**User Interface to Deploy/Manage Entity Framework migrations**

In this article, I’ll describe about the User Interface to Deploy and Manage entity framework migrations for ASP.NET MVC applications.

Please check the below links for information about Entity Framework, ASP.NET MVC, Nuget and GitHub:

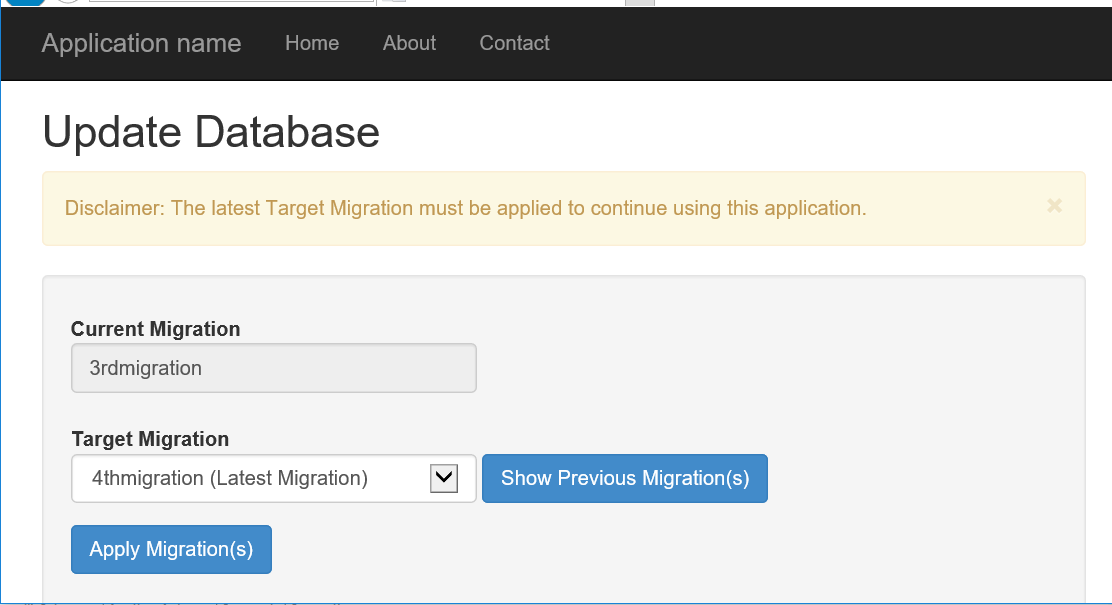
* [Entity Framework Code First to a New Database](https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx)
* [Entity Framework Code First to an Existing Database](https://msdn.microsoft.com/en-us/library/jj200620(v=vs.113).aspx)
* [ASP.NET MVC](https://www.asp.net/mvc)
* Nuget
* Github

**Overview:**

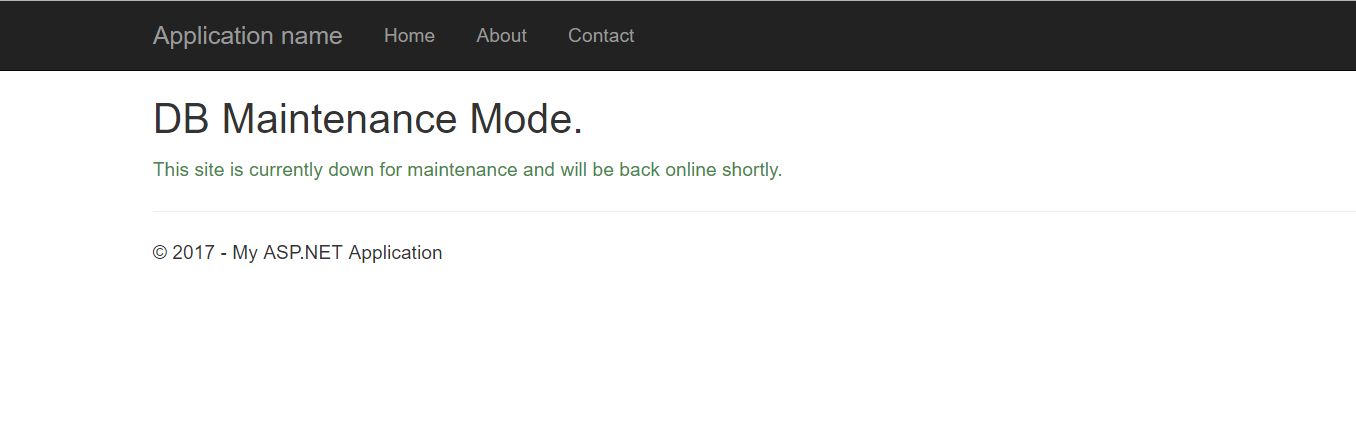
Upon successful deployment of MVC web application, Admin(DevOps/QA/Business) User’s will navigate to the web application and validate the deployment.

