Knockout.js
Tech Demo

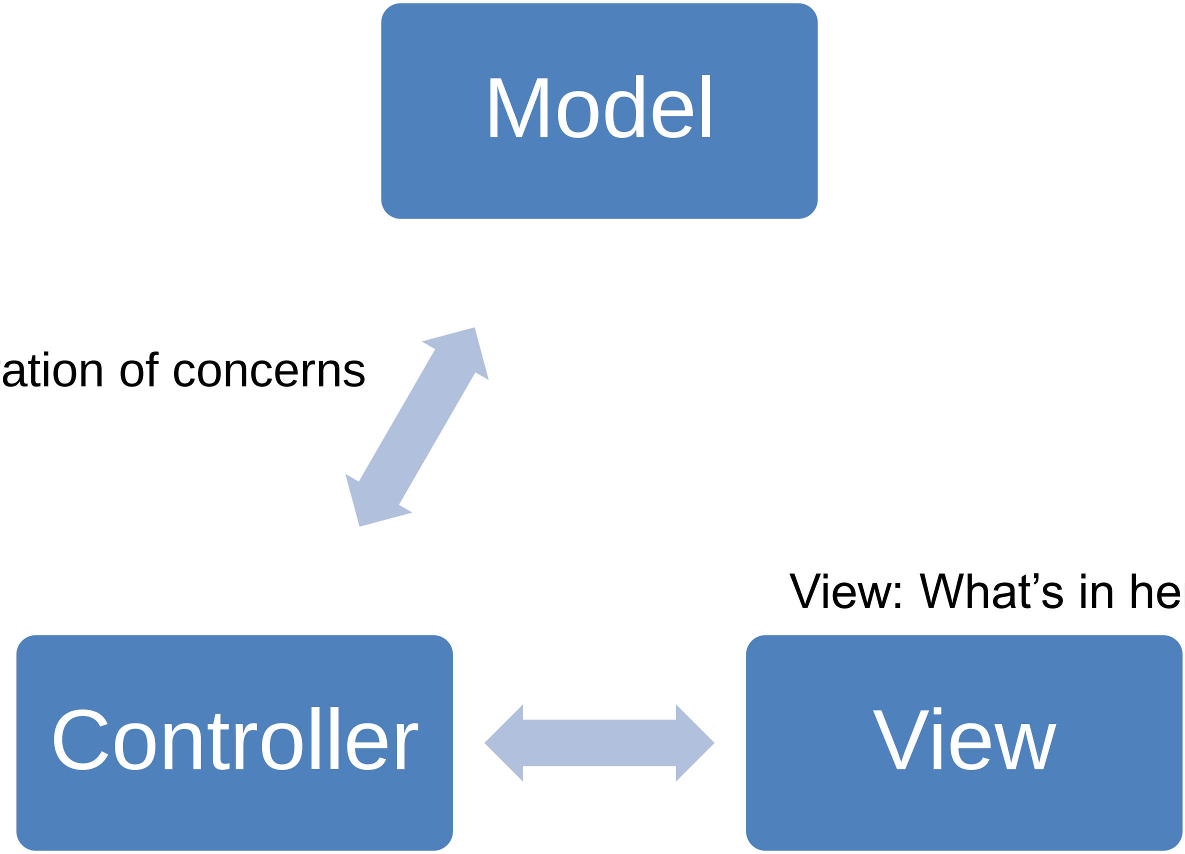
Backed by ASP.NET Web API 2 in Visual Studio 2013

