**Jquery Pratico**

**Select 2**

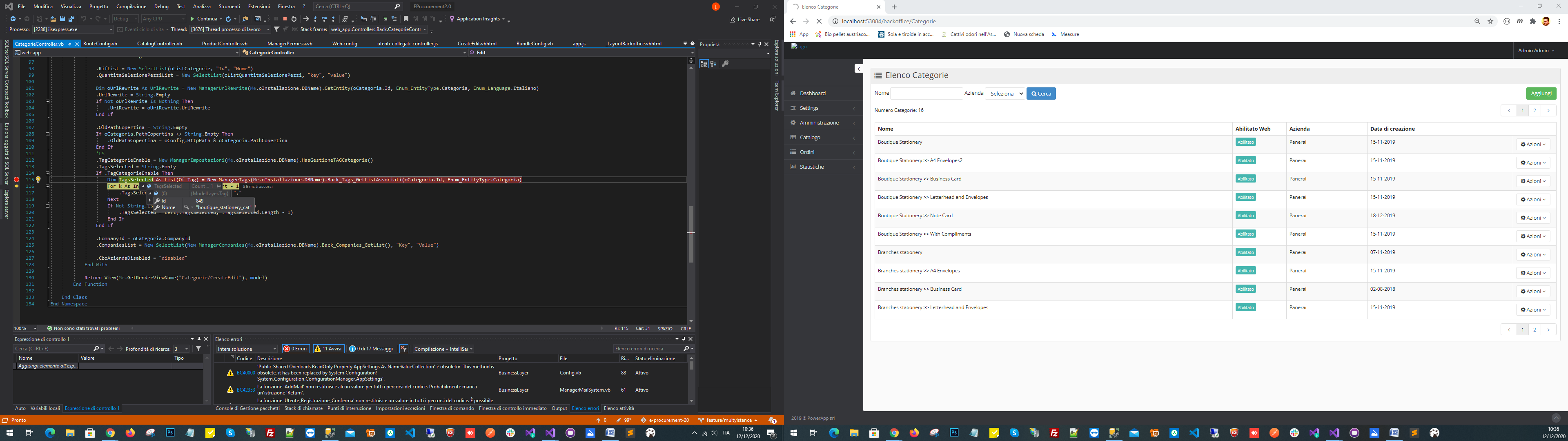
Lato server

Action:

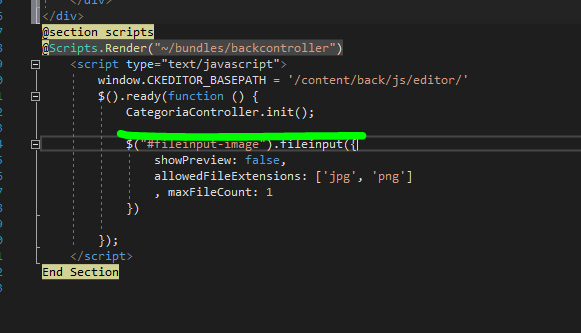
CREATE:

su create vuota 

EDIT: passiamo lista da caricare nella select2 al modelview della edit delle categorie per esempio:

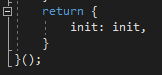


Lato view inizializziamo il controllo categorie

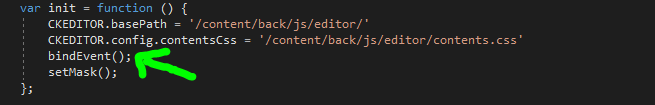


Il controller è un oggetto che viene eseguito quando viene letto e che espone un metodo init che inizializza il tutto.





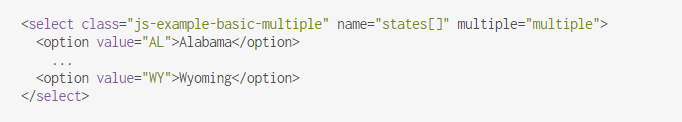
Il metodo init chiama bindevent il quale associa le callback da chiamare, ma quello che ci interessa è il metodo setmask il quale recupera i dati e li carica nella select2





Qui il metodo recupera il riferimento alla select2 chiama in ajax una api che fornisce in post una response e dentro data troviamo i dati (response.data) da passare all’enumerable tags della select 2, proprio qui viene associata la select2 all’elemento #TagsSelected

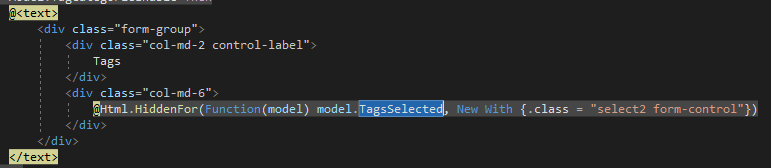
Come andrebbe creato l’html:



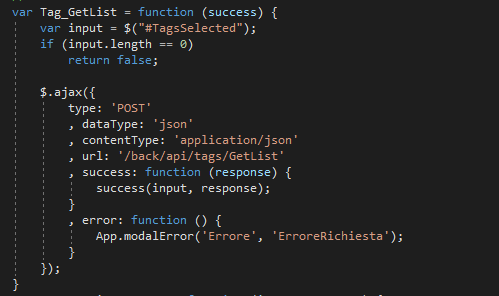
Come andrebbe semplicemente abilitata da javascript:

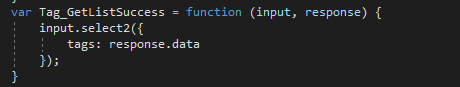


L’elemento sulla view:

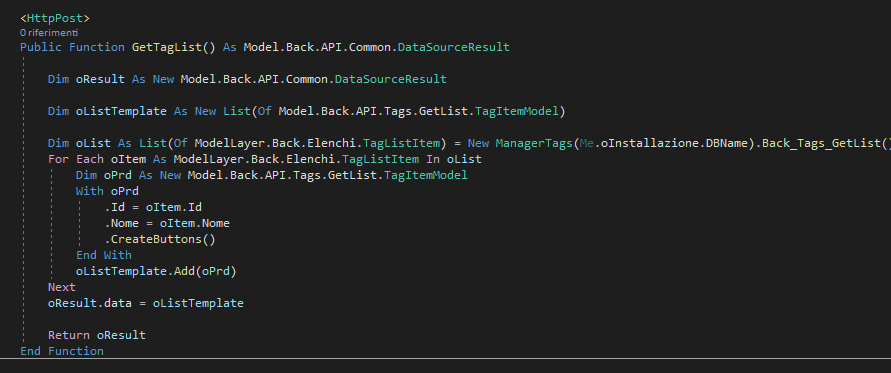


Il metodo…:

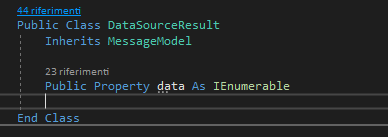


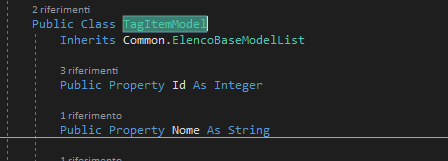


L’API lato server:

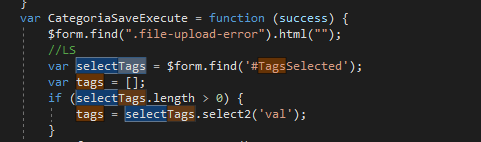


L’oggetto di risposta:





Per salvare dato attraverso js lo raccolgo dal form:



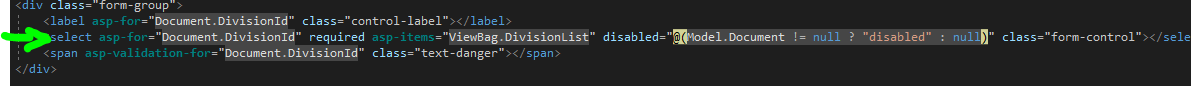
E lo appendo:



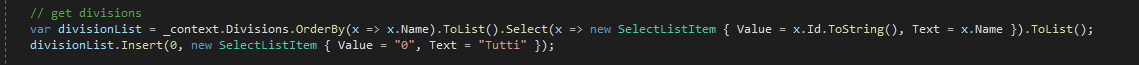
**Per la gestione lato server di select2:**

Esempio di una semplice combo a selezione singola select2 con possibilità di ricerca dei dati caricati

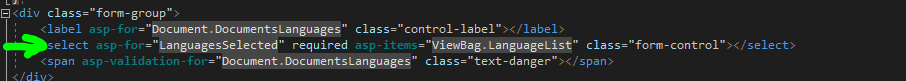
viene dichiarata una select, aspitems è la collection di tipo selectitem



Inserimento del default su create della combo:



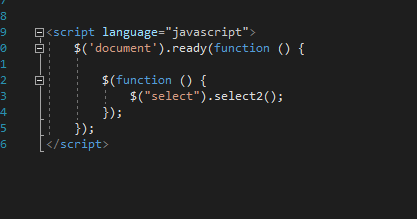
Questo è un esempio di select a selezione multipla (select2)



Qui come viene passata al viewmodel la collection:



La select viene inizializzata con select2:

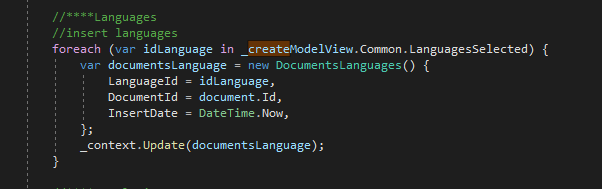


Qui come vengono salvati i dati:

nel caso di select singola recupereremo solo un valore

:

nel caso di select multipla:



**Dialog**

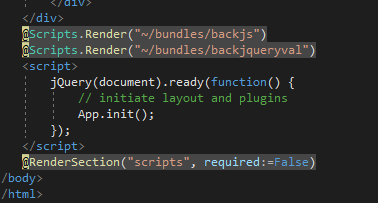
****

Bootstrap 3 Dialog

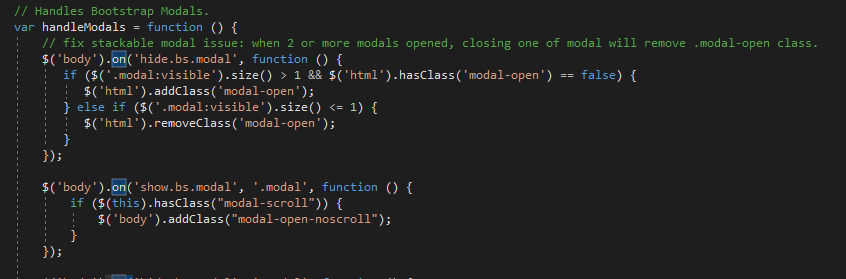
https://www.jqueryscript.net/lightbox/jQuery-Modal-Dialog-Plugin-For-Bootstrap-3-Bootstrap-3-Dialog.html

https://www.jqueryscript.net/demo/jQuery-Modal-Dialog-Plugin-For-Bootstrap-3-Bootstrap-3-Dialog/examples/

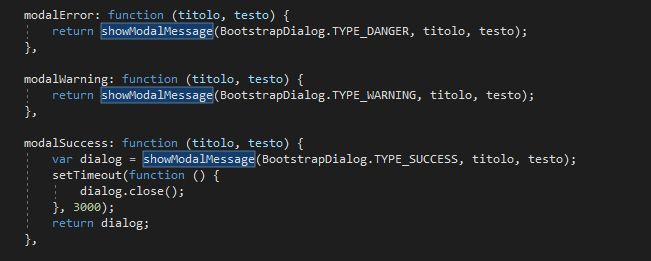
Definire un controller javascript globale (app.js) da inizializzare con init nel layout del progetto (\_layout.vbhtml):



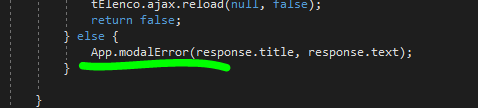
Viene gestito questo comportamento globale



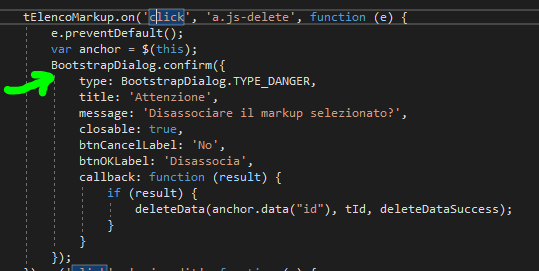
E poi vengono messi a disposizione questi metodi facendoli ritornare alla classe app nello stesso modo di init wrappando i comandi nativi:



…chiamata dei metodi pubblici di app in un altro controller:

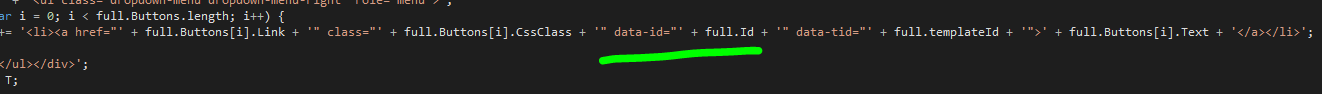


Per chiamare la dialog ad esempio nell’eliminazione della entità si chiede se si vuole procedere con una dialog ecco il metodo:

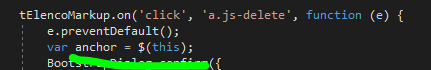


Interessante è la proprietà data (è un modo per recuperare dall’elemento corrente delle info passate:

passaggio dell’id allancora:



Recupero dell’ancora:



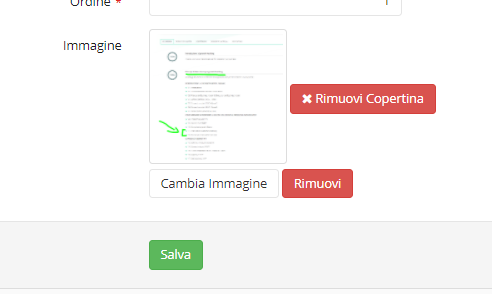
reupero dell’id con la proprietà data



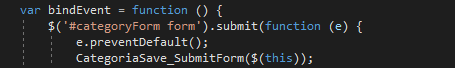
**File upload**

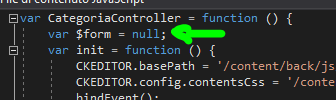
Jansy file upload ed upload del file con ajax

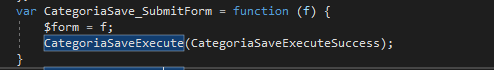
<https://www.jasny.net/bootstrap/2.3.1/javascript.html#fileupload>

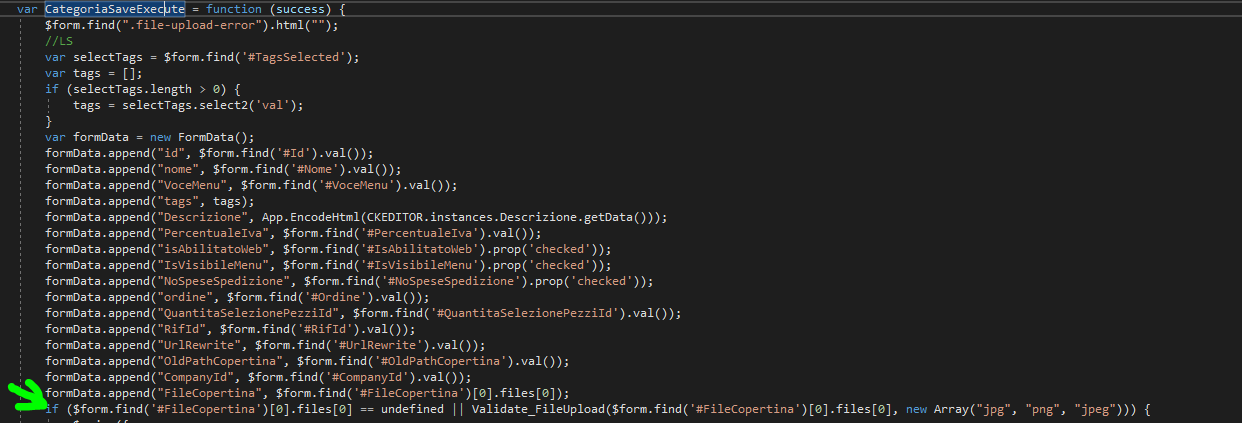


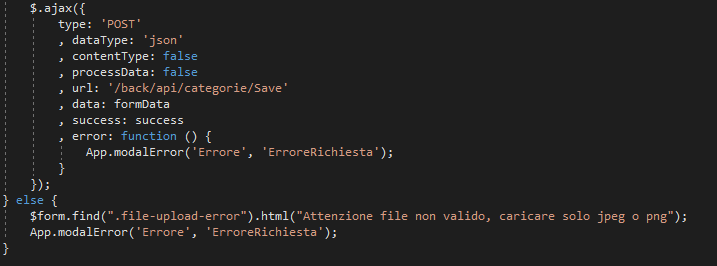
Viene fatto il binding a submit del form:

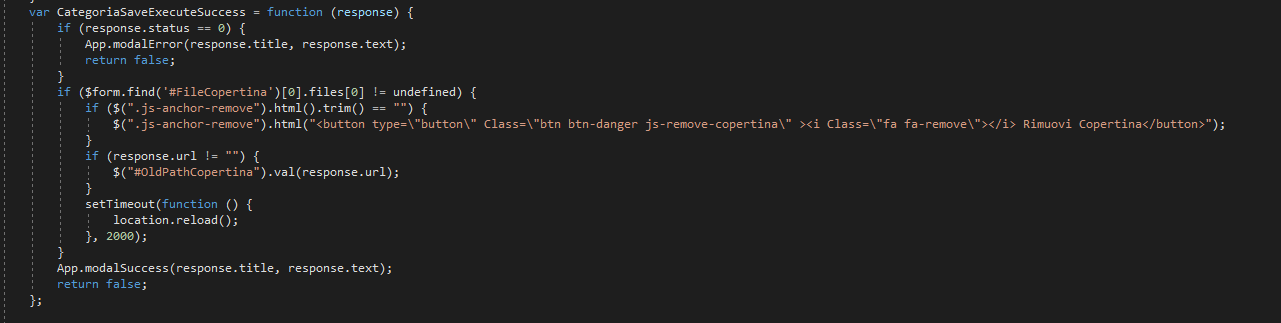




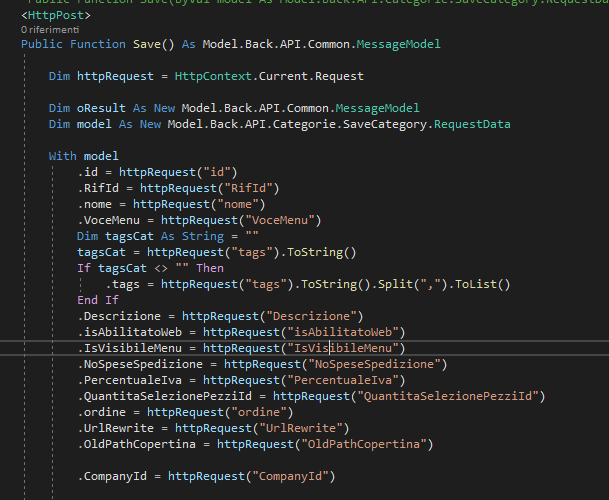




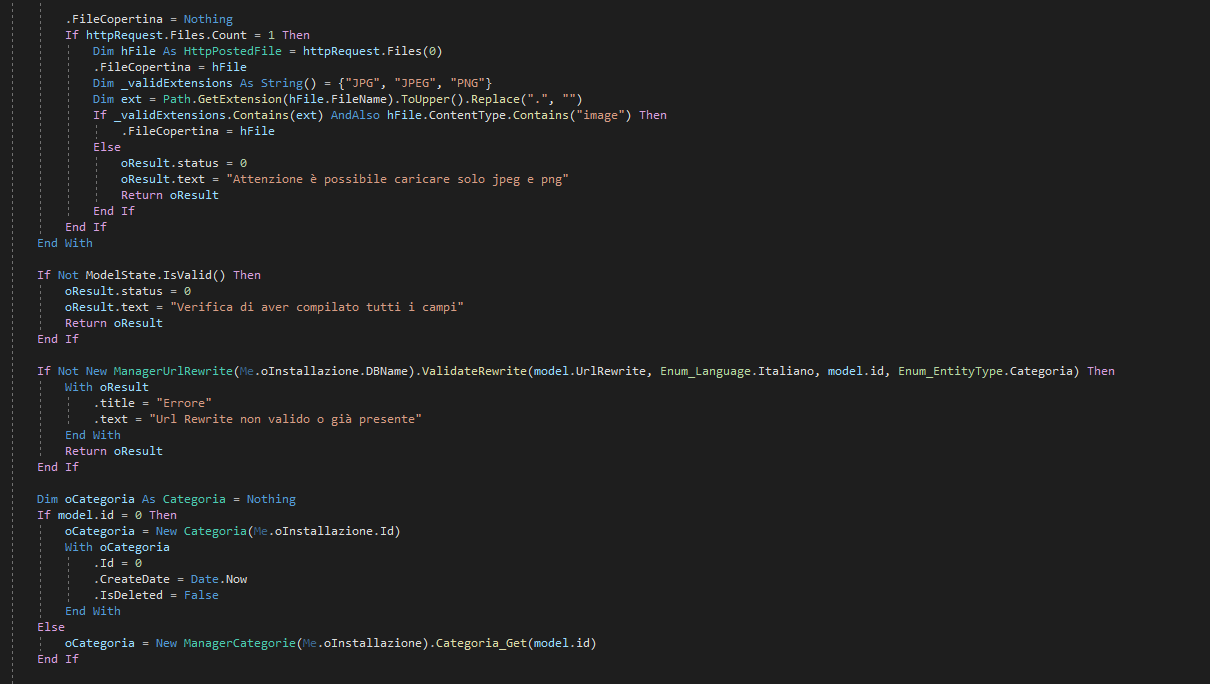


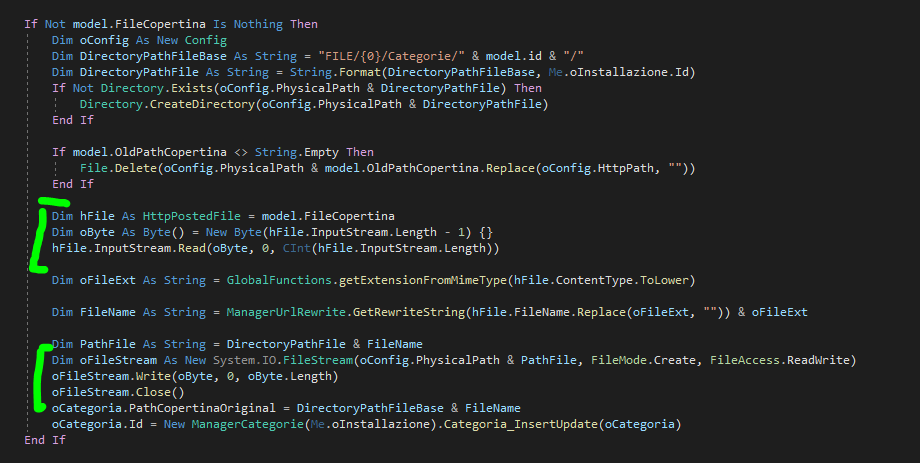


Api la to server:



Recupera il file dalla request e lo memorizza



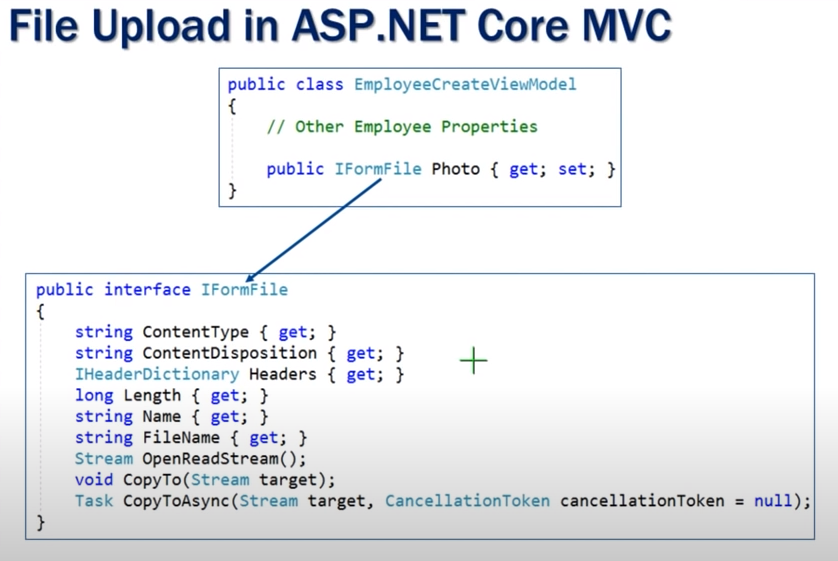


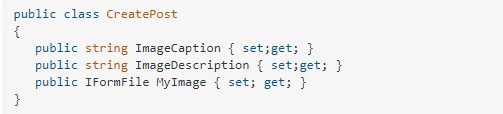
**Fare upload del file lato server:**

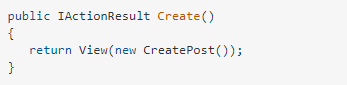
