**User Interface to Manage Entity Framework migrations**

**Introduction:**

While Entity Framework (EF) code first provides good tooling for developers to apply migrations from the command line or Package Manager Console, there is no built-in way to monitor migration status directly from an [ASP.NET](http://asp.net/) web site. Also, troubleshooting EF issues, such as context change exceptions, can require multiple steps. EFMigrationsManagerUI nuget package provides an easy way to understand user interface that your deployment and administrative team can use to verify and apply pending migrations directly from your site. Included exception-handling features ensure that users are given a user-friendly maintenance message if migrations need to be applied, instead of default [ASP.NET](http://asp.net/) error pages. Read on to learn how EFMigrationsManagerUI can help your EF migration workflow.

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

These days, Entity Framework is most popular ORM in Microsoft technologies. Entity framework Code first provides lot of capabilities and advantages for Microsoft developers (I am not covering the entity framework advantages in this article). To deploy/rollback the entity framework migrations to database, developers need to know about the entity framework commands like update-database. There is no simple built-in user interface to manage migrations like below

* Apply Migrations
* Rollback Migrations
* Find the
  + Current Migration Migration applied to database.
  + Pending migrations needs to apply to the database.
* Not everyone in the development team is expert in entity framework to manage migrations. For instance, some of the team members are experts in UI design or client side technologies like Angular JS, jQuery, etc. If the entity framework migrations are not in sync with database then application will throw context change error. Without the Entity Framework migrations deployment, the web application will not run in the local/development machine and UI designer is unable to work on the web site until another team member(s) can help the designer to deploy the database.

**EFMigrationsManagerUI Nuget Package:**

To handle the above cases, created open source plugin called **EFMigrationsManagerUI** to manage entity framework migrations through user interface. This plugin was hosted in Nuget.org, which will be very helpful to download and integrate in other asp.net MVC applications. Following is the URL for **EFMigrationsManagerUI** Nuget package.

<https://www.nuget.org/packages/EFMigrationsManagerUI/>

This package source code is hosted in below GitHub repository under MIT license. <https://github.com/naren-b/EF_MigrationsManagerUI>

**Advantages of the EFMigrationsManagerUI:**

1. With simple user interface, it is easy to manage all the pending migrations or rolling back to the previous migrations.
2. Easy to integrate with ASP.Net MVC web applications with windows and form based authentication.
3. EFMigrationsManager user interface will be helpful to manage migrations with good control over the pending migrations to be deploy to the database.
4. Always helpful to test the web application with production database copy. As part of this process, QA/UAT/Stage database will be restored periodically with production backups. Upon database restores, there is a chance of corresponding web application will break due to mismatched database context.

With the EFMigrationsManagerUI tool, upon database restore, navigate to the corresponding web application URL and publish the pending migrations with simple user interface without waiting for developer to deploy the database migrations.

1. Easy to integrate database deployments with administrative process. Upon successful deployment or swap the Stage/Production slots in azure, navigate to the web site will auto redirect the database deployment page for admin users.
2. Not everyone in the development team members have expertise on Entity Framework code first or package manager console to update/rollback migrations. For instance, some team members have expertise in client side technologies. This user interface will be helpful to development team members to manage migrations even when some of the development team member(s) are not expertise on Entity Framework code first.
3. No need to store SQL connection string in multiple places like Web.config and VSTS build/release configurations for deployment purpose.

(or)

No need to share the connection string with different team members to deploy the database migrations in various builds(qa/uat/stage/prod).

Whereas, EFMigrationManagerUI plugin reads connection string in either of the below ways

* Pass the web.config connection string name
* Pass the connection string

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

MVC web application can still use the capabilities like encrypted connection strings or read from [Key Vault](https://docs.microsoft.com/en-us/azure/architecture/multitenant-identity/key-vault).

