**User Interface to Deploy/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 provides an easy 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

* Apply Migrations
* Rollback Migrations
* Find the
  + Current Migration version applied to database.
  + pending migrations needs to apply to the database.
* Not everyone in the dev team is expert in entity framework to manage migrations. For instance, some of the team members are experts in UI designer/ client side skills set like Angular JS. If the entity framework migrations are not sync with database then application will throw context change error. Without the Entity framework migration deployment, web application will not run in local environment then UI designer is unable to work on the web site until other team members needs to help the designer to deploy the database in local environment.

**EFMigrationsManagerUI Nuget Package:**

To handle the above scenarios, 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. Please check the below URL for **EFMigrationsManagerUI** Nuget package.

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

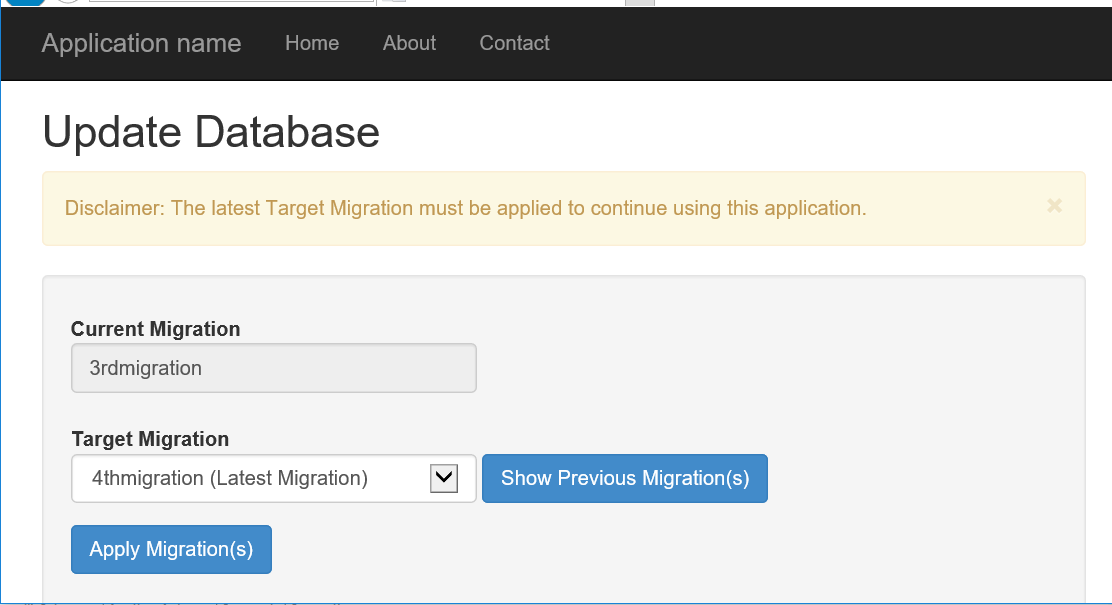
This package was hosted under MIT license and source code is hosted in below GitHub repository. <https://github.com/naren-b/EF_MigrationsManagerUI>