Steve Kollmansberger
Lead Developer, Bridge Preservation Applications

WSDOT Developers Conference
January 14, 2014

Follow along with the code: http://www.kolls.net/wsdot/ko2014

Traditional MVC

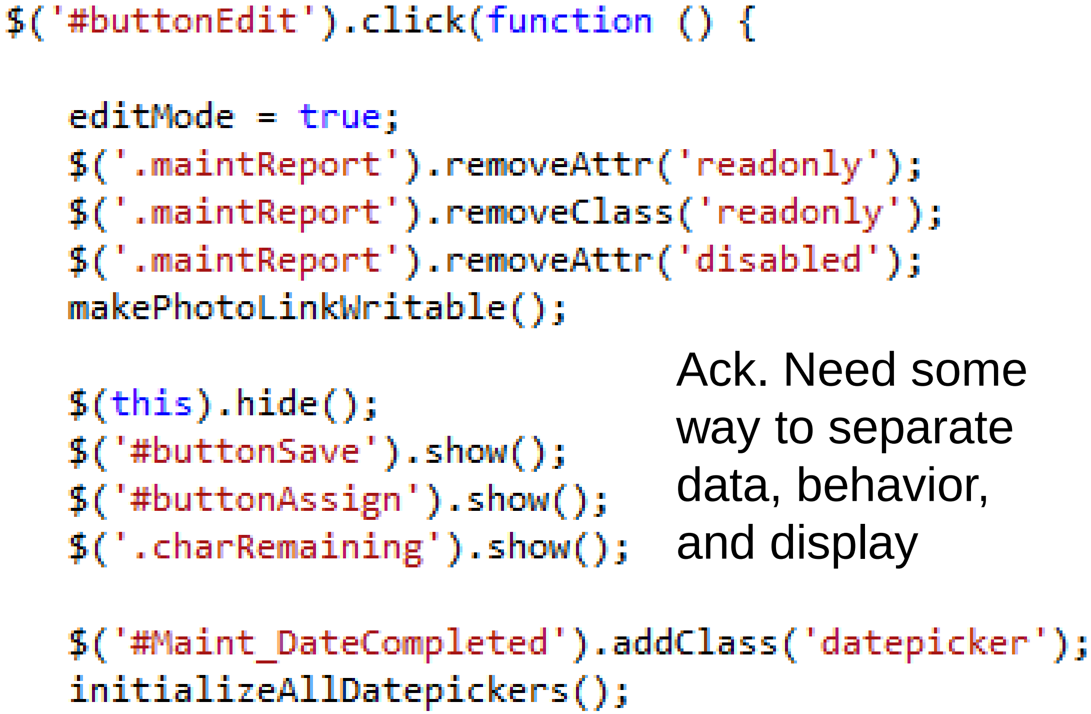


Ideal:
Clean separation of concerns

View: What’s in here?

The “View”

- Giant blended mess of HTML, C# (via Razor), and JS
- Mixing layout and behavior; no clean separation between data and the “controls” that hold them, all in one file
- Looks like (bad) PHP; Tastes like WebForms



```
$('#buttonEdit').click(function () {

    editMode = true;
    $('.maintReport').removeAttr('readonly');
    $('.maintReport').removeClass('readonly');
    $('.maintReport').removeAttr('disabled');
    makePhotoLinkWritable();

    $(this).hide();
    $('#buttonSave').show();
    $('#buttonAssign').show();
    $('.charRemaining').show();

    $('#Maint_DateCompleted').addClass('datepicker');
    initializeAllDatepickers();
```

Ack. Need some way to separate data, behavior, and display

What Do We Want?