If Latest migration was not deployed to the database, depending on the user role, user will see either one of the below pages.

* For admin users (based on the AppSetting entry), Application will auto redirect to the Database Deployment user Interface page



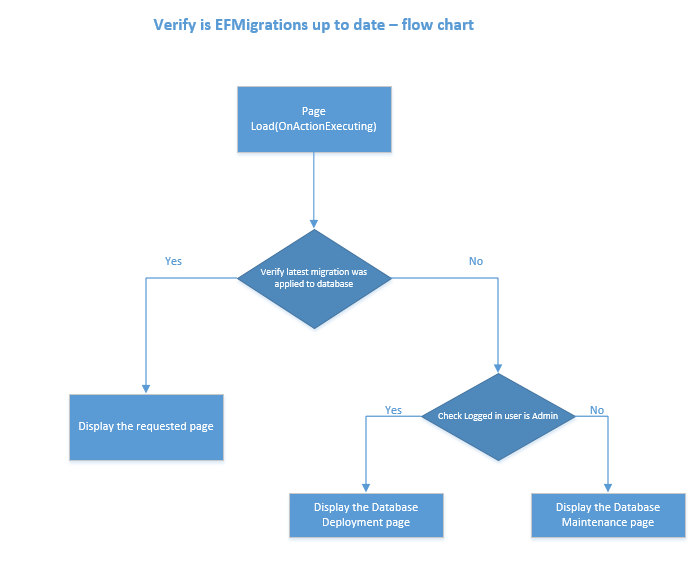
* For non-admin Users, Application will redirect to the Maintenance Mode page.



Note: As views (cshtml files) are part of the MVC Project, User Interface Look and feel can be changed according to the Project standards.

**Auto Detect and Algorithm**:

By default, when install EFMigrationsManagerUI nuget package, AutoDetect feature is enabled. Please check the below flow chart.



**Algorithm**:

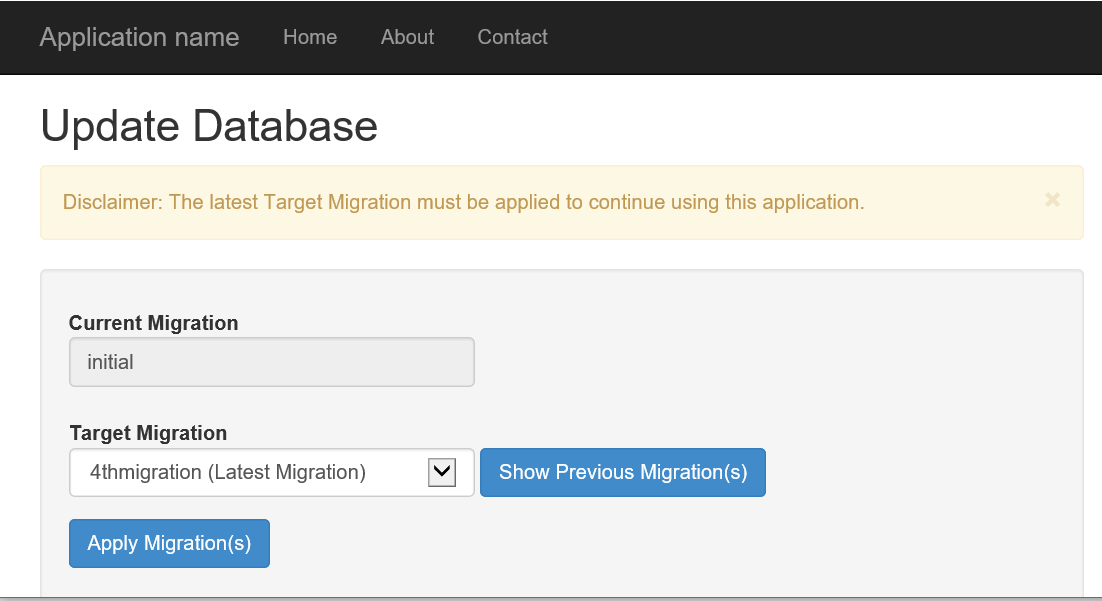
1. On page load (OnActionExecurint), Application will verify that “Latest migration was applied to the database” or not.
2. If latest migration was applied then application will render the content of the user requested page.
3. If latest migration was not applied then application will check weather logged in user is admin or not.
   1. If logged in user is admin then application will redirect to the Manage EF Migrations manager page to update the database.
   2. If logged in user is non-admin, then application will redirect to the Database maintenance page.