**EFMigrationsManagerUI Package - Behaviors:**

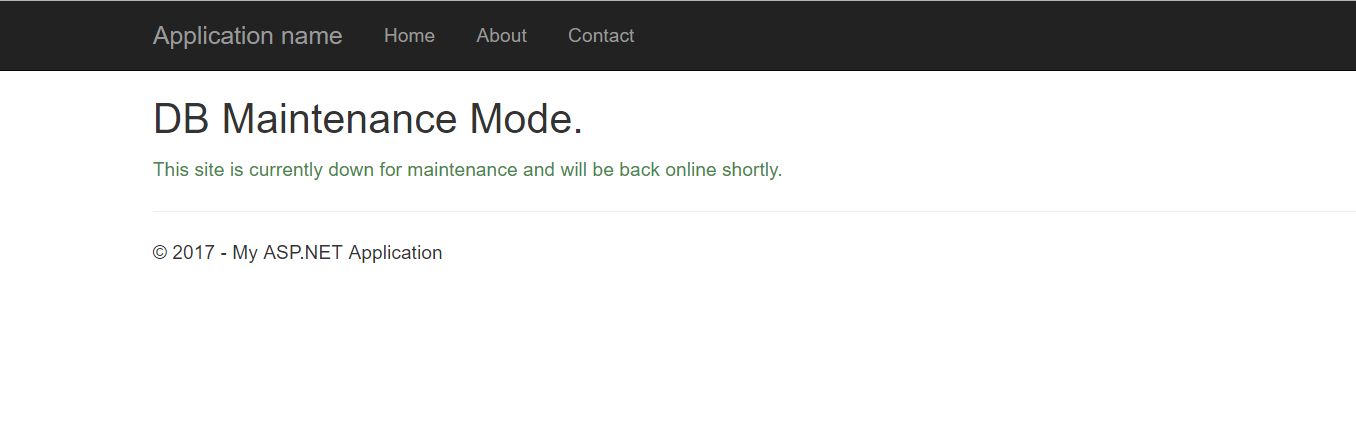
Upon successful deployment of MVC web application, Admin(administrative/QA/Business) User’s will navigate to the web application and smoke test the applications quickly or run the unit test cases.

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.



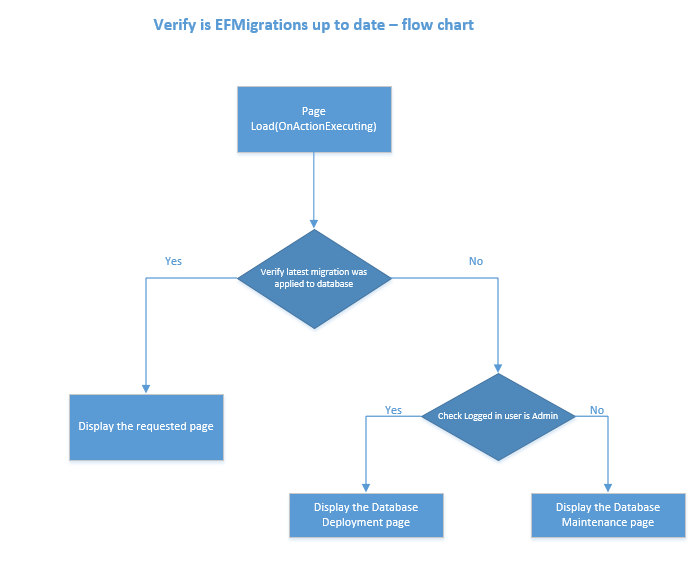
* For non-admin Users, Application will redirect to the Maintenance Mode page.



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.