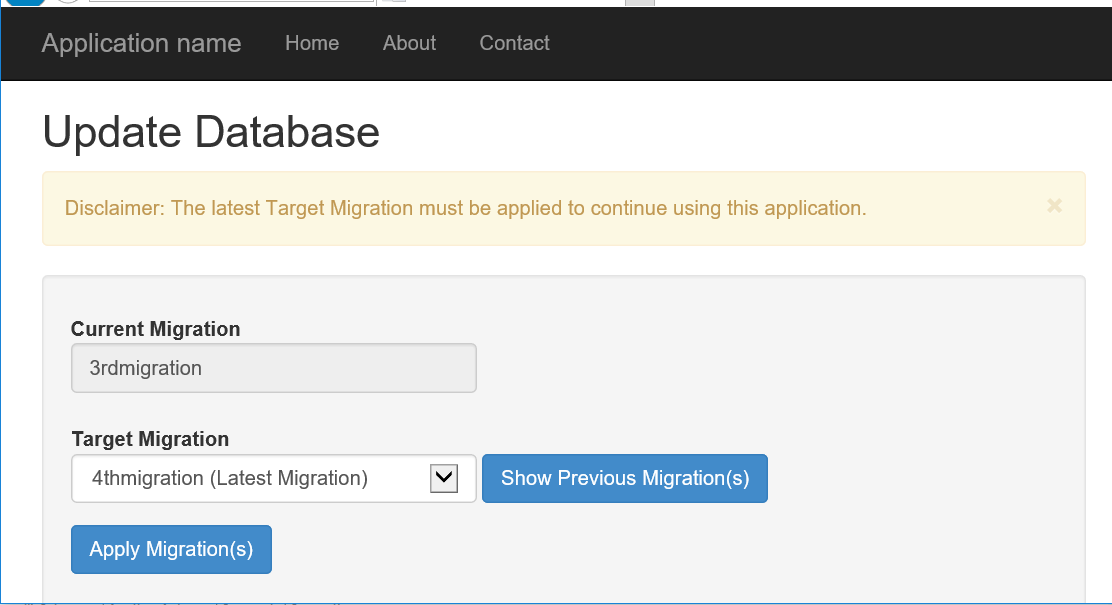
With this approach, connection string can maintain in one place like web.config, Azure Portal or Key Vault, etc.

**EFMigrationsManagerUI Package – Process Flow:**

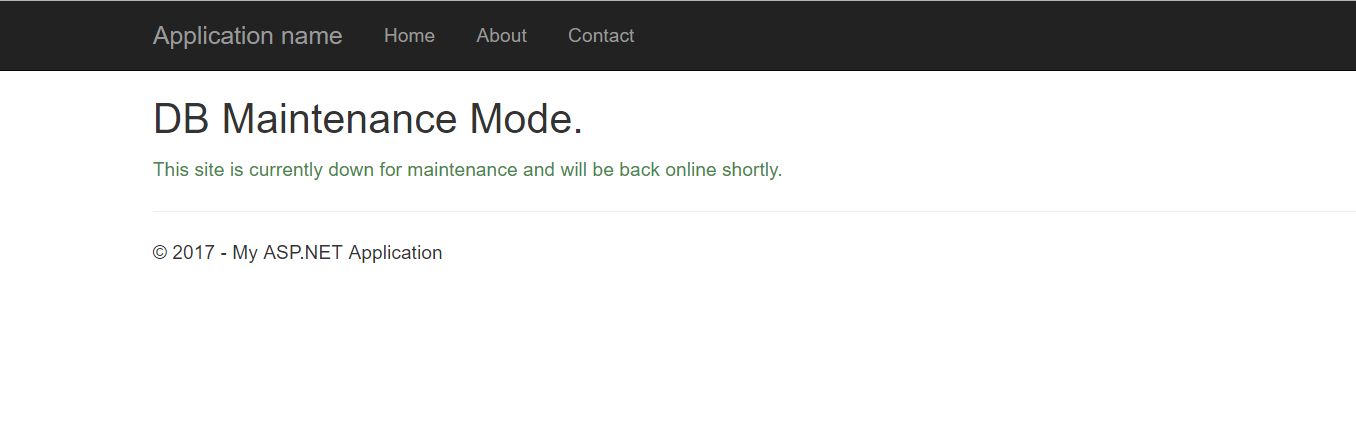
Upon successful deployment of MVC web application, an admin(administrative/QA/business) user will navigate to the web application and smoke test the applications quickly or run the unit test cases.