**Walk thru the User Interface:**

1. **Deploy Pending Migrations:**

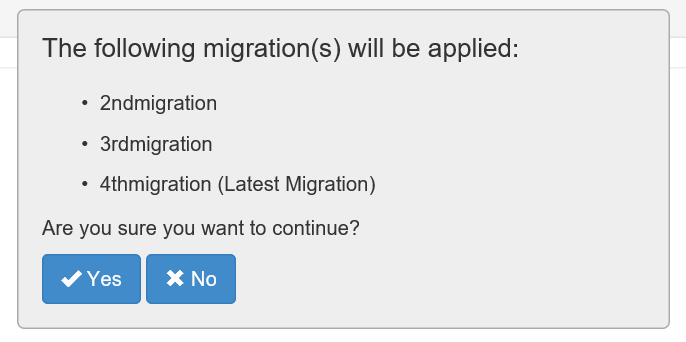
Navigation Url: EFMigrationsManager/Publish

If AutoMigration check is enabled then application will auto redirect to the below page when latest migration was not deployed to the database.



By default, Target Migration is Latest Version and clicking on “Show Previous Migration(s)” button will navigate to the “Rollback Migration(s)” page.

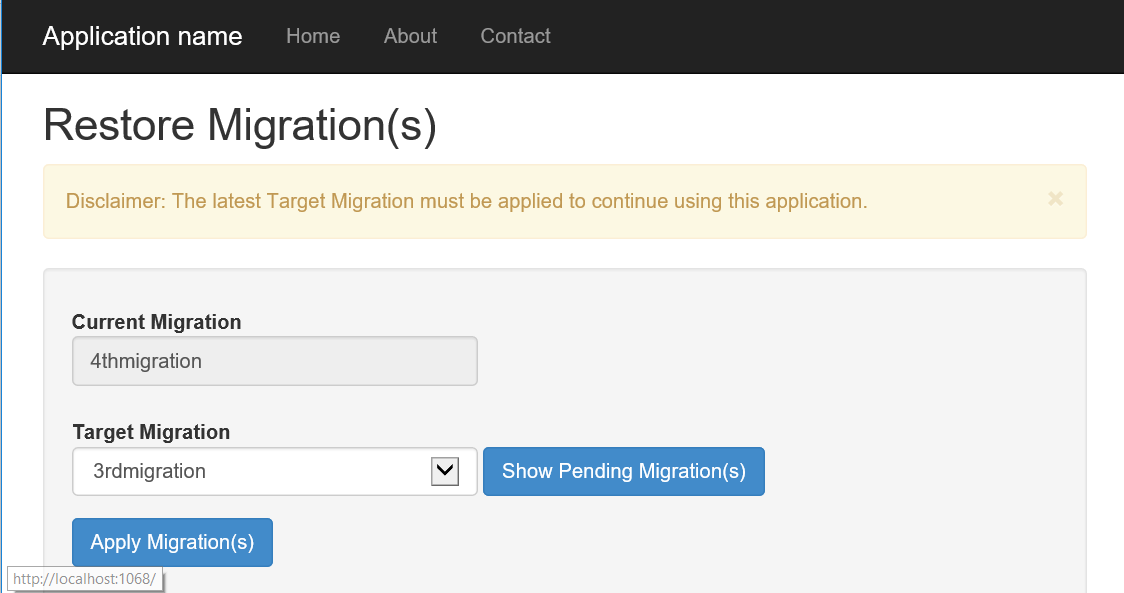
Once user selects the Target Migration and click on Apply migrations will show the below confirmation message.



1. **Rollback Migrations:**

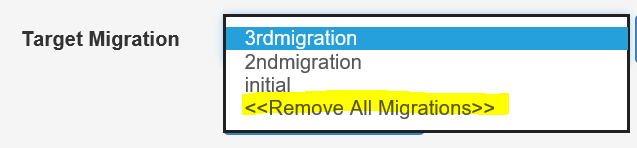
when deployment fails or database migration needs to rollback, then clicking on the “Show Previous Migrations” button on “Update Database” page or navigating to the below URL will give the capability to rollback the database migrations.

