Pro asp net mvc5 pdf

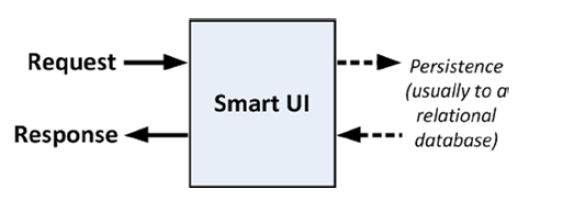
Da 0 a 40 esempio di come creare un form e validarlo con MVC

Da pagina 41 si parla di come applicare gli stili.

I pattern di architettura (da pag 53)

Smart UI

Interfaccia ad eventi con codice che corrisponde ad eventi, risultato applicazione monolitica. Il codice che serve l’interfaccia ed il codice business è fuso assieme.



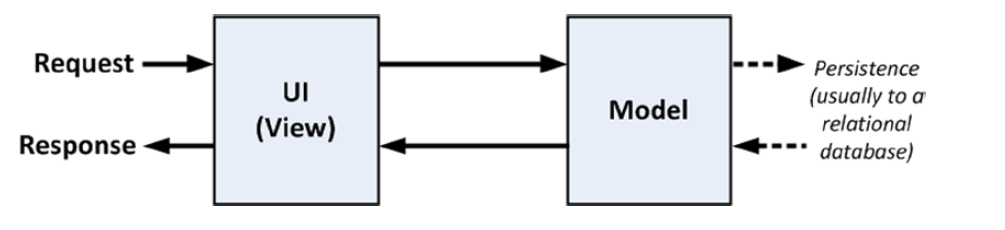
Una interfaccia del genere è utile per semplici progetti, si può sviluppare velocemente, ma difficile da mantenere ed estendere, ed è difficile anche testarla. Molto codice duplicato con copia incolla.

Non bisogna rigettare le smart UI perché sono veloci e facili da sviluppare.

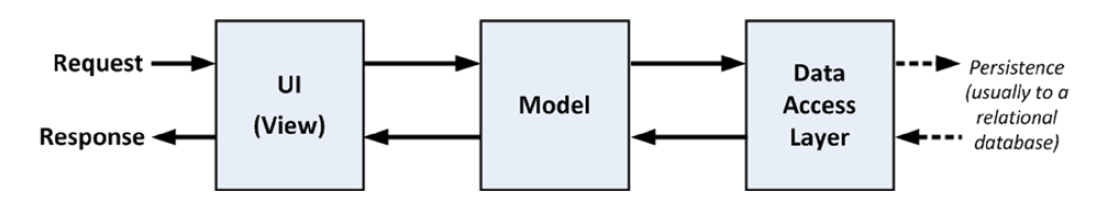
Per produrre un piccolo tool per poche persone la Smart UI è la soluzione perfetta.

Model view architettura.

Qui un piccolo miglioramento viene fatto separando codice business ed accesso ai dati (che vengono posti nel model) dall’interfaccia infatti i problemi maggiori di mantenimento vengono riscontrati nella logica business.



Three tier architectures



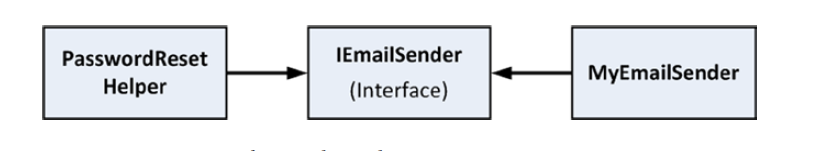
Questo è il pattern architetturale più usato.

Provvede ad una buona separazione senza divenire troppo complicato.

La differenza rispettoad un pattern MVC è che la UI è direttamante accoppiata mediante le routine di gestione eventi e diventa veramente difficile pianificare degli unit test.

Pattern MVC

Ogni componente non dovrebbe sapere nulla dell’altro e si interfaccia con le altre parti mediante interfacce astratte

Ad esempio 

Non c’è dipendenza fra l’helper e la classe myemailsender, potrei unsare uma implementazione mok per testarla

La dependecy injection permette il disaccoppiamento completo, le istanze vengono risolte a runtime da una classe container. Il ruolo di un buon container dovrebbe essere quello di:

1 risolvere a catena le dipendenze: se una dipendenza ne usa un’altra vengono istanziate tutte le dipendenze.

2 Gestione del ciclo di vita:

dovrebbe poter specificare se occorre istanziare:

singleton (la stessa istanza viene usata)

transient (una nuova istanza ogni volta)

instance per thread, per http request, instanza da un pool ecc…

3 Configurazione dei parametri: la possibilità di passare dei paramentri al costruttore.

Tipi di test:

Unit test: isolare e testare il comportamento di una singola classe o routine isolata dalla applicazione.

Integration testing:

il testare il comportamento di più componenti che lavorano insieme fino all’intera applicazione web.

Da pagina 61 spiega come fare a testare.

