Contents

- Settings
- Authentication
  - Forms Authentication
  - Windows Authentication
- Roles
- Profile
- Application Name
- Customizations
  - Credentials (Forms authentication)
  - IEntityLoginManager
  - UserBase
- Troubleshooting

DevForce provides out-of-the-box integration with **ASP.NET security** features (Membership, Roles, Profile). These features can be used in any application -- Silverlight, desktop, or ASP.NET. If developing for Silverlight or ASP.NET, choosing ASP.NET security is often the right choice over a custom security implementation. ASP.NET security is easy to use, highly configurable, and well-documented. See http://www.asp.net/learn/security/ for information on ASP.NET security features.

Settings

In order to use ASP.NET security in DevForce you must set the *UseAspNetSecurityServices* flag in the web.config or server .config to enable it. When enabled, DevForce will use the *AspAuthenticatingLoginManager* to handle login requests from clients.

```
XM  <objectServer>
      <serverSettings useAspNetSecurityServices="true" />
    </objectServer>
```

You must also enable *AspNetCompatibility* in order to allow the DevForce services to integrate with ASP.NET services. You set this in the *system.serviceModel* configuration section. Here's the relevant element in the system.serviceModel section:

```
XM  <system.serviceModel>
      <serviceHostingEnvironment aspNetCompatibilityEnabled="true" />
    </system.serviceModel>
```

You must enable the ASP.NET services you wish to use in the *system.web* configuration section of the config file, as well as choose the type of authentication wanted. These steps are described below.

Authentication

Authentication in ASP.NET can take either of two flavors – Forms or Windows. For either type of authentication, after a successful *Login* completes a *UserBase* instance representing the user is available on both client and server.

Forms Authentication

Forms authentication involves validating user credentials against a Membership provider. The default provider uses a SQL Server Membership database, aspnetdb, to store user information. ASP.NET also supplies a Membership provider for Active Directory, or you can write a custom provider.

To use Forms authentication, specify this authentication mode in the system.web configuration section:

```
XM  <system.web>
      <authentication mode="Forms" />
    </system.web>
```

In your application you can ask the user for login credentials and pass the credential in the *Authenticator.Login* call. DevForce will validate the credentials with the ASP.NET membership provider. If the user is authenticated, a FormsAuthenticationTicket is issued. If you want the ticket to be persistent you should pass a FormsAuthenticationLoginCredential in the Login call, since this credential allows you to set the persistence flag.

You can also call *Authenticator.Login* with a null argument if your application accepts either persistent authentication tickets or the user has already logged in as part of the larger application.

Windows Authentication

When using Windows authentication in ASP.NET the current Windows credentials of the client are transmitted to the server. This can be used in intranet environments only.

To use Windows authentication, specify this authentication mode in the system.web configuration section:

```
XM <system.web>
     <authentication mode="Windows" />
   </system.web>
```

Note that additional configuration changes are required both in IIS and in the communications configuration in order to pass Windows credentials to the EntityServer. These changes are discussed in the configuration topic.

On your client, call the *Authenticator.Login* method with a null credential. The *AspAuthenticatingLoginManager* will check the HttpContext.Current.User for a WindowsPrincipal representing the user, and from that create a UserBase to be returned to the client.

Roles

You must enable the Role service in the configuration file in order to use this feature:

```
XM <system.web>
     <roleManager enabled="true" />
   </system.web>
```

With roles enabled, user role information will be obtained from the ASP.NET RoleProvider, and role-based authorization can be used in your application. Use UserBase.Roles to retrieve all roles for the user, and UserBase.IsInRole() to determine role membership.

Check the ASP.NET documentation for information on how to create and manage roles and assign users to roles.

Profile

You must enable the Profile service in the configuration file in order to use this feature, and define the profile properties wanted. Here's a sample:

```
XM <system.web>
     <profile enabled="true">
       <properties>
       <!-- Sample properties -->
         <add name="WindowSeat" type="bool" defaultValue="false" />
         <add name="Building" type="string" defaultValue="A" />
       </properties>
     </profile>
   </system.web>
```

You also need to extend the *UserBase* class with the custom properties from your profile. DevForce will automatically populate these properties from the Profile if the property name and type match, and the setter is public. Your custom UserBase class must be serializable, since it will be transmitted between client and server tiers.

Application Name

If relying on the application name for authentication you must explicitly set it in all membership providers:

```
XM <system.web>
     <membership>
       <providers>
         <clear/>
         <add name="AspNetSqlMembershipProvider"
              type="System.Web.Security.SqlMembershipProvider, System.Web, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
              connectionStringName="LocalSqlServer"
              enablePasswordRetrieval="false"
              enablePasswordReset="true"
```

```
        requiresQuestionAndAnswer="true"
        requiresUniqueEmail="false"
        passwordFormat="Hashed"
        maxInvalidPasswordAttempts="5"
        minRequiredPasswordLength="7"
        minRequiredNonalphanumericCharacters="1"
        passwordAttemptWindow="10"
        passwordStrengthRegularExpression=""
        applicationName="/MyDevForceApp" />
    </providers>
  </membership>
</system.web>
```

Customizations

Credentials (Forms authentication)

You can pass custom credentials, derived from *ILoginCredential*, *LoginCredential*, or *FormsAuthenticationLoginCredential* with the Login call. With custom credentials, you will generally also want to provide a custom *IEntityLoginManager* implementation to receive these credentials. If you wish to take advantage of existing DevForce ASP.NET service integration, you should derive your class from the *AspAuthenticatingLoginManager* and override methods as needed.

IEntityLoginManager

You can implement your own *IEntityLoginManager* or extend the *AspAuthenticatingLoginManager* to provide custom logic. Any custom implementation will be used if found.

UserBase

You can also extend the *UserBase* class. If you enable the ASP.NET Profile service you will want to use a custom UserBase which contains additional properties retrieved from the profile. DevForce will automatically return your custom UserBase (if found) without the need to implement a custom *AspAuthenticatingLoginManager*.

Troubleshooting

"Error using ASP.NET Membership: Unable to connect to SQL Server database." Message received on a Login call.

This will occur if the ASP.NET membership database cannot be found or opened. You must configure the ASP.NET membership provider if you wish to use ASP.NET security features, and by default the *AspNetSqlProvider* is used. This will use the *LocalSqlServer* connection string from either your web.config or the machine.config. The default connection expects a SQLExpress database named *aspnetdb.mdf*. For more information on configuring ASP.NET membership see the membership tutorials at http://www.asp.net/learn/security/ .

The default in the machine.config:

```
XM  <connectionStrings>
      <add name="LocalSqlServer" connectionString="data source=
        .\SQLEXPRESS;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;
        User Instance=true" providerName="System.Data.SqlClient"/>
    </connectionStrings>
```

To override in your web.config, remove the old connection string and add the new connection string. Here's a sample re-pointing the connection to the default instance of SQL Server on the machine:

```
XM  <connectionStrings>
      <remove name="LocalSqlServer" />
      <add name="LocalSqlServer" connectionString=
       "Data Source=.;Initial Catalog=aspnetdb;Integrated Security=True;"
        providerName="System.Data.SqlClient" />
    </connectionStrings>
```

Can't find *IdeaBlade.EntityModel.Web* assembly
This assembly is automatically installed to web applications by the DevForce Server NuGet package. If you need to reference this assembly in other application types you can find it under the *..\packages\IdeaBlade.DevForce.Server.7.x.y\tools\lib\net45* folder.