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 as seen in the below screen shot.



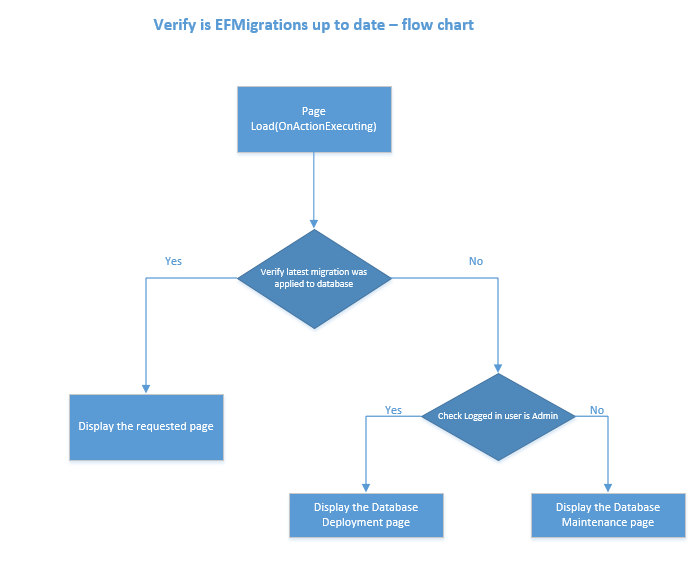
* For non-administrative users, application will redirect to the database maintenance mode page as seen in the below screen shot.



Note: Nuget package will add above views (cshtml files) to the target MVC Project. User Interface look and feel can be changed according to the project standards.

**AutoDetect Pending Migrations - Algorithm:**

By default, when install EFMigrationsManagerUI nuget package, AutoDetect feature is enabled.



**Algorithm**:

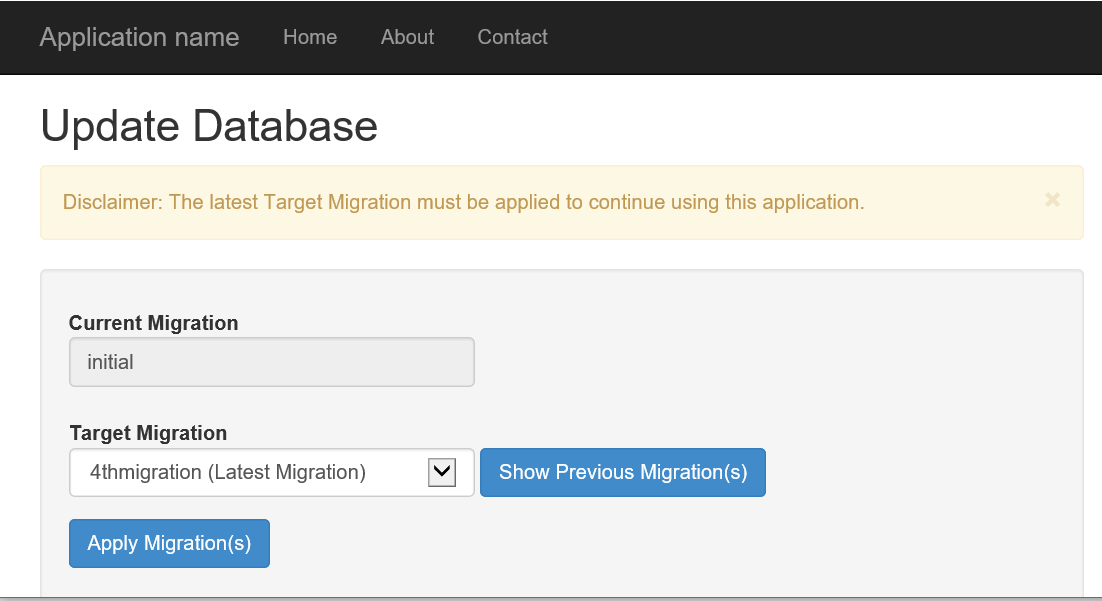
1. On page load (OnActionExecuting), application will verify that “Latest migration was applied to the database” or not.
2. If latest migration was applied then application will render the content of the user requested page.
3. If latest migration was not applied then application will check if logged in user is admin or not.
   1. If logged in user is **admin** then application will redirect to the EF manage migrations page to update the database.
   2. If logged in user is **non-admin**, then application will redirect to the database maintenance page.