**Auto Detect Pending Migrations Algorithm:**

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



**Algorithm**:

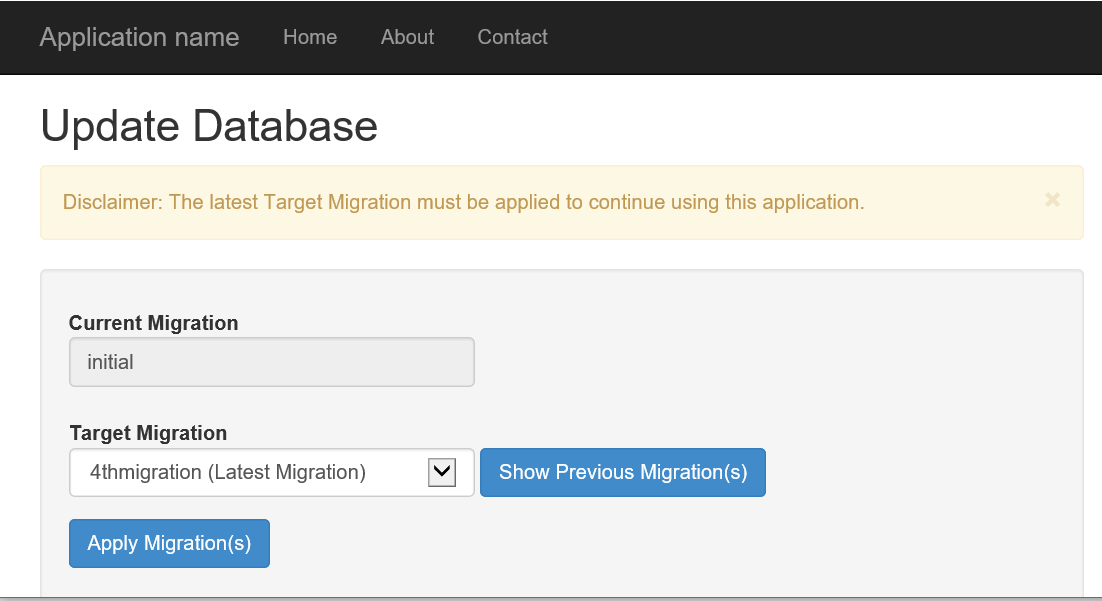
1. On page load (OnActionExecuting), Application will verify that “Latest migration was applied to the database” or not.
2. If latest migration was applied then application will render the content of the user requested page.
3. If latest migration was not applied then application will check weather logged 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.

**User Interface behaviors:**

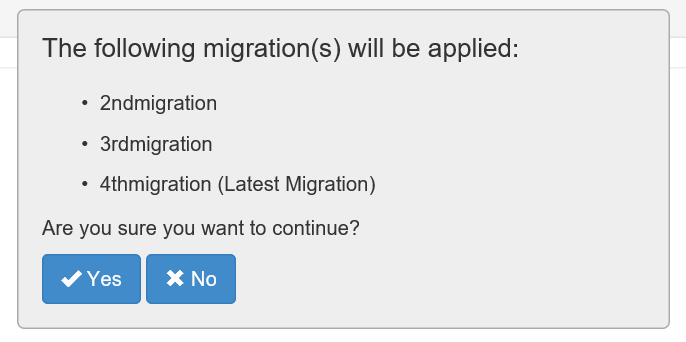
1. **Deploy Pending Migrations:**

If Auto Detect 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 Version. 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.



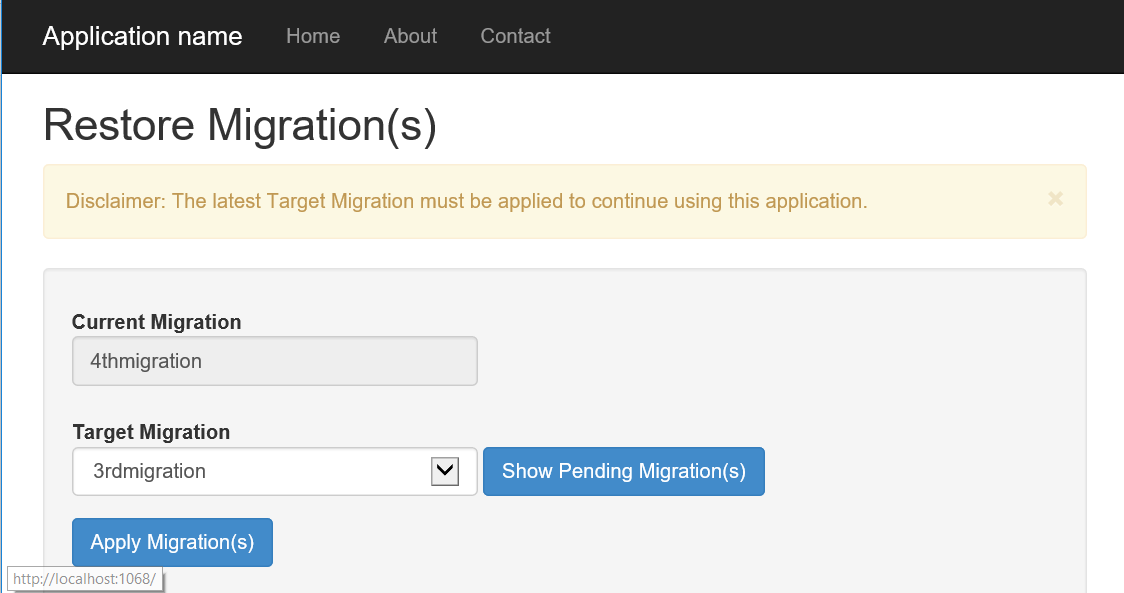
Upon click on ‘Yes’ confirmation dialog will deploy the database with pending migrations.