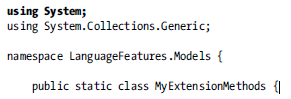
Unit test ed integration test…

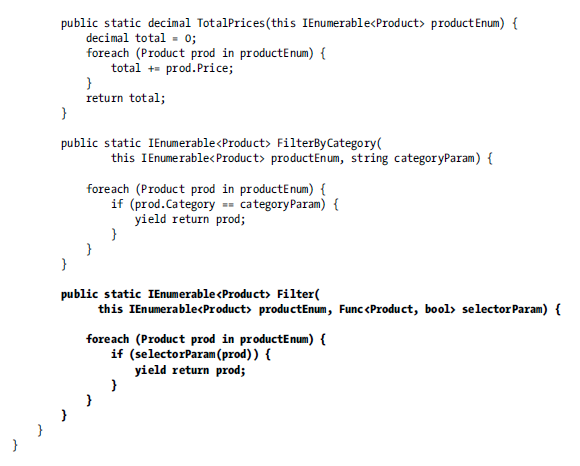
Guardare ienumerable interfaccia e yeald su c# bibbia.

Da pag 67 sintassi di base

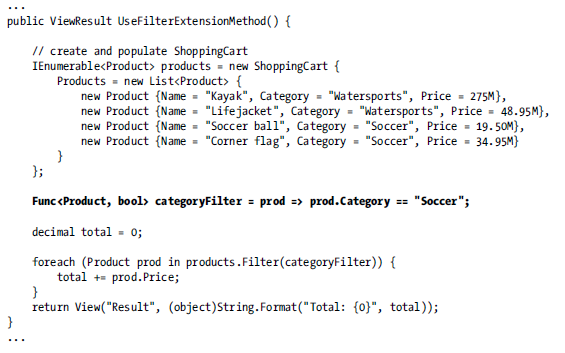
Arrivato a pag 80

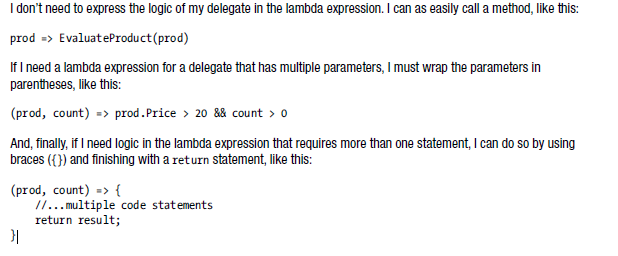
A pagina 81 esempio di come una func può essere passata ed invocata:





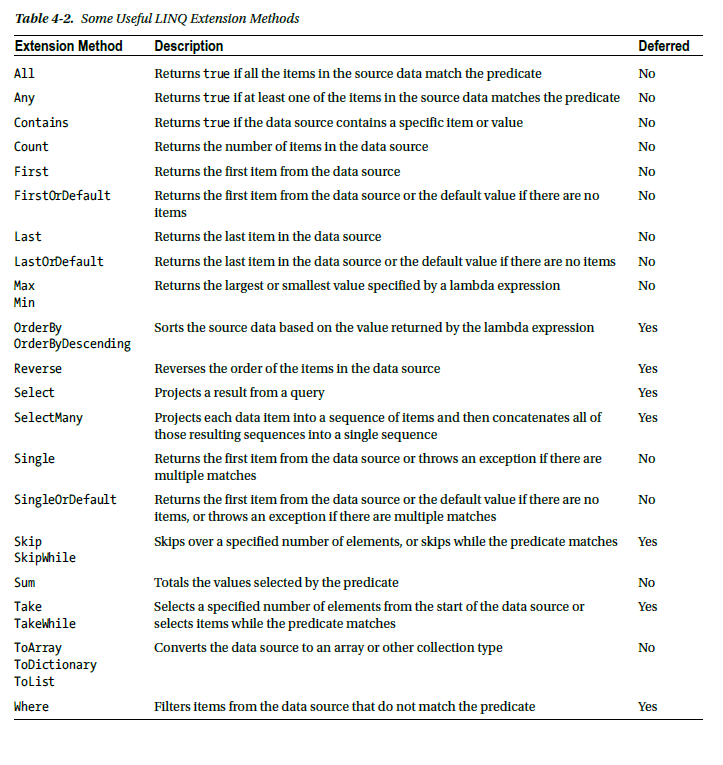
Utilizzo del codice sopra:



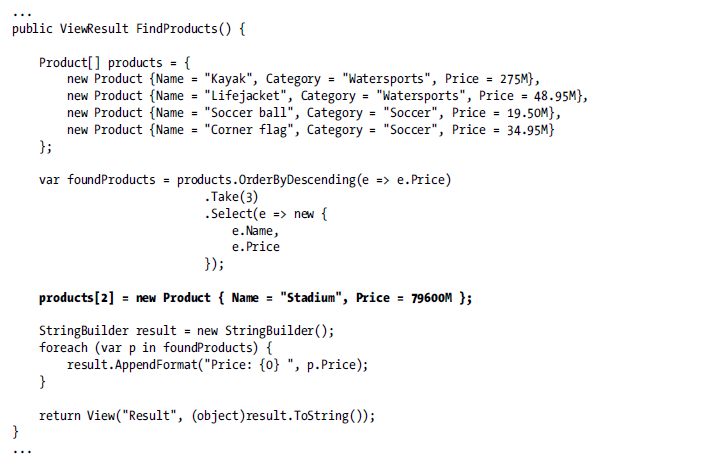


Linq a pag 89:

ogni extenasion method qui sotto aplica un inumerable e ritorna un inumerable questo significa che si può concatenarli per ottenere query complesse.



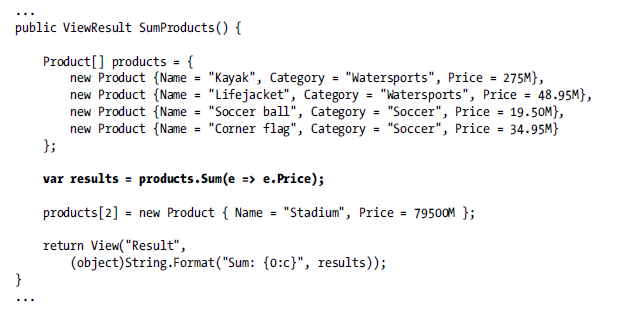
Esempio di deferred method (vedi colonna deferred tabella su)



Il risultato prende in considerazione l’aggiunta di stadium in quanto la query viene eseguita sul for 8quando effettivamente richiede l’enumerazione)



Quest’altro esempio invece non è deferred:



Infatti la somma non tiene presdente di stadium aggiunto dopo in quanto la query viene eseguita su sum

Razor sintax a partire da pagina

Aggiungere layout nel viewstart file:

I can resolve this by using a *view start file*. When it renders a view, the MVC framework will look for a file called

\_ViewStart.cshtml. The contents of this file will be treated as though they were contained in the view file itself and

I can use this feature to automatically set a value for the Layout property.

Inserimento di testo o html dentro codice razor

Inside of the Razor code block, you can include HTML elements and data values into the view output just by

defining the HTML and Razor expressions, like this:

...

<b>Low Stock (@ViewBag.ProductCount)</b>

However, if you want to insert literal text into the view when it is not

contained in an HTML element, then you need to give Razor a helping hand and prefix the line like this:

...

@: Out of Stock

Da pagina 119 esseziali tools per MVC:

dependency injection (DI) container, a unit test framework, and a mocking tool

due passaggi:

linterfaccia dovrebbe essere risolta creando una istanza di LinqvalueCalculator:

ninjectKernel.Bind<IValueCalculator>().To<LinqValueCalculator>();

qui crea l’istanza effettivamente, quindi la prima istruzione è una mappatura mentre la seconda l’esecuzione

IValueCalculator calc = ninjectKernel.Get<IValueCalculator>();

Per disaccoppiare completamente le classi dai controller occorre creare un dependencyresolver che è una classe che implementa l’interfaccia IDependencyResolver

Si dimostra poi la dependency injection:

* Come creare una catena di dipendenze che risolvono
* Come valorizzare delle proprietà pubbliche in fase di risoluzione dell’interfaccia nel dependency resolver:  
  kernel.Bind<IDiscountHelper>().To<DefaultDiscountHelper>().WithPropertyValue("DiscountSize", 50M);
* Come usare il binding condizionale (ad esempio iniettare una certa classe solo se la classe in cui viene iniettata è una certa classe):  
  kernel.Bind<IDiscountHelper>().To<FlexibleDiscountHelper>().WhenInjectedInto<LinqValueCalculator>();
* Come settare lo scope in fase di istanza dell’oggetto (in base al thread, alla richiesta http, alla sessione ecc….)

**Unit test**

È facile organizzare test che dipendono da una sola classe, occorre però testare nella realtà oggetti non isolati. Il moc simula le funzionalità di un vero oggetto. Il moc ti permette di focalizzare l’attenzione su quello che ti interessa, verrà utilizzata la libreria moq (ce ne sono tante ma questa è semplice).

Install-Package Moq -version 4.1.1309.1617 -projectname EssentialTools.Tests

Adding a mock object to a unit test means telling Moq what kind of object you want to work with, configuring its

behavior and then applying the object to the test target

fare riassunto fino a qui

Quando devi testare un metodo di una classe dove la classe dipende da un’altra classe ad esempio un manager che ha un metodo che utilizza un repository, devi creare il moc repository che simula la repository stessa definendo cosa verrà passato in entrata e cosa restituirà questo moc e poi utilizzare il medesimo nel test

Pag 153 si parte con esempio progetto reale.

Prima si creano i progetti vuoti, si aggiungono i pacchetti nugets, si fissano le dipendenza fra progetti, si imposta l’infrastruttura per la dependency injection.