**User Interface Pages - walkthrough:**

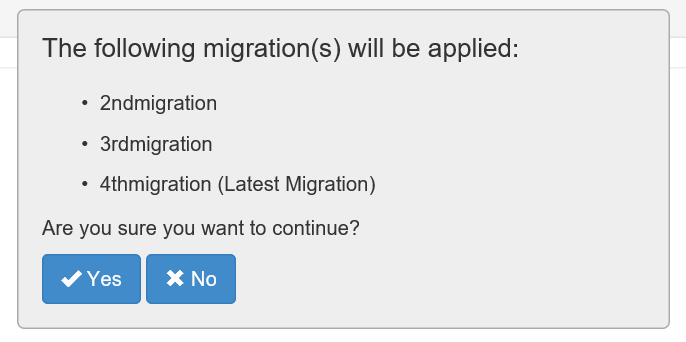
1. **Deploy Pending Migrations Page:**

If ’AutoDetect Pending Migration’ check is enabled then application will auto redirect to the below page when latest migration was not deployed to the database or user can navigate to the page with below URL.

Navigation Url: EFMigrationsManager/Publish



By default, “Target Migration” dropdown will select the Latest Migration. Once user selects the “Target Migration” and click on “Apply Migration(s)” button will show the below confirmation message and display all the migrations to be applied to the database.



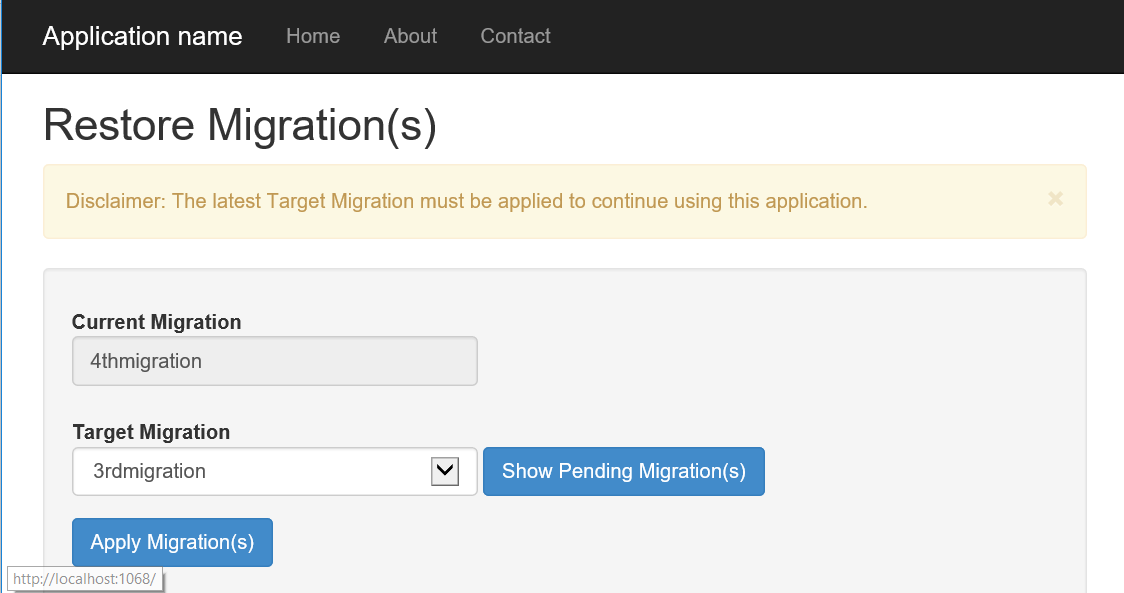
Upon clicking on “Yes” button will deploy the database with pending migration(s).