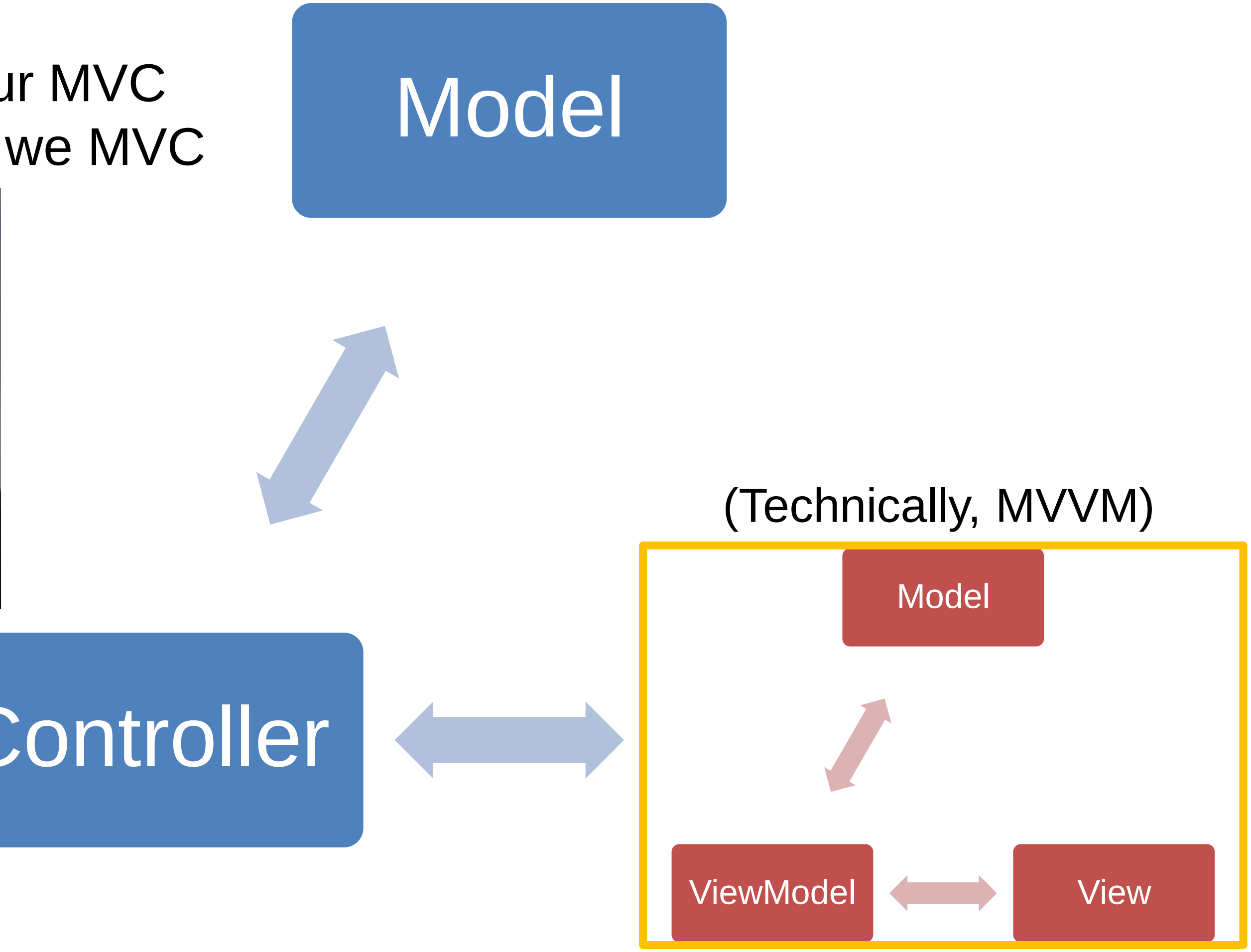
Aggressive separation of concerns
Testability

When Do We Want It?