Pag 169 Poi viene creato il db

Da pagina 178 viene dimostrato come creare un paginatore:

la lista dei prodotti viene fornita in base alla pagina richiesta con skip take di linq mentre il controllo in basso per passare da pagina a pagina viene creato con un modo interessante: con un estension method dei controlli html:

public static MvcHtmlString PageLinks(this HtmlHelper html,

PagingInfo pagingInfo,

Func<int, string> pageUrl) {

StringBuilder result = new StringBuilder();

for (int i = 1; i <= pagingInfo.TotalPages; i++) {

//vengono creati nuovi tag col tagbuilder...

TagBuilder tag = new TagBuilder("a");

//qui viene chiamata la callback:

tag.MergeAttribute("href", pageUrl(i));

tag.InnerHtml = i.ToString();

if (i == pagingInfo.CurrentPage) {

tag.AddCssClass("selected");

tag.AddCssClass("btn-primary");

}

tag.AddCssClass("btn btn-default");

result.Append(tag.ToString());

}

return MvcHtmlString.Create(result.ToString());

}

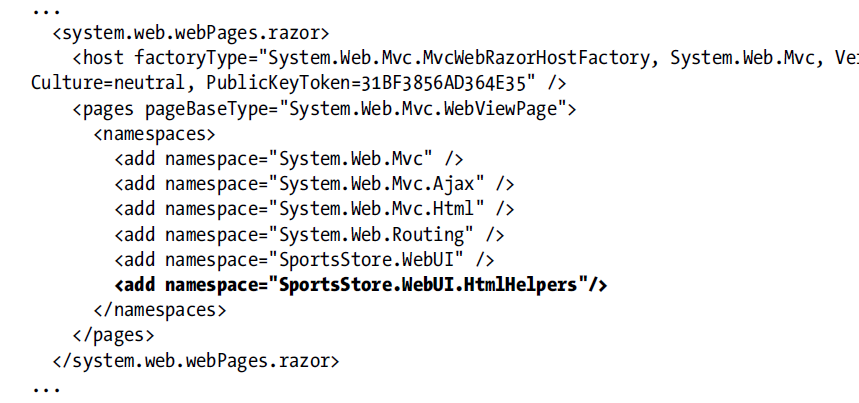
Qui come viene chiamato sulla vies:

<!--questo è un estension method che viene definito in paginghelper....-->

@Html.PageLinks(Model.PagingInfo, x => Url.Action("List", new { page = x }))

Interessante come viene creato l’estension method e come viene passata la funzione anonima per generare il link alla pagina.

Affinchè l’estension method sia visibile in razor occorre aggiungerlo nel webconfig:



Fino ad ora abbiamo ottenuto due grossi risultati:

abbiamo disaccoppiato le repository e quindi i dati dai controller, e poi abbiamo creato una applicazione che riusciamo facilmente a testare.

Da pag 190 a 193 si parla di come introdurre bootstrap nell’applicazione.

arrivato a pagina 193

Parzial view

The ***Html.ActionLink*** renders the hyperlink tag to the specified controller action which uses the Routing API internally to generate URL.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | @Html.ActionLink("Abundant Code on C# and Java Samples", "ActionName","ControllerName")  he ***Html.RouteLink*** is similar to the ActionLink but accepts a parameter for route name and does not include the parameters for Controller name and action name.   |  |  | | --- | --- | | 1 | Eg : @Html.RouteLink("Abundant Code on C# and Java Samples", new { action= "ActionName"})  Url.Action( …. Genera proprio l’url e non il link…. | |

pag. 219 - Esempio di aggiunta bottone mediante form (tanti form nella pagina per ogni riga della griglia che comportano il submit

RedirectToAction

This has the effect of sending an HTTP redirect instruction to the client browser, asking

the browser to request a new URL.

Pag 227 si parla del model binder: il modelbinder è una caratteristica di MVC che si occupa di come creare degli oggetti disponibili nella richiesta e passarli come argomenti in entrata alla action.

A pag 227 viene creato un custom model binder per il carrello affichè possa essere passato al controller cart valorizzato.

A pagina 235 viene spiegato come caricare una vista parziale da una action che ritorna una partialview

SPORTS STORE</a>

Nel seguente codice non viene passato il model dalla view parente quindi il model si trova dentro la action che sottende la view

@Html.Action("Summary", "Cart") (questo chiama una action che ritorna una partial view

In questo caso viene passato il model dalla view parente

@Html.Partial("ProductSummary", p) questo chiama

Arrivato a pagina 236

Pag 237: Esempio di form e datanotation per la validazione e postback con binding automatico dei dati del form e gestione del modelerror e del ModelState.IsValid

:ModelState.AddModelError("", "Sorry, your cart is empty!");

Arrivato a pag 254 Mobile version:

MVC può aiutarti nello sviluppo mobile ma devi scegliere una di queste strategie:

Non fare nulla ed adattare qualcosina

Usare la responsività (ma per adattare a dispositivi mobile il fare nulla è impossibile)

I have used the **visible-xs and hidden-xs** (sotto I 768 px)

The \*-sm classes work on windows wider

than 768 pixels, the \*-md classes work on windows wider than 992 pixels, and the \*-lg classes work on windows wider

than 1200 pixels.