Clicking on “Show Previous Migration(s)” button will navigate to the “Rollback Migration(s)” page.

1. **Rollback Migrations Page:**

Database migration rollback can be done by clicking on the “Show Previous Migrations” button on “Update Database” page or navigating to the below URL will give the capability to roll back the database migrations.

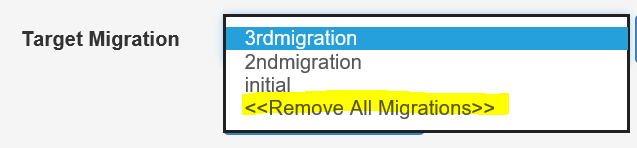
Navigation Url: EFMigrationsManager/Publish?isRollback=True



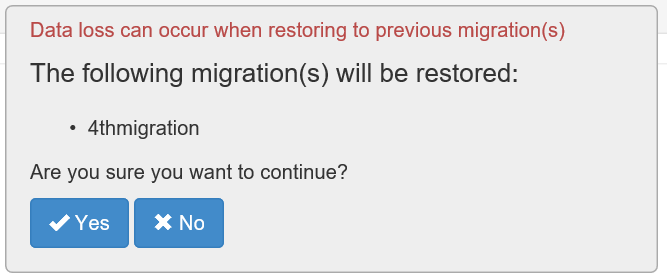
By default, “Target Migration” dropdown will select the immediate previous migration before the current migration.

Using “Remove All Migrations” is useful in the below condition(s):

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



Click on Apply Migration(s) will display the below confirmation box with data loss warning message.



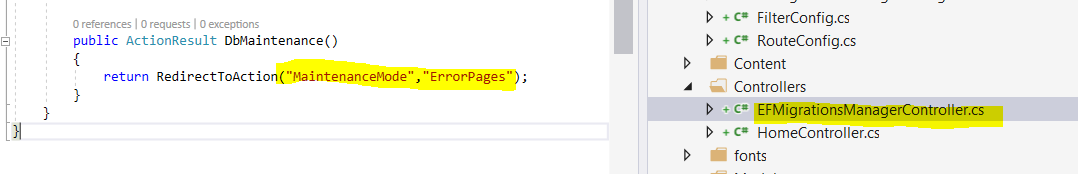
Click on “Yes” button will roll back the selected migration(s). In some cases, there is a chance for data loss when rolling back migrations.

1. **Database Maintenance page:**

This is the static page which will display when the following two conditions are satisfied:

* + Entity framework migrations are not up to date with database.
  + Logged in user is non-administrator.

Note: If the target MVC application has an existing maintenance page then requests can be redirect to the existing maintenance page from the below controller action.



**Conclusion:**

In this first article, focused mainly on EFMigrationsManagerUI nuget package, User Interface behaviors and advantages of the plugin. In the next article, we’ll cover the download and integration of EFMigrationsManagerUI nuget package with ASP.Net MVC web application and demos with Asp.Net MVC windows and forms based authentication.

**References:**

1. [Entity Framework Code First to a New Database](https://msdn.microsoft.com/en-us/library/jj193542(v=vs.113).aspx)
2. [Entity Framework Code First to an Existing Database](https://msdn.microsoft.com/en-us/library/jj200620(v=vs.113).aspx)
3. [ASP.NET MVC](https://www.asp.net/mvc)
4. [Filtering in ASP.NET MVC](https://msdn.microsoft.com/en-us/library/gg416513(VS.98).aspx)
5. [Integrating applications with Azure Active Directory](https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-integrating-applications)
6. [GitHub](https://github.com/)
7. [Nuget](https://www.nuget.org/)
8. [Nuget Package Manager in Visual Studio](https://docs.microsoft.com/en-us/nuget/guides/install-nuget#nuget-package-manager-in-visual-studio)
9. [Azure Key Vault](https://docs.microsoft.com/en-us/azure/architecture/multitenant-identity/key-vault)