Navigation Url: EFMigrationsManager/Publish?isRollback=True

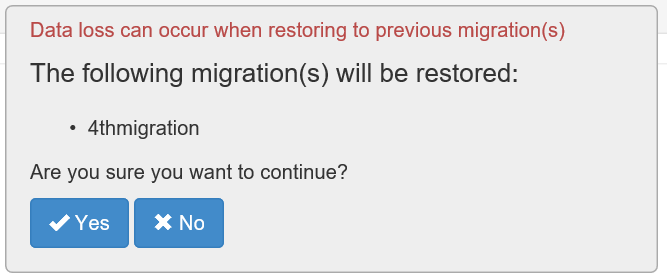


By default, Target Migration is immediate “Previous Migration” to the current migration. Select “Remove All Migrations” in the below scenario(s)

* + - When application is in Initial migration state and want to rollback the current migration
    - Need to rollback all migrations from database



Click on Apply Migration(s) will display the below confirmation box



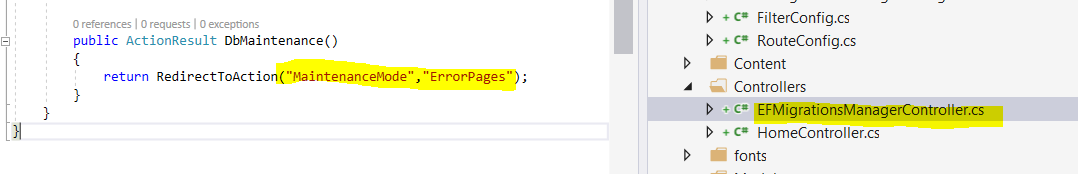
Click on “Yes” button will roll back the selected migrations.

1. **Database Maintenance page:**

This is the static page which will display when satisfying the below two conditions

* + Database is not up to date with EF migrations
  + Logged in user is non-admin.

Note: If application have similar maintenance page then redirect to that page from the below controller action.



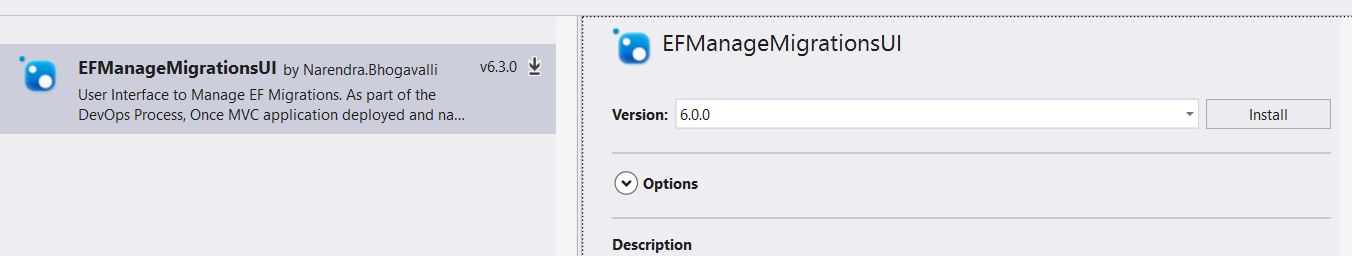
**Advantages:**

1. Not everyone in the DEV team members have knowledge on Entity Framework or Package Manager Console to update/rollback migrations. For instance, some team members are experts in client side technologies. This user interface will helpful to Dev Team members to manage migrations even dev team is not expert on entity framework code first.
2. Easy to integrate with DevOps process. Upon successful deployment or Upon successful swap the Stage/Prod environments, navigate to the Application will auto redirect the database deployment page for admin users.
3. With simple interface, Easy to apply to all the pending migrations or rollback to previous migrations.
4. Works MVC applications with Windows and Forms based authentication.
5. Always helpful to test the application close to the production environment. As part of this process and periodic database restores on QA/UAT/Stage environments with production backups, corresponding web application will break with mismatched database context. With this tool, navigate to the corresponding web application URL and publish using user interface without waiting for another database migrations deployment or DevOps engineer.
6. Manage migrations are now testable and trackable.
7. No need to store SQL connection string in different places like Source Code/VSTS build…. MigrationManager are using connection string as part of the ConnectionString section from web.config.

