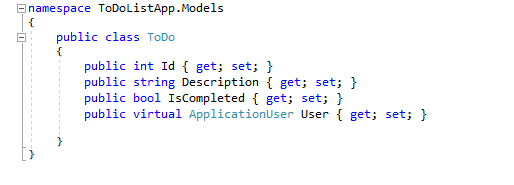
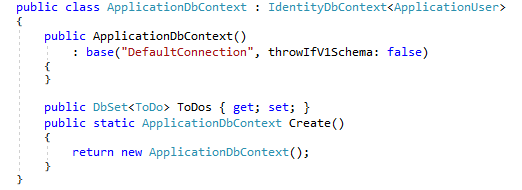
# TODO LIST APPLICATION

Walkthrough

1. The Todo application was built using ASP.NET MVC Framework in Visual studio 2017.
2. Authentication Used 🡪 Individual User Accounts (So that all the user profiles are stored on a server sort in a database which is on our server and the passwords are automatically hashed and they can also be used to sign in to external providers like Facebook, Twitter etc.)
3. Based on the requirements my (ToDo)model would need a text description to define what a to-do item is , a Boolean to let us know if the task is completed or not and each one of these is going to have a unique ID so an integer ID (Primary key in Database) and we also need a user say virtual ApplicationUser.
4. ApplicationUser here defines a user in this framework, that’s how we’re gonna link this ToDo item to a particular user.

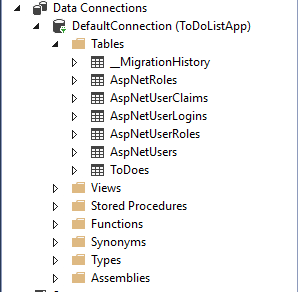


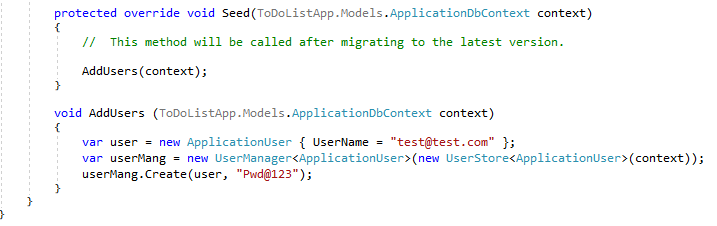
1. The next step is to tell the framework that we want this model to be part of a database and for that we need to define a collection of these items and tell it which database to put them in.
2. There are many ways to do this. But my approach here is code first approach by using the ApplicationUser the one present in Identity model which is automatically created when we select individual user authentication. This model contains ApplicationDBcontext which will be referring to data access layer.
3. So there will be bunch of tables needed for user in identity system and to a table for Todo’s we can do that by adding DBSet. By adding this a collection of todo’s that are in a table in this database(ApplicationDBContext)



1. The base here is telling us what connection or where this Database is ? So if we had our own database we can insert it in here. But in this case I leave it as defaultconnection. Hence the EntityFramework will make sure if there isn’t a database already it would create one by looking through all these classes and relationships between them.
2. Here I am using Migration system so which needs to be enabled through Nuget through Package manager.

* Enable-Migrations 🡪 it looks at what databases are in our application and helps us to start seeding our database when we create it.
* Add-Migration Todo1 🡪 It basically generates a class or a function that defines this migration. It looks at what database was set up previously and what its set up now. So creates tables using the models and creates a primary key and foreign key.
* Update-database 🡪 To create the above created tables



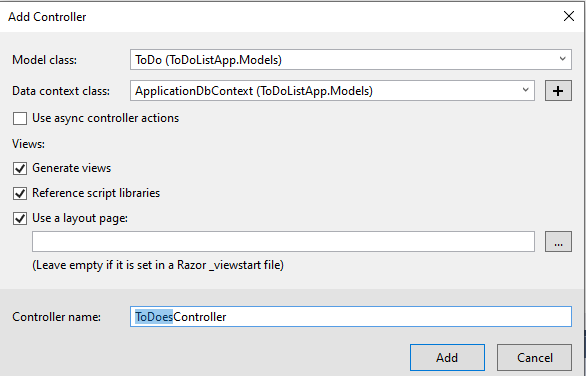


1. So as we can see I have seeded a user with

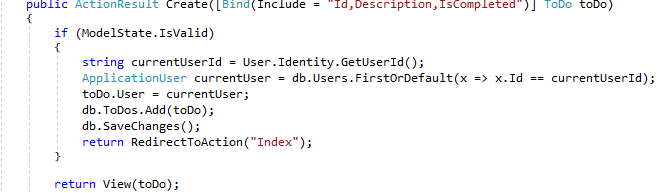
username 🡪 [test@test.com](mailto:test@test.com)

password 🡪 Pwd@123

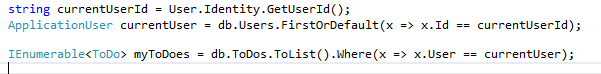
1. And all of these are coming from the Account Controller which was made default by the system
2. Next we need a controller and for this I am using controller with Entity Framework.



1. The controller contains a file for each of these methods Index, Details, Delete, Edit and each of these have a view like Create view, Delete view etc.
2. The nice thing about this framework is it does model binding for us to without even to worry about the models. But what isn’t happening is that there isn’t any link between this Todo and the user that is creating it.



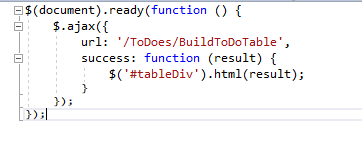
1. To just see or get the task or the list a user has created



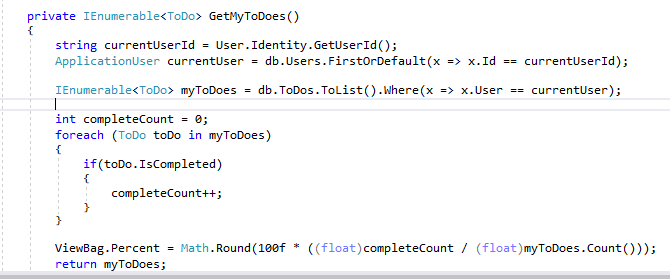
1. At the start of the application we have predefined buttons in the menu bar so to change the name of the application and also change the name .



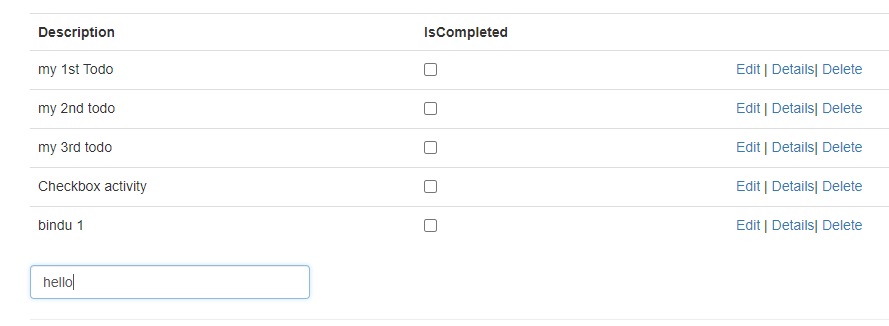
1. We need to write the Javascript for this . As soon as the above page has finished loading and then we have empty Div and then we want our javascript to call build the do table get the HTML that’s generated and then put it in the div that’s already present we are doing this by using Ajax helpers and Jquery.



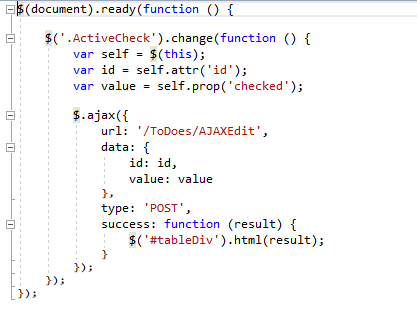
1. No to add the progress bar and also to get my tables we need to create a method so we can use them repeatedly if we want.



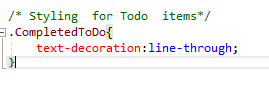
1. So for the post request after we enter a description the page use to return to the index page create form and to avoid this we can add a text box below the list so we can add the task without leaving the current page and make amends like if the task is completed or not and if the task needs to be deleted or removed

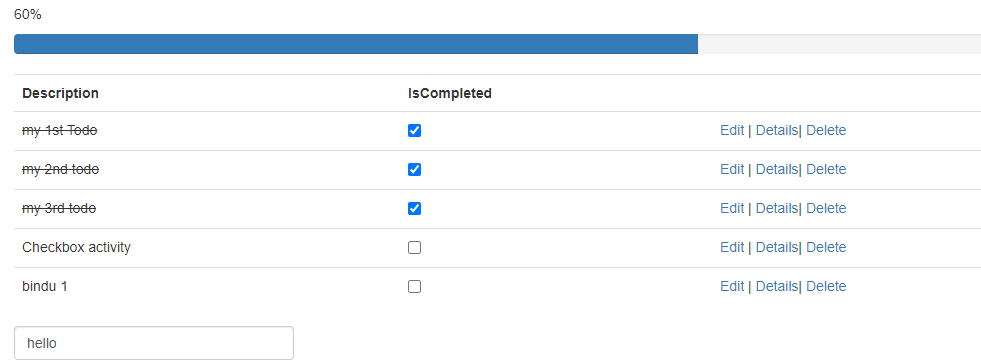


1. Now to makes these checkbox active and individual to each todo description we can add another JavaScript with the help of jQuery.

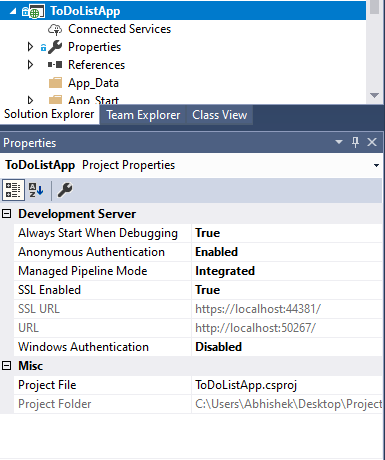


1. Now to have progress bar to show how much is completed or done and also just to show an appearance that the task is completed by striking it out is done by using Site.Css file.

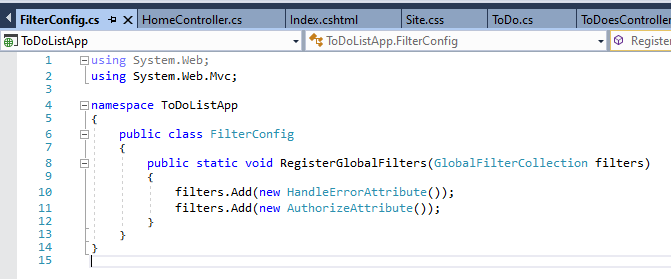




1. And for security purpose it is always better to use HTTPS instead of HTTP and this can be done by changing the value of SSL Enabled to **True** and also by checking the AntiForgeryToken.



1. Last but not least it is import for us to make sure the logged in user is only capable of seeing the tasks so its essential to make the user login first before allowing to interact with the application and to do this we can Authorize the methods which needs to be accessed only by logged in user. But finding all the methods and then authorize the method would be time consuming hence under filterConfig we can add a new attribute to perform authorization.



1. But the user should atleast be able to see the index page without being able to login hence we can use [allowanonymous] for the HomeController.

