2

Model

A model is a representation of something meaningful. Not necessarily something physical, but something real: a concept or a business or an API that's difficult to work with.

When we write object oriented software, we create classes that make up this representation. We can create our representation so that when we use it we are working in our natural human language, like English or Spanish or business jargon, instead of in mere programming language constructs like booleans, meaningless strings and integers.

When working with a UI framework like ASP.NET MVC the meaningful thing we have, the complex problem we manage, is the UI. It's the data in a window, a form submission from a user, the options in a select list. The model represents the screen.

There are other types of models. Domain models typically represent a part of a business, and we cover the domain model in chapter 8. A model that represents the user interface is sometimes called a presentation model or a view model. Many simple applications will share a domain model and a presentation model - they'll effectively be the same. But these are only the most trivial of applications, and even then it's advisable to segregate these duties.

Somewhere in our application we'll build the presentation model. It may come directly from a database, like a plain, tabular report. It may be projected from a domain model. It's common to have a class whose sole responsibility is to translate from a domain model to a presentation model. We'll discuss later how a controller takes this presentation model and sends it to a view. And we'll see how a view is left with the relatively easy task of painting the model on the user's monitor.

One key tip that we learned the hard way: don't share models between views. A presentation model exists for one screen. Sharing them makes changing one screen or the other difficult, because they both depend on the same thing. We want to allow our models the flexibility to fit each screen and not be tied to another, common structure.

ViewModel

In ASP.NET MVC the model goes in an object called ViewData. It's of type ViewDataDictionary, which is a regular dictionary but it also has a special property called Model.

- ViewDataDictionary

- code snippet of magic string stuff from default template

- show controller

- why this is bad and error prone

- the Model property

- why this is better

- show page directive

- show <%= summary.FirstName %>

- show controller

Display model example

Input model example

- the input model represents user input

- Helpers (quick)

- Modelbinding (quick)

Complex scenarios