Note: These connection strings can manage from Azure portal if it is Azure Web apps.

**Install EFManageMigrationsUI:**

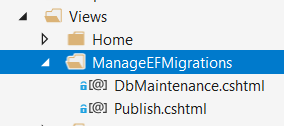
1. In Visual Studio, Open the MVC project
   * Right click on the MVC Project References and choose **Manage Nuget Packages.** Add the EFManageMigrationsUI Nuget package.



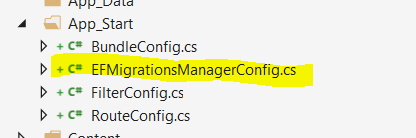
This Nuget package have dependencies with the following packages

* + EntityFramework
  + Microsoft.AspNet.Mvc

1. Above Nuget package will add the below files
   * EFManageMigrations assembly reference
   * In Controllers folder, Add EFMigrationsManagerController class
   * In Views Folder, Adds EFMigrationsManager folder with below views
     1. Publish
     2. DbMaintenance



* + Under App\_Start folder, Adds EFMigrationsManagerConfig.cs file



1. AppSettings, Adds the below App setting entry
   * <add key="EFManageMigrations:AuthorizedUsers" value="Comma seperated admin user names to deploy EF migrations" />

Update the connection string name. Preferably different connection string with Admin permissions as EF Migrations may need Create/Alter/Drop tables.

Usually application needs Read, Write and execute permissions. It’s better to separate Application connection string with EfManageMigrations connection string.

**Configure the Application:**

In this section, we will talk about the ways to configure the application.

* Install the “EFMigrationsManagerUI” nuget package
* Open the “EFMigrationsManagerConfig” class under App\_Start folder
  + Uncomment below line

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

* + Pass the Entity framework configuration as parameter to the SetEFConfiguration method.
* 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: For windows authentication, Use domain\username

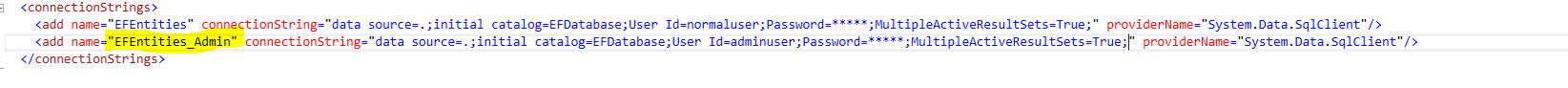
For Forms authentication, [username@domain.com](mailto:username@domain.com) or [firstname.lastname@domain.com](mailto:firstname.lastname@domain.com)

**Advanced Configuration:**

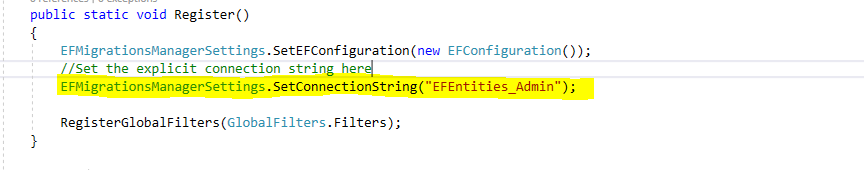
1. Recommendation: Its recommend using different connection string for EFMigrationsManager. Most of the scenarios, Entity Framework migrations (For update database) need permission for schema changes (Create/Alter/Drop table permissions).

Whereas applications connection string doesn’t need those admin permissions and it’s not recommendable also.

* 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.



1. If any 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. For example, if application is throwing some database exception then application will redirect default error pages. If again error pages are hitting the database then application might throw another error in error pages. These kinds of situations skip the verification by using above attribute.

1. Disable auto detection of pending migrations and redirect to the deployment page:

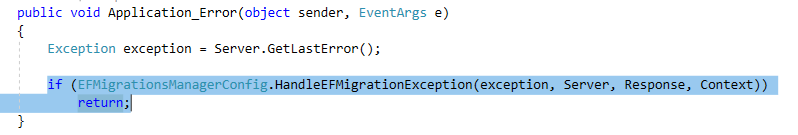
By default, when install nuget package, AutoDetect feature is enabled. To disable that feature, remove below line from EFMigrationsManagerConfig class.



1. Alternative approach to Auto Detection: Default configuration is with MVC action filter on above step. MVC action filter will run for every MVC page request. To avoid checking the migrations are up to date on every page request,
   * Follow the above step. (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: Application will throw the EFException on calls with database interactions. 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.