

jQuery Templates vs Partial Views in ASP.NET MVC

Asked 9 years, 4 months ago Active 9 years, 4 months ago Viewed 4k times



8



5



I'm taking a look at jQuery templates. It looks really interesting - easy syntax, easy to use, very clean.

However, I can't really see why it's better to use jQuery templates instead of simply fetching partial views via AJAX. It simply seems like the partial views would be much easier to maintain and helps to avoid duplication of code.

I want to use jQuery templates. But when would it be better than partial views?

[jquery](#)[asp.net-mvc](#)[jquery-templates](#)

asked Feb 13 '11 at 7:16



[Jaco Pretorius](#)

23.1k ● 9 ● 53 ● 90

1 "I want to use jQuery templates" - why? – [bzlm](#) Feb 13 '11 at 10:12

1 @bzlm: It's all about the hype ;-) – [Adrian Grigore](#) Feb 13 '11 at 10:21

1 @Adrian Yeah, maybe. :) @Jaco With regard to "easy syntax" in your question, I beg to differ. All the `{{}}` stuff is just one more "view syntax" to learn. If you're already on ASP.NET MVC, jQuery template will just slow you down. – [bzlm](#) Feb 13 '11 at 11:55

3 Answers

[Active](#)[Oldest](#)[Votes](#)

11



I agree that these do overlap. There are several different ways to implement the same piece of software, and much of your decision on what to use depends on your personal preference and the context of your software.

Partial view advantages:

- Type-safe (if using strongly-typed viewmodels)
- Allows static syntax and type checking.
- Full code completion / syntax highlighting support in Visual Studio



jQuery Templates advantages:

- Allows the page to be updated before performing postbacks or without any postbacks to the server side at all. This can be handy when creating heavily ajaxed interfaces, possibly also with html5-driven offline capabilities.
- You can retrieve data from the server in JSON format and render that to HTML. JSON is much shorter than HTML formatting, so this can make a difference in page loading times for long lists of data entries when using a slow internet connection.

So in essence partial views are the more stable and jquery templates are (for ajax websites) the more performant choice. I therefore use partial views for less frequented web pages which need to be developed quickly and jQuery templates for heavily ajaxed web pages where performance really matters.

edited Feb 13 '11 at 10:09

answered Feb 13 '11 at 10:02



Adrian Grigore

31k ● 30 ● 125 ● 204

I was hoping I could make a more compelling argument for jQuery templates but I guess this is the best we can do. Thanks Adrian – **Jaco Pretorius**
Feb 14 '11 at 17:55

-
- 1 @Jaco Pretorius: Don't get me wrong - jQuery templates is indeed a great plugin and I am using it regularly. It's just that I prefer server-side technology for it's ease of use and stability unless I have a good reason to go with a more client-side heavy solution. I imagine there are people who actually feel more comfortable using Javascript than Razor / C# and for them jQuery Templates is the way to go. Neither is a poor choice IMO. – **Adrian Grigore** Feb 14 '11 at 18:53

Also: jQuery templates work great with [KnockoutJS](#). Try it if you haven't already, its really neat. – **Anders Fjeldstad** Jun 30 '11 at 13:08

@Anders: It is indeed neat, but as I mentioned I really prefer a more server-centric approach. It's not as slick, but more robust in my experience. – **Adrian Grigore** Jun 30 '11 at 21:09

@Adrian Have you had negative experiences of jQuery templates with respect to robustness? Would be interesting to hear about. – **Anders Fjeldstad**
Jul 1 '11 at 6:50



3




One example is when you want to display a number of rows initially, but then also add/remove single rows over the course of the page view (think displaying and then editing an invoice document). Having the granular control to be able to render a single row at a time gives you the ability to significantly improve the responsiveness of your app.

answered Feb 13 '11 at 7:18



Dave Ward

55.3k ● 10 ● 113 ● 134

- 1 But you could also do the same by loading a partial view that only shows a single row and inserting that into the DOM. `$.load(...)` does exactly that. – [Adrian Grigore](#) Feb 13 '11 at 10:04
- 2 Not exactly. In many applications, you may want to give the user a real-time editing experience by instantly adding templated rows on the client-side. There wouldn't be a good reason to block the user's workflow with client-server roundtrip delays in that case, when the additions could be saved in the background or as a group later. When you depend entirely on the server for markup generation, you put an unnecessary limitation on the level of interactivity your application can achieve. – [Dave Ward](#) Feb 13 '11 at 14:46
- 1 I understand what you mean, and it is indeed a on optimal implementation performance-wise, but the price for this is that you have to deal with and synchronize a server-side and client-side data store. It means a more complex implementation and ultimately more javascript that can easily break and is therefore harder to refactor than server-side code. It's the only way to go if you are coding a web app with offline capabilities, but for a regular website I'd rather go with a simpler solution. – [Adrian Grigore](#) Feb 14 '11 at 19:00 
- 2 I've been using client-side templates for years and never encountered any of that. Quite the opposite, using JavaScript for UI code has dramatically simplified most internal applications I've worked on, where JavaScript support can be mandated. Not only do you offer users better performance and a higher level of interactivity, but you achieve higher developer productivity too. In my experience, those gains greatly outweigh any losses in refactoring productivity. – [Dave Ward](#) Feb 14 '11 at 19:27



2

i would add that one of the benefits of the jqt is that you can very easily pass the front end development off to a 3rd party (who specialise in this type of work) and they only need to know the object model of the json. (plus maybe a few 'interfaces' back to the server)



this way, your 'backend' developers can focus on the task at hand and the 'front' end team can develop the jqt. the other advantage of jqt being that they are totally isolated from the .net environment, therefore it's feasible (if unlikely) that the backend technology could be changed (j2ee, php etc) for a subset of actions without the front-end caring beyond the rest url.



it's an interesting 'space' to observe and one that i'm only just beginning to dabble with. however, on both sides of the fence, things will only get better and as i mentioned to dave (ward) a few weeks back, perhaps some of the razor engine endpoints will offer a way to integrate directly with jqt in future versions - seems kinda inevitable.

answered Feb 13 '11 at 11:58

[jim tollan](#)

21.3k ● 4 ● 42 ● 62

If such a fence exists, I propose tearing it down. :) – [bzlm](#) Feb 13 '11 at 13:50

Have a look at blazor. you use razor and C# to make single page web apps on the client. blogs.msdn.microsoft.com/webdev/2018/02/06/... – [Harry](#) Mar 13 '18 at 18:33 