ALLAN THRAEN | 1 years ago | PDF |

Tech Talk C# Tips and Tricks Optimizely (Episerver) CMS .NET Development

# CHRISTMAS COUNTDOWN: #9 WHAT? VIEWMODELS? NAH, WE DON'T NEED THOSE


**This is another classic - with a big impact! Since recycling is great, why don't we just reuse the content model as a view model? We can just enrich it in the controller, right?**

## The 2023 Christmas Countdown: 12 Common Pitfalls in Optimizely CMS - and how to avoid them

*This blogpost is part of the 2023 Christmas Countdown series where I each day for the last 12 days before Christmas go through my Top 12 list of the most common and dangerous pitfalls I typically see in Optimizely (EPiServer) CMS 12 Implementations. If you want to learn more and perhaps have your own site evaluated feel free to reach out to us here at CodeArt.*
Here are links to all the posts in this series as they are published:
12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Today's pitfall, number #9 on my Top 12, is something so scary that I strongly advice not reading this blogpost before just before bedtime as I won't be liable if I spoil your sleep with nightmares.
In fact, it's so terrifying that I've previously dedicated an entire blogpost ( https://www.codeart.dk/blog/2021/10/anti-pattern-dont-modify--optimizely-cms-episerver--content-objects-in-the-controller/ ) to warn about it. In fact, the only reason this isn't all the way on the top of the list is that it after all isn't as common as some of the others - yet, I've seen it in quite a few codebases in different variations.

"Could you remind me again what it is and why it's so scary?" you might ask? Sure - in short - it's when a developer uses the content model object provided by the CMS to also contain data specific to the current user, as both needs to be shown in the view. This is typically done in the controller or in inline code in the view - but it could also happen in a factory or a service they have built with the purpose of enriching the content with user specific data.

Consider the code block below:

```
public class MyAwfulBlockController : BlockController<MyAwfulBlock>
{
    public override ActionResult Index(MyAwfulBlock currentBlock)
                                                    {
        var userDetails = FetchCurrentUserDetails(); //Imaginary helper method that fetches information specific to this request/user
        currentBlock.UserDetails = userDetails;
        return PartialView(currentBlock);
    }
}
```

To understand why this is so bad, you have to keep in mind that the CMS caches all content objects in memory to improve performance. So, when you are handed a content object in a controller, it's not just an object just created for you right there and then, but an object that's shared for all users and stored in memory.

This means, **if you modify that object - even without saving it - you are modifying it for all users!**

And this is where we are approaching Freddy Krueger scary: Most of the time it will work as intended. Because any request will typically first hit the controller, have the right values set and then the view a few milliseconds later, where they are rendered and all is well. But the moment another request for the same ressource hits the logic that sets data before the first has finished rendering you have a classic race condition with a potentially serious data leak as the result.

So, for todays scary and fun experiment search your codebase for something like "*[Ignore]*" and see if any content models have an ignored property that is set in their controllers (in most cases the property has to be ignored, otherwise the CMS will typically throw an exception if you try to set it).
If there are ignored properties - there can be perfectly legit reasons for it - so next step is to see where the property is set - and you can pray it's not in your controllers. Otherwise you probably won't sleep tonight, but instead filling out the required legal paperwork explaining why you have had a potential data leak.
And - if you're the developer that was too lazy to add a viewmodel in the first place? Well - all I can say is that Santa supposedly knows if you have been naughty or nice, so I wouldn't hold my breath for any cool gadgets under the christmas tree this year.

Tech Talk C# Tips and Tricks Optimizely (Episerver) CMS .NET Development