NOW

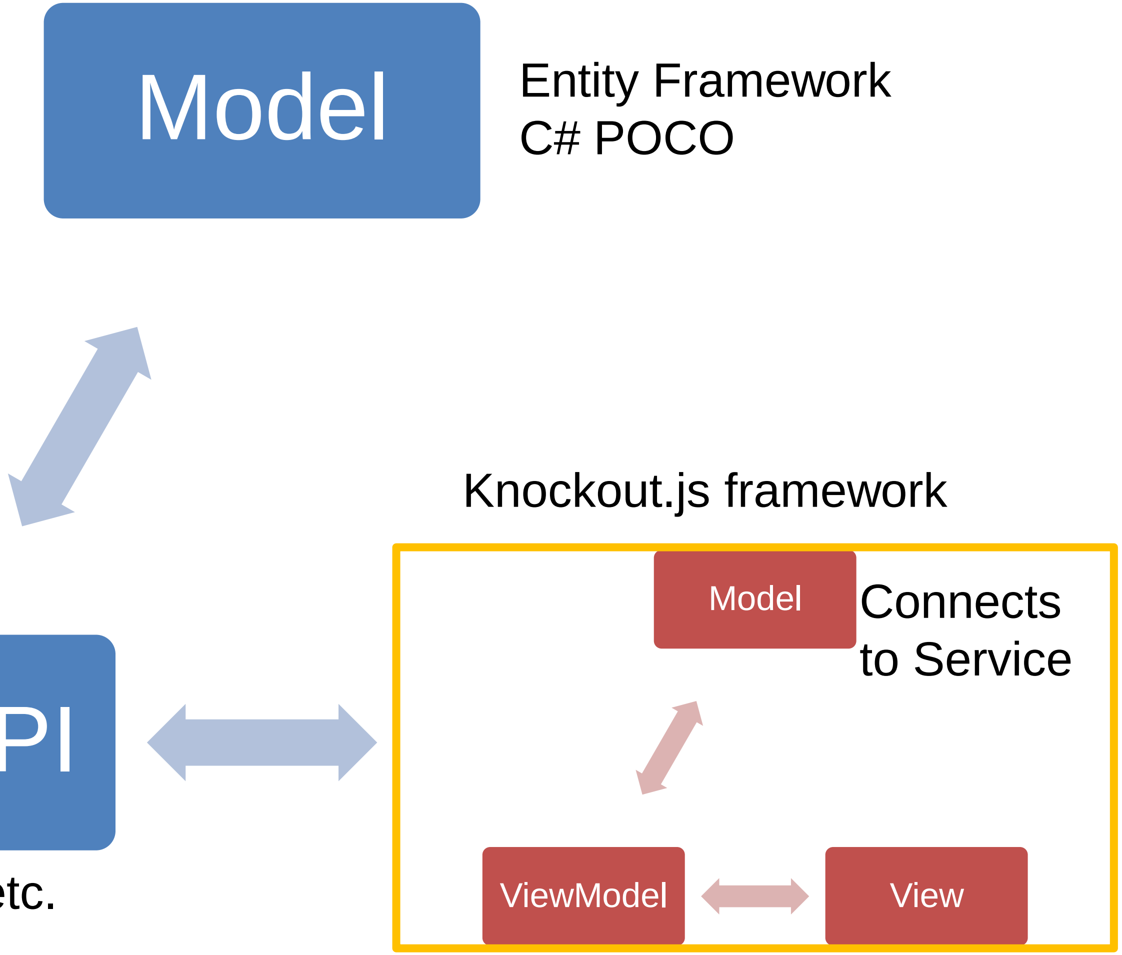
Solution

We put an MVC in our MVC
So we can MVC while we MVC



(Technically, MVVM)