[**https://www.youtube.com/watch?v=aoxEJii70\_I**](https://www.youtube.com/watch?v=aoxEJii70_I)

**https://stackoverflow.com/questions/35379309/how-to-upload-files-in-asp-net-core**

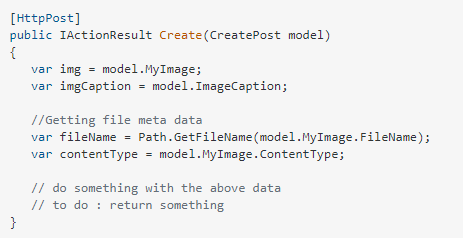
il modelview deve avere l’attributo del file di tipo IFormFile





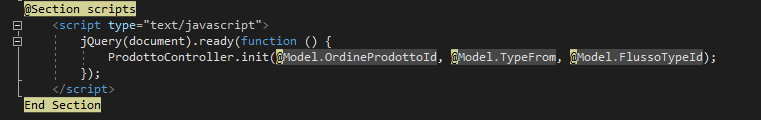


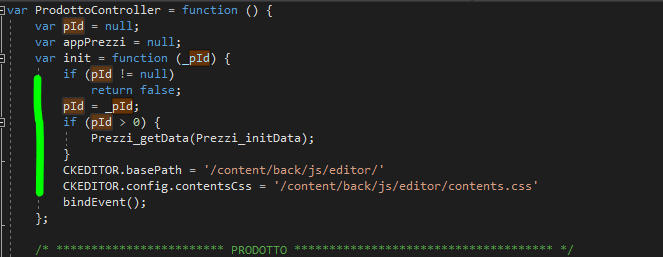




**Varie**

Ad un controllo è possibile passare dei dati al costruttore:





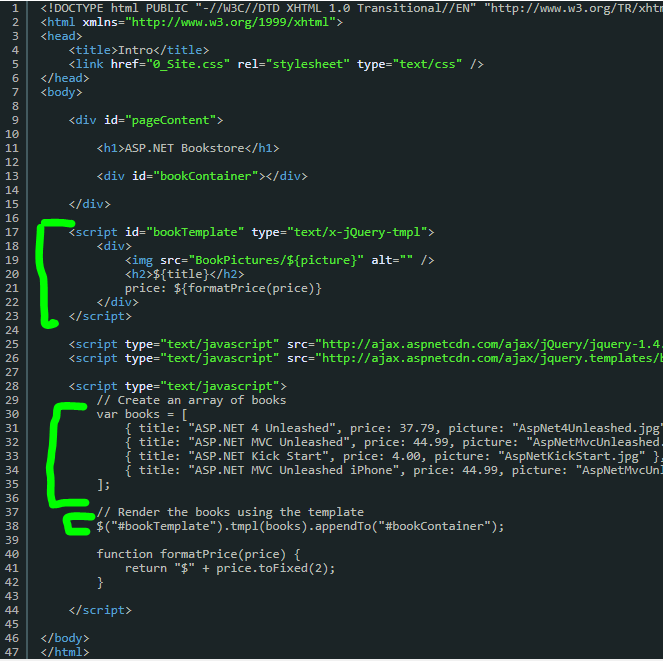
**Jquery templates.**

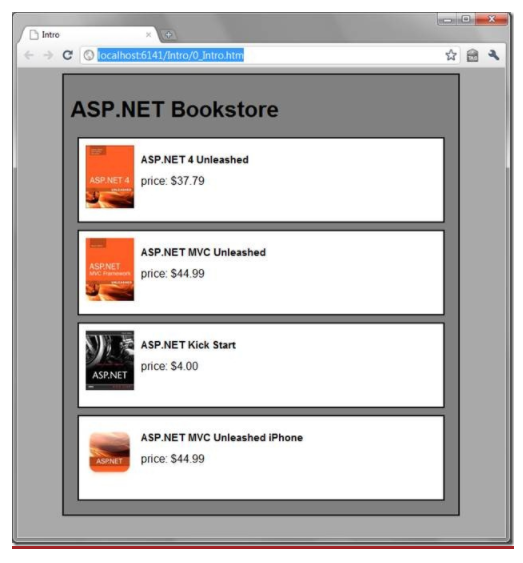
<http://stephenwalther.com/archive/2010/11/30/an-introduction-to-jquery-templates>

JQuery templating **is a** fast and efficient **technique to render client-side templates using a**[**JSON**](https://www.w3schools.com/js/js_json_intro.asp)**data source**. It refers to the client-side **data binding method with the JQuery**

Templating is essentially a **combination of data and layout**. The advantage of doing this is that **your data and how the data are kept are separate from each other**. Now, templating is usually used to format frequently used contents.

Esempio base:





**1)First, notice that the page uses both the jQuery 1.4.4 and jQuery Templates libraries.** Both libraries are retrieved from the ASP.NET CDN:

<script type=”text/javascript” src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.4.4.js”></script>

<script type=”text/javascript” src=”http://ajax.aspnetcdn.com/ajax/jquery.templates/beta1/jquery.tmpl.js”></script>

2)Second, you should notice that the actual template is included in a script tag with a special MIME type

When a browser encounters a SCRIPT tag with an unknown MIME type, it ignores the content of the tag. This is the behavior that you want with a template. You don’t want a browser to attempt to parse the contents of a template because this might cause side effects.

3)Third, notice that the expression ${…} is used to display the value of a JavaScript expression within a template.

4) Fourth, and finally, the template is rendered with the help of the tmpl() method.

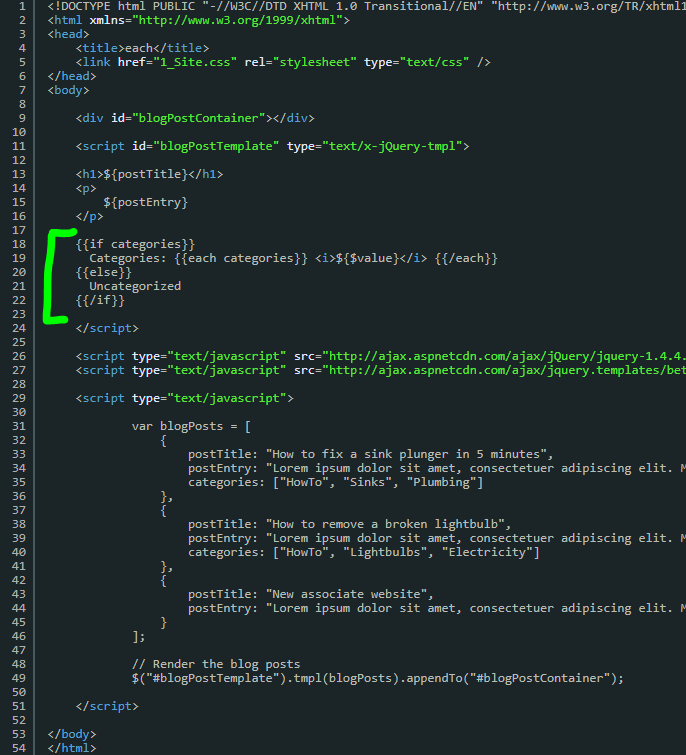
$(“#bookTemplate”).tmpl(books).appendTo(“#bookContainer”);

