**AltaSoft Mvc CRUD Scaffolder**

V 2.0

Read Me

**მოკლე აღწერა**

AltaSoft Mvc CRUD Scaffolder - დაგვეხმარება ASP.NET MVC-ს ტიპის პროექტებში, სტანდარტული CRUD ოპერაციებისთვის View-ების და Controller-ის გენერაციაში. ანალიზის შედეგად გამოიკვეთა რომ CRUD ოპერაციის 85% არის რუტინული და გვიწევს ერთიდაიგივეს კეთება. Scaffolder-ის მიზანია დეველოპერის წარმადობა მაქსიმალურად გაზარდოს და რუტინული საქმე მთლიანად მოაშოროს მას. Scaffolder-ს შეუძლია მუშაობა ნებისმიერ ობიექტთან, არაა აუცილებელი რომ იყოს Entity Framework-ის Model-ი.

პროექტში გენერაციის შედეგად ვიღებთ შემდეგ ობიექტებს (მაგალთისთვის ვაგენერირებთ Customer ობიექტისთვის CRUD-ს):

* Controllers/CustomerController
* Views/Customer/DisplayTemplates/ListFilter.cshtml
* Views/Customer/\_ListItems.cshtml
* Views/Customer/Create.cshtml
* Views/Customer/Delete.cshtml
* Views/Customer/Details.cshtml
* Views/Customer/Edit.cshtml
* Views/Customer/List.cshtml

**უპირატესობები**

მოკლე ჩამონათვალი იმ უპირატესობების რასაც Scaffolder გვაძლევს, შემდგომ უფრო დეტალურად გავივლით თითოეულ მათგანს:

* Paging
* Filters
* List Columns Configuration
* Fast Controller Implementation
* Filters Management
* Search Feature
* One2Many Feature

**წინასწარი საჭიროებები**

Scaffolder-ი არის დამოუკიდებელი ყველა პროექტისგან და შესაძლებელია მისი გამოყენება ნებისმიერ პროექტში.

წინასწარი საჭიროება რაც არსებობს ესენია:

* „ნადიას სტილების“ არსებობა პროექტში, რათა კორექტულად დაიხატოს View-ები.
* Altasoft.js ფაილის ჩასმა პროექტში და Layout ფაილში ჩატვირთვა.

**Views**

ყველა დაგენერირებული View წარმოადგენს მარტივ ფორმას, სადაც გამოტანილია გარკვეული რაოდენობის ველები, ფორმა და Submit ღილაკი.

ერთადერთი შედარებით კომპლექსური View არის List.cshtml სადაც დატანილია რამდენიმე ფუნქციონალი. მისი მუშაობის პრინციპი შემდეგია:

* მთელი List გატანილია Partial View-ში
* Paging-ის და ფილტრაციის დროს ხდება ამ Partial View-ს განახლება

ასევე ამ View-ზე ხდება ფილტრაციის ელემენტების გამოტანა DisplayTemplate-ს გამოყენებით.

**უპირატესობები - Paging**

List View-ში ავტომატურად არსებობს Paging მექანიზმი, ListItems Action-ზე ხდება skip და limit-ის გადაწოდება და ასე ხდება გაკონტროლება. ეს მხოლოდ ინფორმაციულად, წარმოდგენა რომ გვქონდეს, ისე ხელთ არაფრის გაკეთება აქ არ არის საჭირო.

მის ჩასართავად საჭიროა JS-ში შემდეგი მეთოდის გამოძახება (default-ად ჩართულია):

AltaSoft.General.ConfigureListPaging('#ItemsList', '#ListFilterForm');

**უპირატესობები - Filters**

სურვილის შემთხვევაში შეგვიძლია List View-ს დავუგენერიროთ ფილტრები, ეს „ნავაროტკა“ არის Optional. გენერაციისას გადავცემთ ფილტრის მოდელს და ავტომატურად ხდება შესაბამისი DisplayTemplate-ს შექმნა და ჩასმა.

მის ჩასართავად საჭიროა JS-ში შემდეგი მეთოდის გამოძახება (default-ად ჩართულია თუ ავირჩევთ ფილტრის მოდელს):

AltaSoft.General.ConfigureListFilters('#ListFilterFormPanel', '#ItemsList');

**უპირატესობები - List Columns Configuration**

List View-ში შესაძლებელია კონკრეტული Column-ების დამალვა ან გამოჩენა, User-ის სურვილიდან გამომდინარე, ინფორმაცია ინახება Client-side cookie-ებში

მის ჩასართავად საჭიროა JS-ში შემდეგი მეთოდის გამოძახება (default-ად ჩართულია):

AltaSoft.General.ConfigureListColumnsVisibility('#ColumnModal', '#ItemsList');

**უპირატესობები - Fast Controller Implementation**

კონტროლერის გენერაციისას საჭიროა იმპლემენტაცია გავაკეთოთ შემდეგი მეთოდების:

* GetItems
* GetItemById
* [HttpPost] Create
* [HttpPost] Edit
* [HttpPost] Delete

სანამ კონტროლერის იმპლემენტაციას გავაკეთებთ, პროექტი იბლიდება ჩვეულებრივ.

კონტროლერში დასაიმპლემენტირებელ მეთოდებში გამზადებულია მზა კოდები კომენტარის სახით, რომელიც თავიდან აგაცილებთ რუტინული კოდის დაწერას და მხოლოდ შესაბამისი Manager კლასების მინიჭებაა საჭირო

**უპირატესობები - Filters Management**

ეს ფუნქციონალი მოსახერხებელს ხდის სხვადასხვა ფილტრების შენახვას კონკრეტული მომხმარებლისთვის და შემდგომ მათ ჩატვირთვას.

ფილტრები მომხმარებელს შეუძლია შეინახოს ორ მდგომარეობაში

* Individual - მხოლოდ თვითონ ხედავს
* Common - საერთო, ანუ ყველა ხედავს

ეს ფუნქციონალი არის Optional და შესაძლებელია საჭიროებისამებრ ჩართვა.

მისი იმპლემენტაციისთვის საჭიროა:

* ან კონტროლერში დაგენერირებულუი ფილტრის მეთოდების იმპლემენტაცია (ყველა მეთოდი მოქცეული იქნება Filters Management რეგიონში)
* ან გაკეთდეს ერთი ცენტრაილიზირებული კონტროლერი ყველა ფორმისთვის და შესაბამისი Action-ები გადამისამართდეს ამ კონტროლერზე, დაგენერირებული Filters Management რეგიონი კი წაიშალოს ჩვენი კონტროლერიდან.

**უპირატესობები - Search Feature**

როდესაც რომელიმე სხვა ფორმაში დაგჭირდებათ გენერირებული მოდელის ობიექტის მოძებნა, დაგეხმარებათ Search Feature. პწიჩკის მონიშვნის შემთხვევაში თქვენ დამატებით დაგიგენერირდებათ ორი Partial View (\_Search, \_SearchListItems) და ერთი Action (Search), რის შემდეგაც გახდება მარტივად გასაკეთებელი Search Box-ები სხვადასხვა ფორმაზე.

ეს ფუნქციონალი არის Optional და შესაძლებელია საჭიროებისამებრ ჩართვა.

მისი იმპლემენტაციისთვის საჭიროა:

იმ ფორმაში სადაც საჭიროა Search Box, უნდა ჩავსვათ შესაბამისი სკრიპტის გამოძახება, MeterPoint-ების შემთხვევაში იქნება:

AltaSoft.General.ConfigureSearchItem('#SearchMeterPointModule', '#SearchMeterPointResult', '#MeterPoint\_AdminNumber', 'MeterPointNumber');

