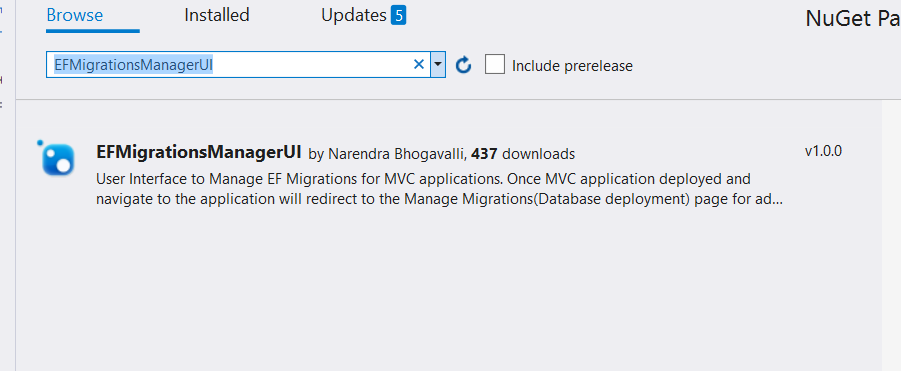
**EFMigrationsManagerUI – Install and Configuration**

In the previous article we discussed about the introduction of EFMigrationsManagerUI nuget package and its advantages. In this article, we will talk about how to install and integrate the EFMigrationsManagerUI nuget package with the asp.net MVC applications along with the sample demo web applications.

**Install EFMigrationsManagerUI:**

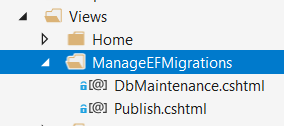
1. In Visual Studio, Open the target MVC project to enable Manage migrations user interface.
   * Right click on the MVC Project References and choose **Manage Nuget Packages.**
   * Search and add the EFMigrationsManagerUI Nuget package.



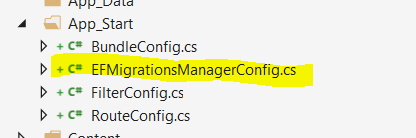
This Nuget package have dependencies with the following packages

* EntityFramework (>= 6.1.0)
* WebActivatorEx (>= 2.2.0)
* Microsoft.AspNet.Mvc (>= 5.2.3)

1. EFMigrationsManagerUI Nuget package will add the below files in target project.
   * EFMigrationsManager assembly reference and dependent assemblies if not available.
   * EFMigrationsManagerController.cs file under controller’s folder.
   * Adds EFMigrationsManager folder with below cshtml files under Views folder.
     1. Publish
     2. DbMaintenance



* + EFMigrationsManagerConfig.cs file under App\_Start folder.



1. Below app setting entry will add under web.config file.
   * <add key="EFMigrationsManagerUI:AuthorizedUsers" value="Comma seperated admin user names to deploy EF migrations" />

**Configure the Application:**

1. Install the “EFMigrationsManagerUI” nuget package.
2. Open the “EFMigrationsManagerConfig” class under App\_Start folder
   * Uncomment below line and pass the Entity Framework Configuration instance as parameter.

//EFMigrationsManagerSettings.SetEFConfiguration(new EFConfiguration());

* + 1. Search for Class in solution (usually in models or data projects) that inherits from DbMigrationsConfiguration class.
    2. Replace EFConfiguration with Class found in above step.

1. Open the web.config and navigate to the appSettings section
   * Update the user names in below app setting section.

<add key="EFMigrationsManager:AuthorizedUsers" value="Comma separated admin user names to deploy EF migrations" />

Note:

1. For multiple admins, enter comma separated **identity name**s.
2. For windows authentication, enter name as domain\username
3. For Forms authentication, enter name as [username@domain.com](mailto:username@domain.com) or [firstname.lastname@domain.com](mailto:firstname.lastname@domain.com) based on the identity name.

Now all set and ready to manage Entity Framework migrations through the simple user interface. Run the MVC application and navigate to the below pages.

Database deployment: EFMigrationsManager/Publish

Rollback database : EFMigrationsManager/Publish?isRollback=True

**Additional Configuration:**

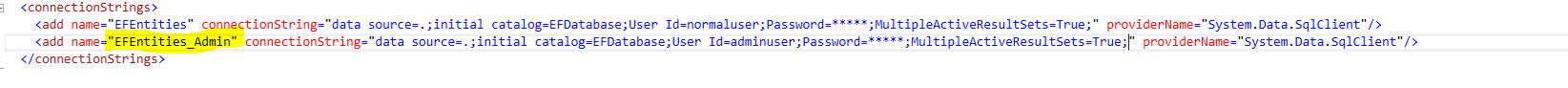
In this section, we will talk about the some additional optional configuration.

1. **Admin Connection String**:

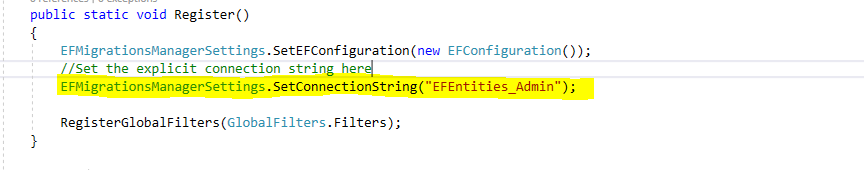
Its recommend using different connection string for EFMigrationsManagerUI. Most of the scenarios, Entity Framework migrations (to update database) need permission for schema changes (create/alter/drop table permissions).

Whereas application connection string doesn’t need those admin permissions and it’s not recommendable to provide admin permissions to application connection string.

* Create another connection string like below



* + Open the “EFMigrationsManagerConfig” class under App\_Start folder and navigate to the **Register** method. Call the EFMigrationsManagerSettings.SetConnectionString method with admin connection string. Please check the below image for quick reference.



Now application page will use connection string with read/write permissions based on the application requirement and EFMigrationManagerUI (database deployment pages) will use admin connection string.

1. **Ignore EF Migration Verification:**

If any specific application page is not required to check migrations are up to date or not, then use the below attribute at controller/action level



Example: Application Error pages. If application is throwing some database exception then application will redirect default application error pages. If again error pages are hitting the database to verify pending migrations, then application might throw cascading error in error pages. These kinds of situations skip the verification by using above attribute.

1. **Disable AutoDetect pending migrations – application level:**

By default, when install nuget package, AutoDetect feature is enabled. To disable this feature, remove below line from EFMigrationsManagerConfig class.



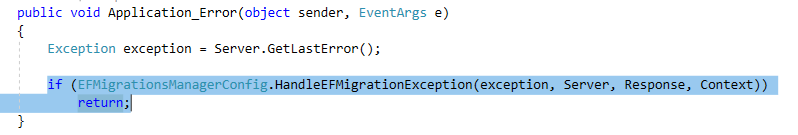
1. **Alternative approach to AutoDetect pending migrations:**

Default configuration is with MVC action filter on above step. MVC action filter will check pending migrations for each MVC page request. To avoid checking the migrations are up to date on every page request,

* + Remove the below line from EFMigrationsManagerConfig class.



* + Call the below highlighted method in Application\_Error. This method will check if exception is related to the Entity Framework and EF migrations context is mismatched with the database then redirect to the database deployment page.



Note: Entity Framework will throw the exceptions on database call if context is mismatched. If no database call on any specific pages then user will see the requested page instead of database deployment/maintenance pages even database is not up to date.

**Upcoming Enhancements:**

The current version of EFMigrationsManagerUI nuget package will support single database or single Entity Framework Configuration. If MVC web application have multiple Entity Framework configurations/databases then the current version of nuget package will handle to manage single configuration. Upcoming version of EFMigrationsManagerUI will support multiple configurations.

**Examples/Demos:**

Created below samples for MVC based applications and hosted in GitHib repository.

1. **MVC application with windows authentication:**

Follow the below steps to setup the working sample.

* + Download/Clone the source code from the below GitHub URL

<https://github.com/naren-b/EFMigrationsManagerUI_Sample_WindowsAuthentication>

* + Open the project (downloaded on above step) in visual studio.
  + Set ‘Sample\_WindowsAuthentication’ as startup project.
  + Open web.config file and update the below AppSetting entry with admin usernames.

<add key="EFMigrationsManager:AuthorizedUsers" value="domain\username" />

Note: For multiple admins, enter comma separated names.

* + Update the below connection string if required to test on different database or server.

<connectionStrings>

<add name="EFEntities" connectionString="data source=.;initial catalog=EFDatabase;Integrated Security=SSPI;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />

</connectionStrings>

1. **MVC application with forms based authentication**

Created sample web application integration with azure active directory. To know more about the Azure Active directory integration, please follow the below link.

<https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-integrating-applications>

Follow the below steps to setup working sample.

* + Download/Clone the source code from the below GitHub URL.

<https://github.com/naren-b/EFMigrationsManagerUI_Sample_FormsAuthentication>

* + Open the project (downloaded on above step) in visual studio.
  + Set ‘FormsAuthentication’ project as startup.
  + Open web.config file and update the below AppSetting entry with admin usernames.

<add key="EFMigrationsManager:AuthorizedUsers" value="username@domain.com" />

Note: For multiple admins, enter comma separated names.

* + Update the below connection string if required to test on different database.

<connectionStrings>

<add name="EFEntities" connectionString="data source=.;initial catalog=EFDatabase;Integrated Security=SSPI;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />

</connectionStrings>

* + Update the below azure active directory integration related settings

<add key="ida:ClientId" value="" />

<add key="ida:ClientSecret" value="" />

<add key="ida:Domain" value="" />

<add key="ida:TenantId" value="" />

<add key="ida:PostLogoutRedirectUri" value="" />

**Conclusion:**

Entity Framework code first is very useful in [Domain Driven Design](https://msdn.microsoft.com/en-us/magazine/dd419654.aspx) and integrating the EFMigrationsManagerUI plug in will provide friendly user interface to manage migrations. In the next article, we will cover about the integrating multiple Entity Framework configurations to the web application.