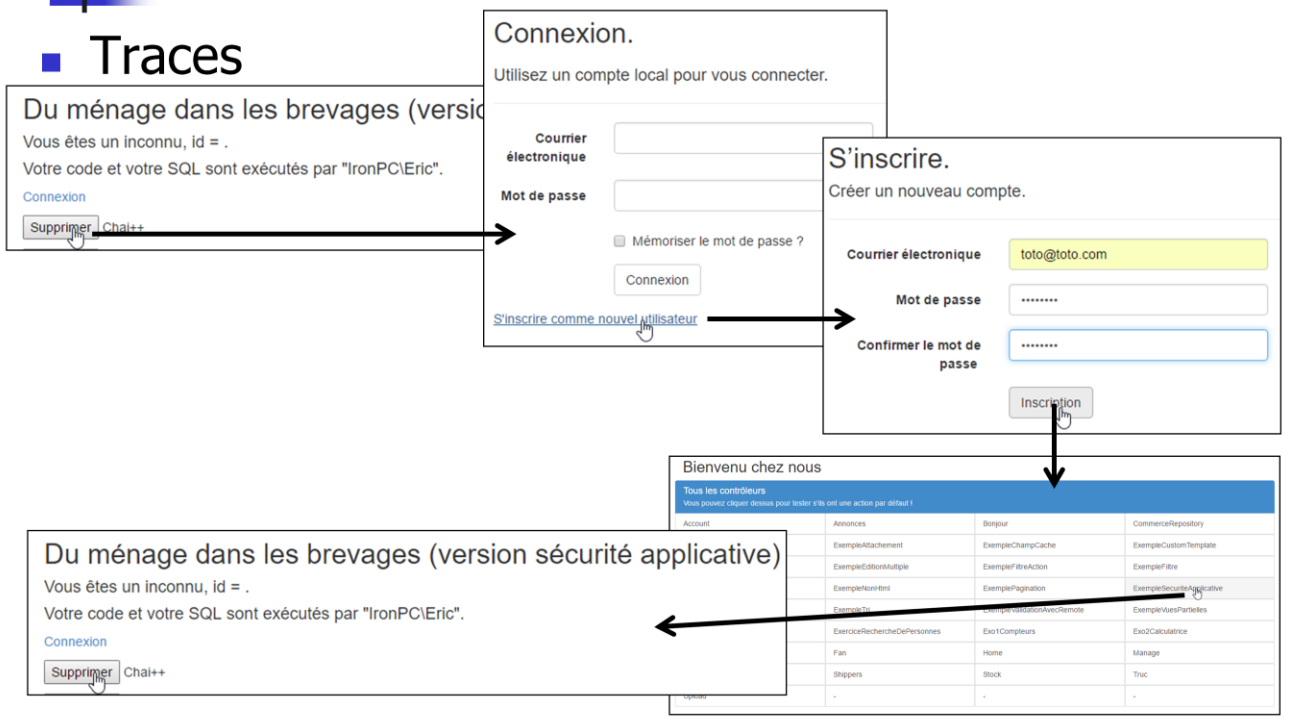
ASP.NET Identity - Exemple

- Proposer connexion / déconnection dans la vue

```
@using WebApplication1.Models
@model List<ProductPourListeViewModel>
@{ViewBag.Title = "Du ménage dans les brevages (version sécurité applicative)";
  var lesBrevages = Model; string message = ViewBag.Message;
  string identiteIIS = ViewBag.IdentiteIIS;
  string utilisateur = ViewBag.Utilisateur == string.Empty ? "un inconnu" : ViewBag.Utilisateur;
  string userIdApplicatif = ViewBag.UserIdApplicatif;}
<h2>@ViewBag.Title</h2>
<h4>Vous êtes @(utilisateur), id = @userIdApplicatif.</h4>
<h4>Votre code et votre SQL sont exécutés par "@(identiteIIS)".</h4>
@if( userIdApplicatif != null ) {
  using( Html.BeginForm( "LogOff", "Account" ) ) {          ← LogOff est disponible seulement sur Post, évidemment
    @Html.AntiForgeryToken()                                ← Critique, évidemment
    <input type="submit" value="Déconnexion" class="btn-link btn-warning" />    ← La déconnexion renvoie vers Home
  }
}
else {
  @Html.ActionLink( "Connexion", "Login", "Account", new { returnUrl = Request.Url.PathAndQuery }, ← Pour revenir
                                                     new { @class = "btn-link btn-warning" } )
}
<p>@message</p>
@foreach( var b in lesBrevages ) {
  using( Html.BeginForm( "Supprimer", "ExempleSecuriteSysteme" ) ) {
    <p><input type="submit" value="Supprimer" /> <input type="hidden" name="productID" value="@b.ProductID" />
       @b.ProductName</p>
  }
}
```

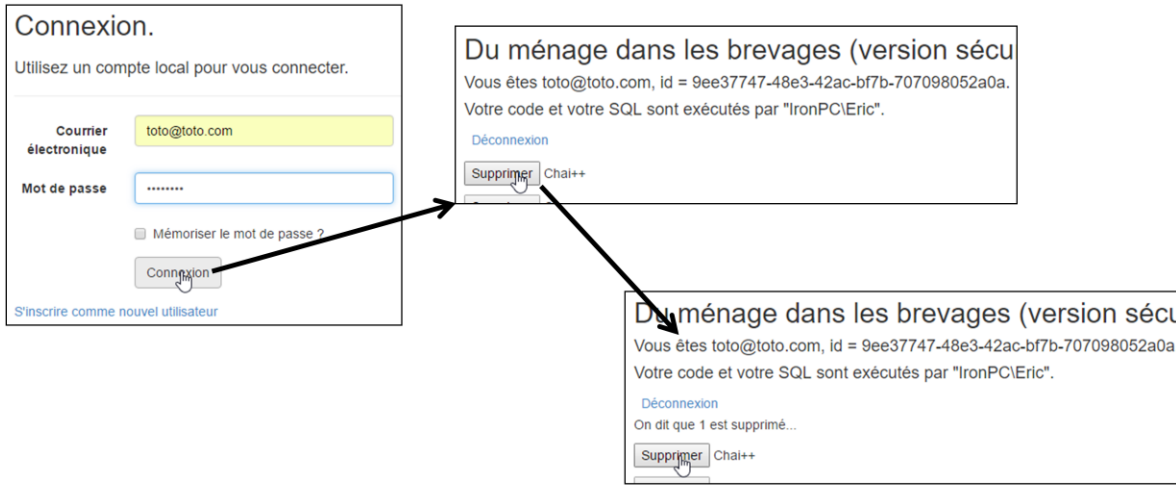
ASP.NET Identity - Exemple

- Traces



ASP.NET Identity - Exemple

- Traces



ASP.NET Identity - Gérer les rôles

- Gérer les l'utilisateurs et les rôles : pas drôle

ASP.NET Identity - Gérer les rôles

- ApplicationUserManager ou ApplicationDbContext, c'est à voir

```
[Authorize( Users = "agathe.zepower@monsieurmadame.fr" )]
public class ExempleGestionUtilisateursController : Controller {
  ApplicationUserManager _ApplicationUserManager = null;
  public ApplicationUserManager ApplicationUserManager {                    ← Pour les opérations classiques
    get {
      if( _ApplicationUserManager == null ) // Ssi using Microsoft.AspNet.Identity.Owin
        _ApplicationUserManager = HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
      return _ApplicationUserManager;
    }
  }
  private ApplicationDbContext _DC = null;
  public ApplicationDbContext DC {                                          ← Pour un accès libre à la base de données
    get {
      if( _DC == null )
        _DC = new ApplicationDbContext();
      return _DC;
    }
  }
protected override void Dispose( bool disposing ) {
  if( disposing ) {
    if( _DC != null )
      _DC.Dispose();
    if( _ApplicationUserManager != null )
      _ApplicationUserManager.Dispose();
  }
  base.Dispose( disposing );
}
```

ASP.NET Identity - Gérer les rôles

- Lister utilisateurs et rôles
  - Contexte mal fichu : les rôles d'un utilisateurs sont des couples RoleId / UserId !
  - Idem pour les rôles d'un utilisateur !

```
public ActionResult Index() {
// u.Roles ne contient des (RoleId + UserId) !
  ViewBag.Utilisateurs = DC.Users.Include( u => u.Roles ).OrderBy( u => u.UserName ).ToDictionary( u => u.Id );
  // Idem pour r.Users !
  ViewBag.Roles = DC.Roles.Include( r => r.Users ).OrderBy( r => r.Name ).ToDictionary( r => r.Id );
  return View( DC.Users.Find( User.Identity.GetUserId() ) );

  //var alternative = from r in DC.Roles
  //                  select new {
  //                    Name = r.Name,
  //                    Users = from ruID in r.Users
  //                            join u in DC.Users on ruID.UserId equals u.Id
  //                            select u.UserName
  //                  };
}
```