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

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 roll back the database migrations.

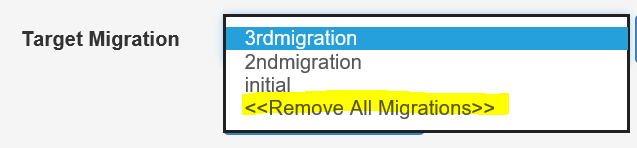
Navigation Url: EFMigrationsManager/Publish?isRollback=True



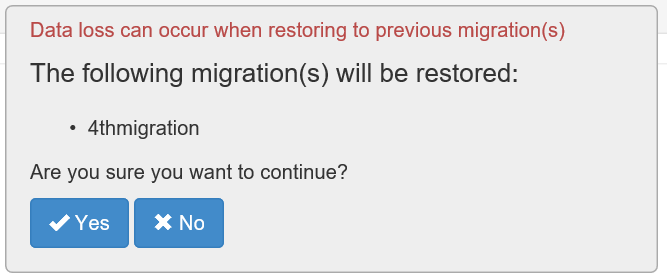
By default, Target Migration dropdown will select with immediate “Previous Migration” to the current migration.

Select “Remove All Migrations” in the below scenario(s)

* + - When application is in Initial migration state and want to 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.



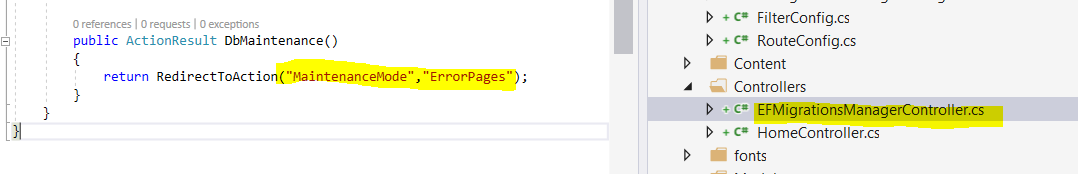
Click on “Yes” button will roll back the selected migrations. In some cases, there is a chance on data loss when roll back migrations.

1. **Database Maintenance page:**

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

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

Note: If target MVC application have maintenance page then redirect to the existing maintenance page from the below controller action.



**Advantages of the EFMigrationsManagerUI:**

1. With simple interface, Easy to apply to all the pending migrations or rollback to previous migrations.
2. Not everyone in the DEV team members have knowledge on Entity Framework Code First 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.
3. Entity framework migration can also deploy as part of the deployment or application startup. As a best practice, these automatic migrations deployments are **not** recommendable as we lose the control of migrations deployment and don’t know what migrations are deploying.

Using this EFMigrationsManagerUI user interface will be helpful to manage migrations with control.

1. Easy to integrate with DevOps/administrative process. Upon successful deployment or Upon successful swap the Stage/Prod environments in azure environment, navigate to the web site will auto redirect the database deployment page for admin users.
2. Works with ASP.Net MVC web applications with Windows and Forms based authentication.
3. 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 due to mismatched database context. With this tool, navigate to the corresponding web application URL and publish using user interface without waiting for developer to deploy the database migrations.
4. No need to store SQL connection string in multiple places like Web.config for web application database call and VSTS build/release configurations for deployment purpose.

(or)

No need to share the connection string with different members to deploy the database. Migrations.

Whereas, MigrationManager 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,

**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)