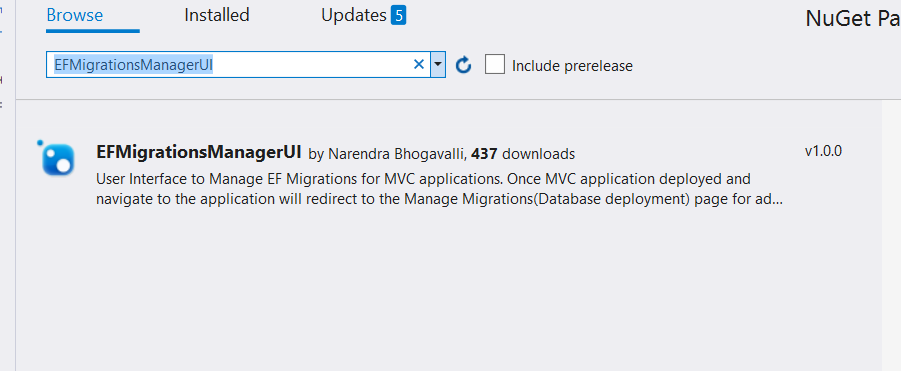
**EFMigrationsManagerUI – Installation and Configuration**

In the previous article we discussed the user Interface to manage Entity Framework migrations 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 web applications.

**Install EFMigrationsManagerUI:**

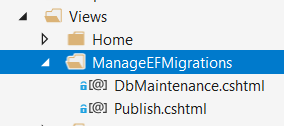
1. In visual studio, Open the target MVC project to enable manage migrations user interface.
   * Right click on the MVC project references and choose **Manage Nuget Packages.**
   * Search and add the EFMigrationsManagerUI Nuget package.



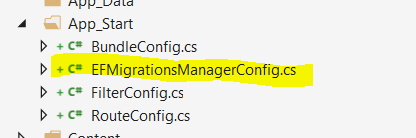
This Nuget package have dependencies with the following packages:

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

1. EFMigrationsManagerUI Nuget package will add the following files in the 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. Adds the below AppSetting entry in web.config file.
   * <add key="EFMigrationsManagerUI:AuthorizedUsers" value="Comma seperated admin user names to deploy EF migrations" />

**Configuration:**

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());

Follow the below steps to find the EFConfiguration class in solution.

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

1. Open the web.config and navigate to the AppSetting section
   * Update the user names for the below AppSetting.

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

Note:

1. For multiple admins, enter comma separated **identity names**.
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:**

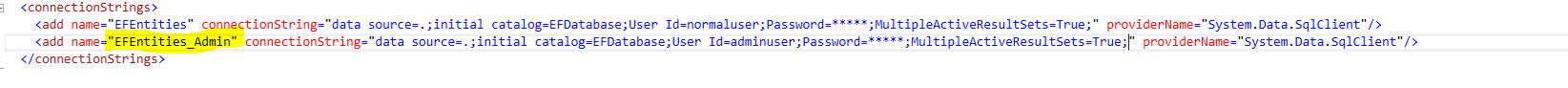
In this section, we will talk about the some additional optional configuration.

1. **Admin connection string or EFMigrationsManagerUI specific connection string:**

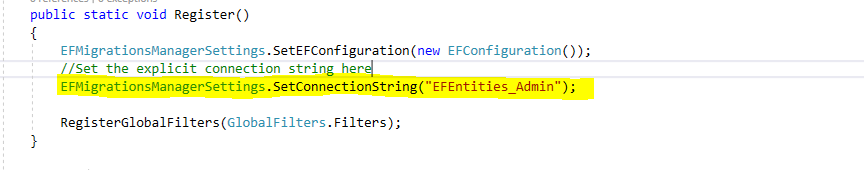
Its recommended to use different connection string for EFMigrationsManagerUI. In most cases, Entity Framework migrations (to update database) needs permission for schema changes (create/alter/drop permissions).

Whereas application connection string doesn’t need the admin permissions and it’s also not recommended. Follow the below steps to provide different connection string to deploy database other than the connection string from Entity Framework configuration.

* 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 name. Please check the below image for quick reference.



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

1. **Disable AutoDetect pending migrations – controller/action level:**

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



Example: No need to check the EF Pending migrations for error pages. If application is throwing some database exception then application will redirect to the default application error page. Again, if error pages are hitting the database, then application will throw cascading error in error pages. To avoid these cases, skip the verification check by using above attribute.

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

By default, when installing the Nuget package, AutoDetect feature is enabled. To disable this feature, remove below line from EFMigrationsManagerConfig class and access the database deployment pages with URL or navigation links.



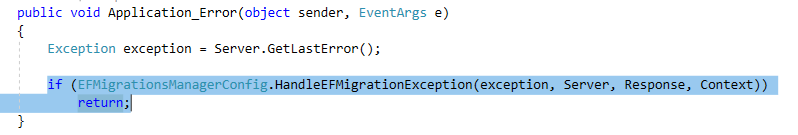
1. **Alternative approach to AutoDetect pending migrations:**

Default configuration is with MVC action filter. MVC action filter will check pending migrations for each MVC page request. Follow the below steps to avoid checking the migrations are up to date or not 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 migrations context is mismatched with the database then redirect to the database deployment page based on the user permissions.



Note: Using this approach, application will redirect to the database deployment page when requested page is trying to talk to the database. If no database calls are present on the requested page then user will see the requested page even database is not up to date with Entity Framework migrations.

**Upcoming Enhancements:**

The current version of EFMigrationsManagerUI 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 manage single configuration only. Upcoming version of EFMigrationsManagerUI will support multiple EF configurations.

**Examples/Demos:**

Created below samples for MVC based applications and hosted the source code in GitHub repository.

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

Follow the below steps to setup the working sample.

* + Download/Clone the source code from the following 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 the 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 authentication**

Created the sample web application integrated 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 following GitHub URL.

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

* + Open the project (downloaded on above step) in visual studio.
  + Set ‘FormsAuthentication’ as startup project.
  + 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 EF migrations. In the next article, we will talk about the user interface to manage migrations for multiple Entity Framework configurations/databases.