ASP.NET Identity - Gérer les rôles

- Gérer les rôles

```
[HttpPost, ValidateAntiForgeryToken]
public ActionResult CreerRole( string nvRole ) {
  DC.Roles.Add( new IdentityRole( nvRole ) );
  DC.SaveChanges();
  return RedirectToAction( "Index" );
}

[HttpPost, ValidateAntiForgeryToken]
public ActionResult DonnerLeRoleALUtilisateur( string role, string utilisateur ) {
  ApplicationUserManager.AddToRole( utilisateur, role );
  return RedirectToAction( "Index" );
}

[HttpPost, ValidateAntiForgeryToken]
public ActionResult EnleverLeRoleALUtilisateur( string role, string utilisateur ) {
  ApplicationUserManager.RemoveFromRole( utilisateur, role );
  return RedirectToAction( "Index" );
}
```

ASP.NET Identity - Gérer les rôles

- Vue

```
@using WebApplication1.Models                    @* Pour ApplicationUser *@
@using Microsoft.AspNet.Identity.EntityFramework  @* Pour IdentityRole *@
@model ApplicationUser
@{
  ViewBag.Title = "Gestion des utilisateurs";
  Dictionary<string, ApplicationUser> utilisateurs = ViewBag.Utilisateurs;
  Dictionary<string, IdentityRole> roles = ViewBag.Roles;
  var utilisateurCourant = Model;
}

<h2>@ViewBag.Title</h2>

<p>Vous êtes @utilisateurCourant.UserName (@utilisateurCourant.Id)</p>

@using( Html.BeginForm( "CreerRole", "ExempleGestionUtilisateurs" ) ) {
  @Html.AntiForgeryToken()
  <p class="form-inline">
    <input type="text" name="nvRole" placeholder="Nouveau rôle" class="form-control" />
    <input type="submit" value="Créer un nouveau rôle" class="btn btn-default" />
  </p>
}
```

ASP.NET Identity - Gérer les rôles

- Outil pour concaténer une collection en une chaine

```
@functions {
  string Concatener<T>( IEnumerable<T> trucs, Func<T, string> extraireChaine ) {
    var sb = new System.Text.StringBuilder();
    foreach( var truc in trucs )
      sb.AppendFormat( "{0}, ", extraireChaine( truc ) );
    if( sb.Length > 0 ) {
      sb.Length -= 2;
      return sb.ToString();
    }
    return " - ";
  }
}
```

ASP.NET Identity - Gérer les rôles

- Utilisateurs et rôles

```
<div class="row">
  <div class="list-group col-md-6">
    <p class="list-group-item active">Utilisateurs (et leurs rôles)</p>
    @foreach( var u in utilisateurs.Values ) {
      <label class="list-group-item">
        <input type="radio" name="UserId" value="@u.Id" onchange="$('[name=utilisateur]').val(this.value)" />
        @u.Email (@(Concatener<IdentityUserRole>( u.Roles, role => roles[role.RoleId].Name ))
      </label>
    }
  </div>
  <div class="list-group col-md-6">
    <p class="list-group-item active">Rôles (et leurs utilisateurs)</p>
    @foreach( var r in roles.Values ) {
      <label class="list-group-item">
        <input type="radio" name="Role" value="@r.Name" onchange="$('[name=role]').val( this.value )" />
        @r.Name (@(Concatener<IdentityUserRole>( r.Users, u => utilisateurs[u.UserId].UserName ))</label>
    }
  </div>
</div>
```

ASP.NET Identity - Gérer les rôles

- Déclencher les actions

```
@using( Html.BeginForm( "DonnerLeRoleALUtilisateur", "ExempleGestionUtilisateurs",
                        FormMethod.Post, new { @class = "inline" } ) ) {
  @Html.AntiForgeryToken()
  <input type="hidden" name="role" id="role" />
  <input type="hidden" name="utilisateur" />
  <input type="submit" value="Donner le rôle à l'utilisateur" class="btn btn-success" />
}
@using( Html.BeginForm( "EnleverLeRoleALUtilisateur", "ExempleGestionUtilisateurs",
                        FormMethod.Post, new { @class = "inline" } ) ) {
  @Html.AntiForgeryToken()
  <input type="hidden" name="role" />
  <input type="hidden" name="utilisateur" />
  <input type="submit" value="Enlever le rôle à l'utilisateur" class="btn btn-danger" />
}
```

Exercice

- Activer la sécurité applicative par défaut
- Personnaliser
  - Désactiver l'identification tierce
  - Utiliser une BD existante
  - Exiger l'authentification sur un contrôleur existant
    - Ajouter à sa/ses vue(s) une possibilité de déconnexion
  - Ajouter la gestion "*des*" données personnelles de l'utilisateur