**უპირატესობები - One2Many Feature**

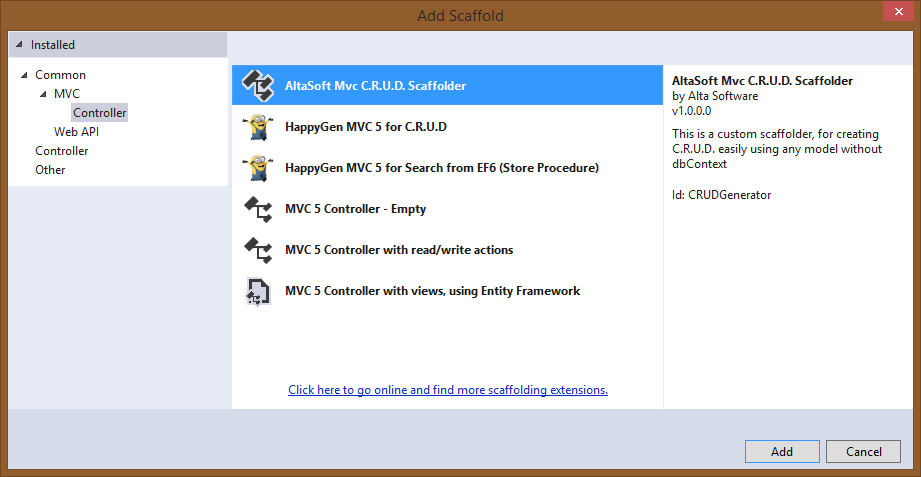
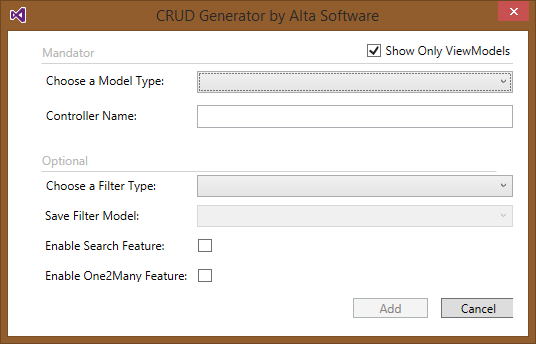
როდესაც საჭიროა რომელიმე სხვა ფორმაში, გენერირებული მოდელის ერთი ბევრთან გაფორმება ვიზუალურად, ეს ფუნქციონალი დაგეხმარებათ ამ ამოცანის მარტივად გადასაწყვეტად. ავტომატურად დაგენერირდება მოდელის შესაბამისი Partial View (\_IdList) და Action (CheckByAdminNumber), რომელიც უზრუნველყოფს მონაცემების ვალიდაციას, სანამ მოხდება დამატება.

ეს ფუნქციონალი არის Optional და შესაძლებელია საჭიროებისამებრ ჩართვა.

მისი იმპლემენტაციისთვის საჭიროა:

**როგორ ხდება გამოყენება**

Scaffolder-ის გამოყენებისთვის საჭიროა გავიაროთ სამი მარტივი ეტაპი:

1. ვაყენებთ Extensions მისამართიდან: [\\iisdev\NuGet\Extensions](file:///\\iisdev\NuGet\Extensions)
2. ახალი კონტროლერის დამატებისას ვირჩევთ „AltaSoft Mvc C.R.U.D. Scaffolder“-ს  
   
3. ვირჩევთ მოდელს და თუ გვჭირდება ფილტრის შესაბამის კლასს:  
   

**ცვლილებები**

საჭიროების შემთხვევაში შეგიძლიათ შეცვალოთ დაგენერირებული კოდი, როგორც ამოცანას სჭირდება, არაა აუცილებელი რომ ერთი ერთში ისე იყოს იმპლემენტირებული კონტროლერი როგორც დაგენერირდა, თუმცა სასურველია რომ ცვლილებები იყოს მინიმალური, ამოცანიდან გამომდინარე.

**Helpers**

თუ გვსურს რომ მოდელში კონკრეტულ ველზე არ მოხდეს Scaffolding, შეგვიძლია მას დავადოთ შემდეგი ატრიბუტი და ეს ველი დაიგნორდება:

[ScaffoldColumn(false)]

მუშაობს როგორც მთავარ მოდელზე, ასევე ფილტრზე.

Happy scaffolding :)