Responsive CSS features like the ones that Bootstrap provide are based on the size of the browser window,

not the device screen.

Per le griglie responsive vedere gli esempi qui:

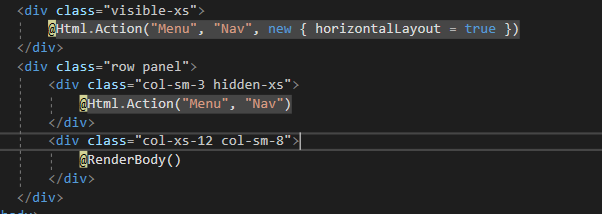
<https://www.w3schools.com/bootstrap4/bootstrap_grid_examples.asp>

There is an alternative philosophy called *mobile first* design/development which, as the name suggests, starts

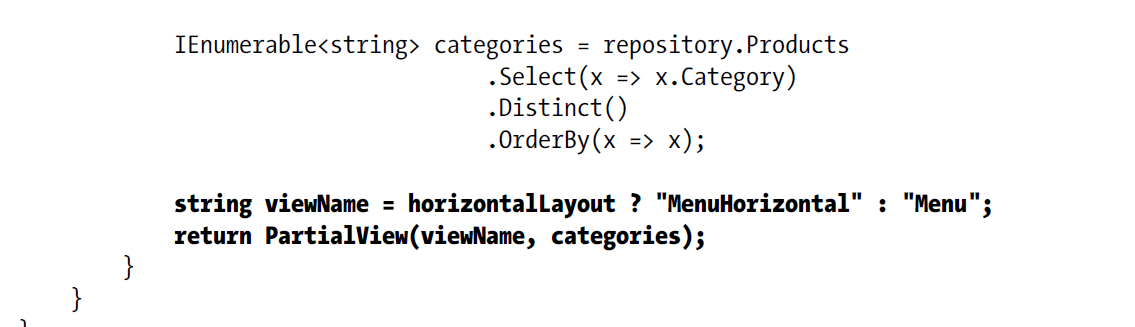
with the mobile client as the foundation for the application and adds features to take advantage of more capable

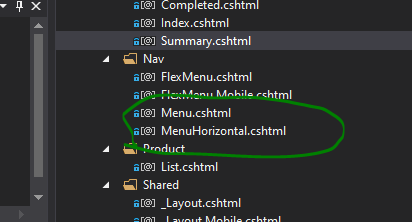
desktop browsers.

Interessante come viene gestito il caricamento dinamico delle partial view con bootstrap responsivo:



E qui come viene ritornta la view nel controller:

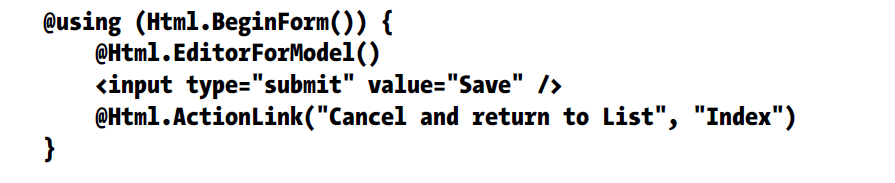




A pagina 270 viene dimostrato come creare un layout alternativo usando jquerymobile, questo tipo di funzionalità si può utilizzare quando cambia completamente il layout da desktop a mobile nell’aspetto, semplicemente le pagine con .mobile (es FlexMenu.Mobile.vhtml) verranno caricate se la richiesta perviene da dispositivo mobile al posto di quelle senza .mobile, in assenza delle .mobile vengono caricate quelle non mobile.

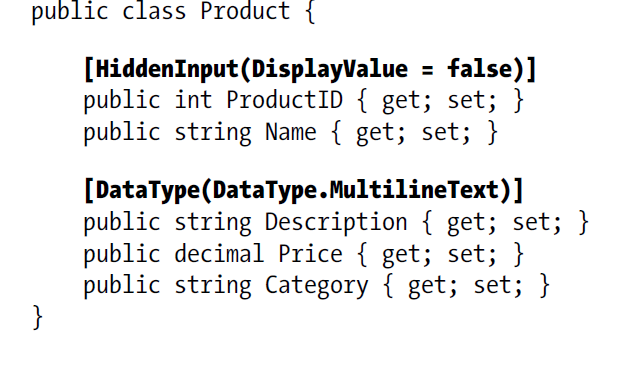
Arrivato a pag 275: qui viene creato il backoffice quindi la gestione dei prodotti: crud operation su prodotti.

Interessante come viene gestita la pagina di edit:



Dove html.EditorForModel è un modo velocissimo ma poco flessibile per rappresentare in automatico e gestire l’interfaccia di modifica.

Come nascondere un campo e generarne uno multyline con el datanotation:



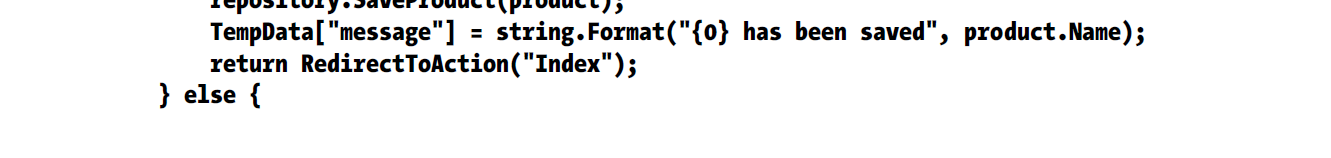
Ci sono tre modi per gestire la formattazione dei campi:

1 sfruttare le calssi applicate da bootstrap e ridefinirle.

Il secondo è creare dei templates per gli helper

Il terzo creare i singoli elementi

Nel caso una action redirezioni ad un’altra action non si può usare il viewback (che passa i dati tra controller e view) ma piuttosto:



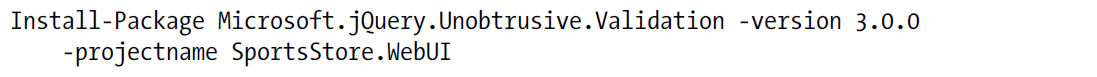
a pagina 296 si parla della validazione lato client con jquery:

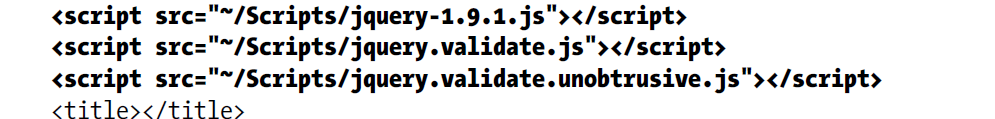
The MVC Framework can

perform client-side validation based on the data annotations I applied to the domain model class

This feature is enabled by default, but it has not been working because I have not added links to the required

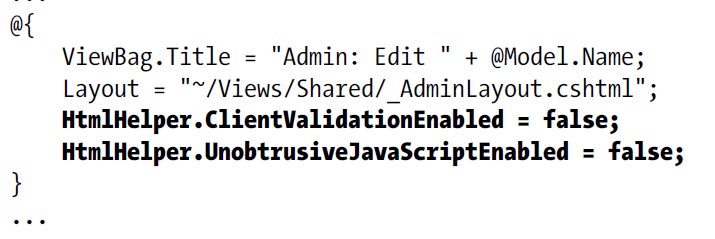
JavaScript libraries



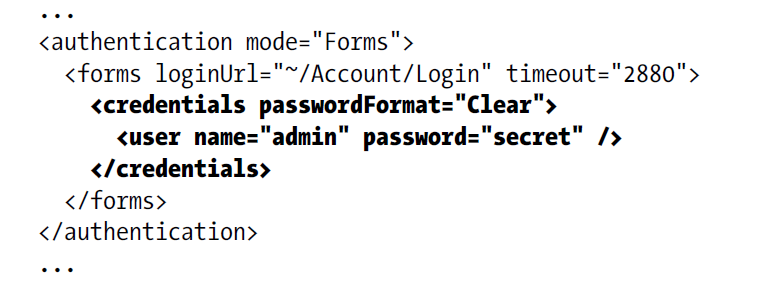


but if for some reason you do not want to validate at the client, you need to add the following

statements to the view



Pag 306 si parla di form autentication

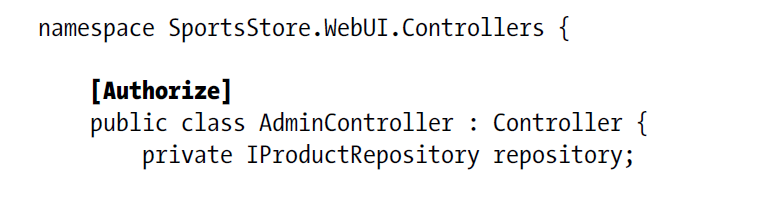


Si parla anche di filtri

The MVC Framework has a powerful feature called *filters*. These are .NET attributes that you can apply to an action

method or a controller class and they introduce additional logic when a request is processed to change the behavior of

the MVC Framework



When applied without parameters, the Authorize attribute grants access to the controller action methods

to all authenticated users

arrivato a pagina 321

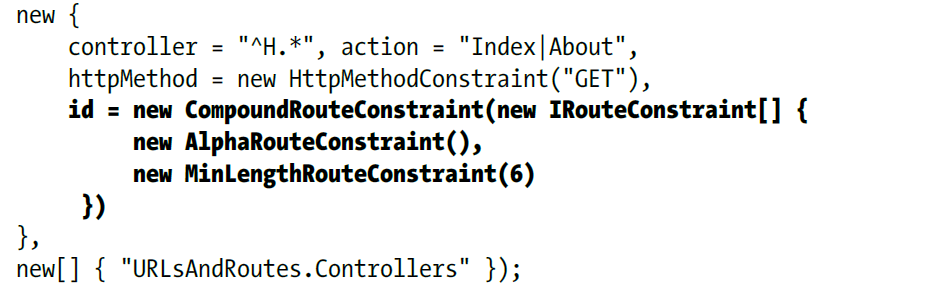
pag 345 - lavorare con MVC:

interessante qui fra gli strumenti di debug come abilitare il debug and continue

arrivato a pagina 398

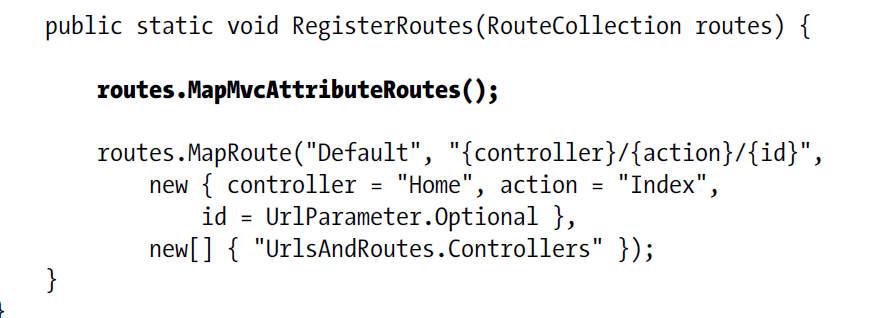
a pag 403 ci sono interessanti metodi di utilizzare i costraint per definire un range od un tipo valore, o val massimi o minimi ecc…

si possono anche esplicitare più costraint raggruppandoli:



A pag 405 si spiega che È possibile anche definire dei custom costraint facendo implementare l’interfaccia IrouteConstraint ad una classe.

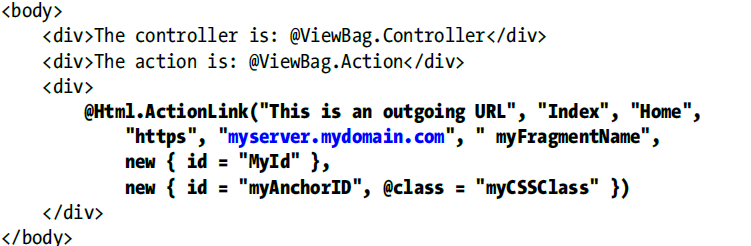
A pagina 407 viene spiegato l’attribute routing, la possibilità cioè di esplicitare il routing mediante degli attributi.

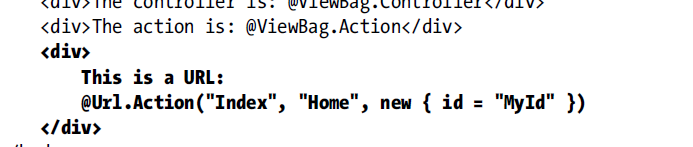


E una action ha una route attribute questa ha il sopravvento rispetto

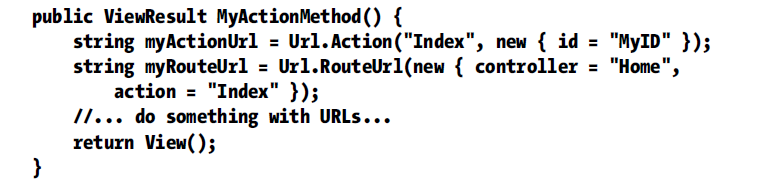
Arrivato a pagina 403

Si parla poi di html.actionlink, vengono esaminati i vari overload fino al più complesso per generare un fully qualified name e di url.action, quest’ultimo non genera un’ancora ma direttamente un link, esso ha una versione per view ed una per action (lato codice)

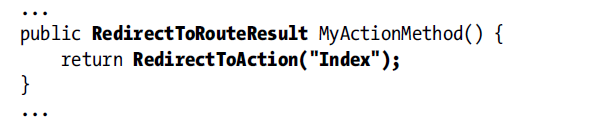




La differenza tra url.action e url.routeurl è che il secondo utilizza un oggetto per definire le sue caratteristiche

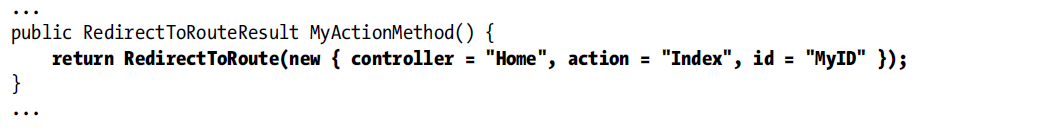


Qui invece per reindirizzare ad un altro indirizzo una view:

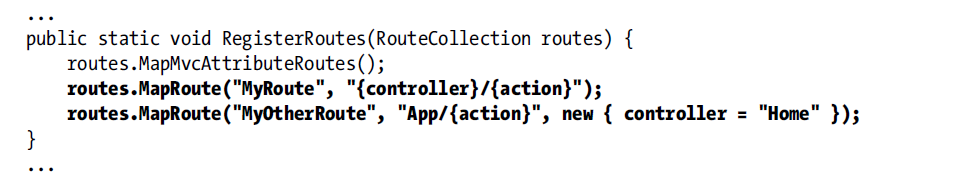


If you want to send a redirect using a URL generated from just object properties, you can use the

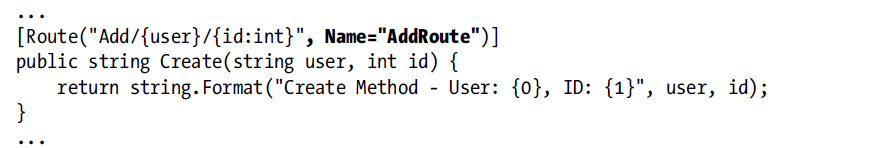
RedirectToRoute method

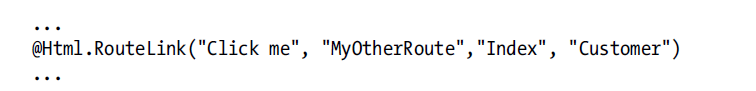


Dare un nome al mapping serve per potersi riferire esplicitamente alla regola di route:



Questo codice è equivalente a questo con attributi di route:





A pag 430 viene spiegato come getire una serie di link di richiesta e generare codice che in base a queste irichieste generi una rotta personalizzata mediante la classe RouteBase.

The purpose of the IRouteHandler interface is to provide a means to generate implementations of the

IHttpHandler interface, which is responsible for processing requests

IrouteHandler permette di definire una personalizzazione nelle regole di routing vioa codice

I register the custom route handler in the RouteConfig.cs

routes.Add(new Route("SayHello", new CustomRouteHandler()));

routes.RouteExistingFiles = true; forza a considerare prima una eventuale regola di route prima di fornire il file

I also have to tell IIS Express not to intercept

requests for disk files before they are passed to the MVC routing system.

a pagina 450 si parla del controller, interessante pagina 457 dove abbiamo tutto ciò che si può ottenere dall’oggetto richiesta (Request)

qui occorrerà approfondire cosa c’è all’interno di this nella action e l’oggetto request e response

a pag 462 viene mostrato come creare una action result customizzata.

Arrivato a pag 465

Quando viene chiamata una view da un controller il sistema prima cerca nelle cartelle areas guardando prima se trova aspx e ascx col nome passato al viewresult e poi cshtml e vbhtml e poi cerca nella cartella view con la stessa sequenza.

A pag 476 si parla di **redirect e redirecttoroute**, il secondo metodo redireziona creando il nuovo url indicando la action, il controller e quindi è meglio del primo che invece vuole un semplice link per il redirezionamento.

Esiste anche la possibilità di usare redirecttoaction

Uso di tempdata pagina 478

You can get a value from TempData without marking it for removal by using the Peek method, like this:

...

DateTime time = (DateTime)TempData.Peek("Date");

...

You can get a value from TempData without marking it for removal by using the Peek method, like this:

...

DateTime time = (DateTime)TempData.Peek("Date");

...

A pagina 480:

404 not found:

public HttpStatusCodeResult StatusCode() {

return new HttpStatusCodeResult(404, "URL cannot be serviced");

}

public HttpStatusCodeResult StatusCode() {

return HttpNotFound();

}

401 Unauthorized

public HttpStatusCodeResult StatusCode() {

return new HttpUnauthorizedResult();

}

A pagina 483 si parla dei filtri

Arrivato a pagina 493 (authorization e autentication filter)

Exception filter pag 499 (come creare filtra di ecceazione)

Pag 506 come usare builtin filters

A pag 509 abbiamo gli action filters

Salto avanti al capitolo 20 view

Prima viene fatto vedere dietro le quinte come viengono chiamate e renderizzate le view, poi si passa a considerare solo il razor engine

Le view vengono precompilate nei file temporanei (pag 568)

@Section e @rendersection (pag 575)

Child method:

The ChildActionOnly attribute ensures that an action method can be called *only* as a child method from

within a view.

Pag 587 :helper method

Inline helper pag 589

External helper method (pag 591)

Pag 606 es di overload più lugno per specificare il form.

Arrivato a pag 606

A pag 610 ci sono gli helper html

Poi seguono i controlli dinamici editor

Poi display e label

Quindi l’utilizzo dei metadata per condizionare la renderizzazione con i giusti controlli del form (pag 630)

Arrivato a pagina 636