Using Template Tags

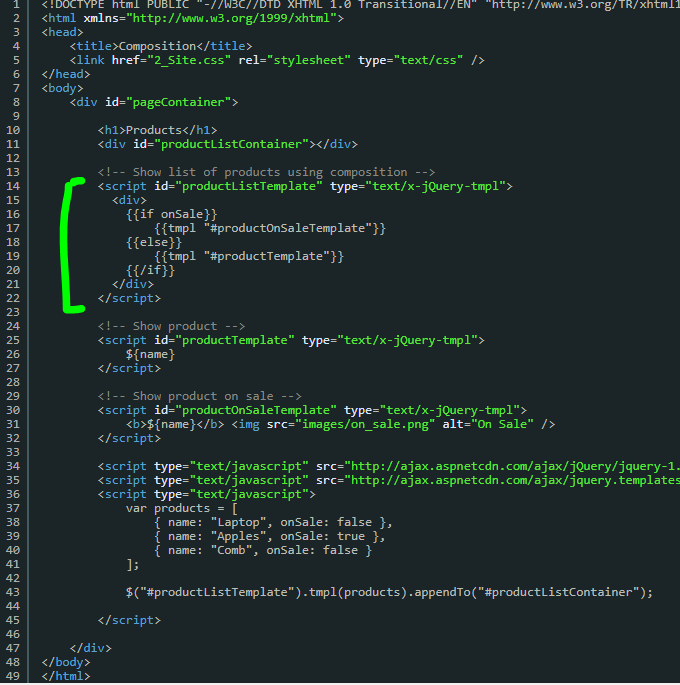
Within a template, you can use any of the following *template tags*.

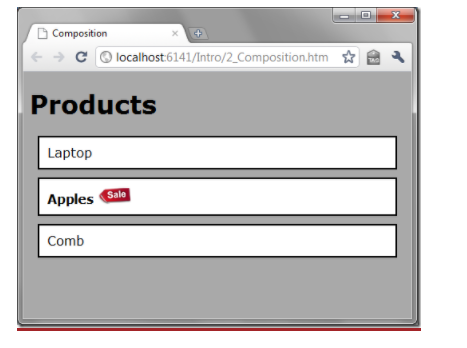
* *{{tmpl}}* – Used for template composition. See the section below.
* *{{wrap}}* – Used for wrapped templates. See the section below.
* *{{each}}* – Used to iterate through a collection.
* *{{if}}* – Used to conditionally display template content.
* *{{else}}* – Used with *{{if}}* to conditionally display template content.
* *{{html}}* – Used to display the value of an HTML expression without encoding the value. Using ${…} or {{= }} performs HTML encoding automatically.
* *{{= }}*— Used in exactly the same way as ${…}.
* *{{! }}* – Used for displaying comments. The contents of a *{{!…}}* tag are ignored.

Example of use of each and if:

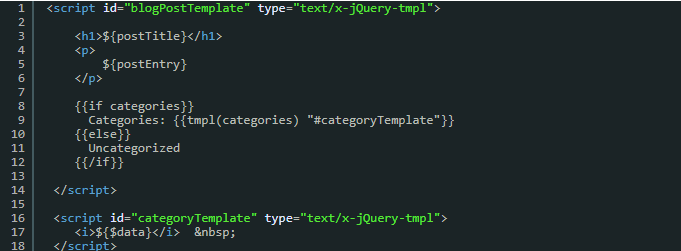


The following page illustrates how you can use the *{{tmpl}}* tag to build a template from multiple templates:





You can pass data to innher template:



A common feature request for jQuery templates is support for remote templates. Developers want to be able to separate templates into different files.

Adding support for remote templates requires only a few lines of extra code ([Dave Ward has a nice blog entry on this](http://encosia.com/2010/10/05/using-external-templates-with-jquery-templates/)). For example, the following page uses a remote template from a file named BookTemplate.htm: