

Re: Code Behind vs not

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.aspnet/2005-02/6626.html>

From: tshad (*tscheiderich_at_ftsolutions.com*)

Date: 02/22/05

Date: Tue, 22 Feb 2005 08:08:15 -0800

"Kevin Spencer" <kevin@DIESPAMMERSDIEtakempis.com> wrote in message news:eTYXPIOGFHA.1740@TK2MSFTNGP09.phx.gbl...

> *Very good points all, Xepol. I was particularly impressed with the
> discussion of compromise in programming, which is a rather advanced but
> important idea to embrace. Once you have a good grasp of the tools, it is
> important to remember that code efficiency is not the only goal to shoot
> for. A few cycles here and there don't make a lot of difference with
> computing power at its present level, but code readability and
> maintainability are certainly high priority, as programmer man/hours are
> expensive.*

I agree.

Tom

>

> --

> *Kudos,*

>

> *Kevin Spencer*

> *Microsoft MVP*

> *.Net Developer*

> *Neither a follower nor a lender be.*

>

> *"Xepol" <Xepol@discussions.microsoft.com> wrote in message*

> *news:679A4772-0253-48C1-9188-5697508C1E2A@microsoft.com...*

>>

>>

>> *"tshad" wrote:*

>>> *If I have a choice between the most efficient and the most readable and*

>>> *maintainable – the best way is not necessarily going to be the same.*

>>

>> *The best code is both readable and maintainable and balances efficiency.*

>> *Why? Because truly efficient code is NEVER readable, and NEVER*

>> *maintainable.*

>>

>> *True efficiency comes from breaking rules, and the code is ALWAYS ugly*

>> *and*
>> *you can only maintain the code only if you truly understand it. True*
>> *understanding is a result of readability. If you had to read the code to*
>> *understand it, and understand the code to read it, you've got a catch22*
>> *that*
>> *most people will never get past.*
>>
>> *Besides which, you are writing code for a stack based machine without*
>> *registers which will ultimately be compiled to SOME hardware SOMEWHERE*
>> *with*
>> *or without registers, using SOME sort of optimization. It is impossible*
>> *to*
>> *reach the true ideal of "efficient" code with that many levels of*
>> *abstraction*
>> *in the way.*
>>
>> *Shoot for reasonable efficiency and make your code both readable and*
>> *maintainable, because you ALWAYS end up having to work on code again 3*
>> *years*
>> *after you though you'd never see it again (or worse, someone else). And*
>> *frankly, spending copious amounts of time trying to read code to figure*
>> *out*
>> *what it does and how to modify it without breaking everything is*
>> *decidedly*
>> *inefficient.*
>>
>> *That said, inline code is good for single page projects where very simple*
>> *operations are being performed (I even have 1 page designed like that, it*
>> *contains about 3 lines of code, hardly worth code behind!). However,*
>> *inline*
>> *code is horribly inefficient when dealing with a project that contains*
>> *more*
>> *than 3 pages which interact (and even if you don't think you are making*
>> *objects, think again, each page is an object).*
>>
>> *I think, however, we need to discuss your use of tools. I believe you*
>> *mentioned you were using Dreamweaver. I suspect that you will find that*
>> *OTHER tools, such as Visual Studio are significantly more efficient when*
>> *it*
>> *comes to working with ASP.NET pages (you will also find tools from other*
>> *vendors such as borland, so you are not restricted just to MS, but even*
>> *as a*
>> *long time delphi programmer, I prefer VS to borland in this arena).*
>>
>> *If you would like to see how a larger scale project works with code*
>> *behind,*
>> *check out dotnetnuke (www.dotnetnuke.com), and just imagine how*
>> *unbelievably*
>> *hard it would be to manage the project with inline code (esp. since*
>> *they've*
>> *abstracted commonly used code into objects with no associated pages).*

microsoft.public.dotnet.framework.aspnet: Re: Code Behind vs not

>>

>> *Hope that helps.*

>

>