Separation of Concerns



Model

Entity Framework
C# POCO

Knockout.js framework

Model

Connects
to Service

ViewModel

View

Observable
Properties;
Actions

Static HTML 5
With bindings

Server-Side
REST Service
Data Access
Business Logic

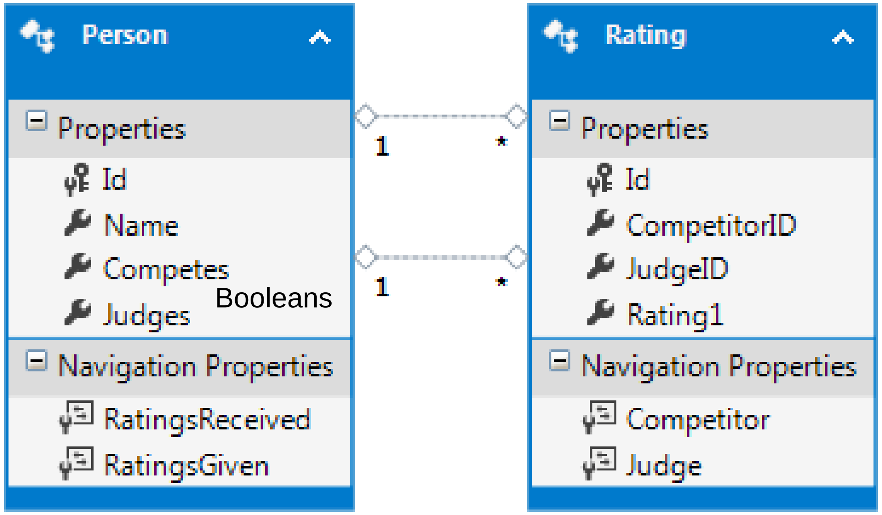
Web API

Can serve thick client, etc.
Not HTML specific

Tech Demo

A judging/rating system

- **Each person can be judge, or competitor, or both**
- **All judges can judge (rating 1 to 10) all competitors**
  - **Except if both, person can't judge themselves**
- **Display overall rankings based on average rating**
- **Assume all persons "pre-loaded" in database**

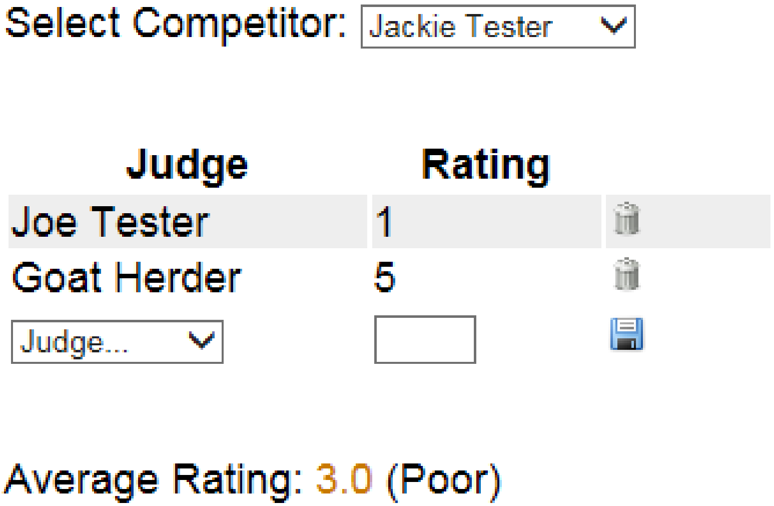


I changed the name of the navigation properties to be more clear.

Also, the rating is called "Rating1" in EF because it conflicts with the type name.

Tech Demo: Workflow

- Select Competitor from dropdown
- Existing ratings shown, may be deleted
- If any judges have not judged, have a line to enter new rating
- If two or more ratings exist, show average rating for this competitor



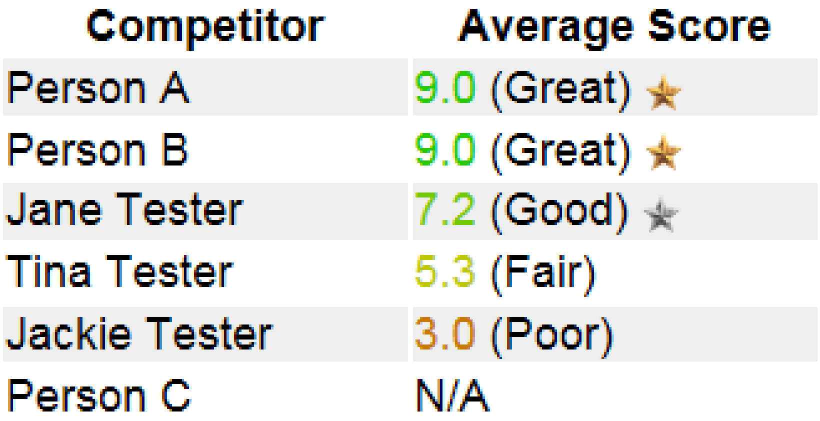
Select Competitor: Jackie Tester

Judge	Rating	
Joe Tester	1	🗑
Goat Herder	5	🗑
Judge...		💾

Average Rating: 3.0 (Poor)

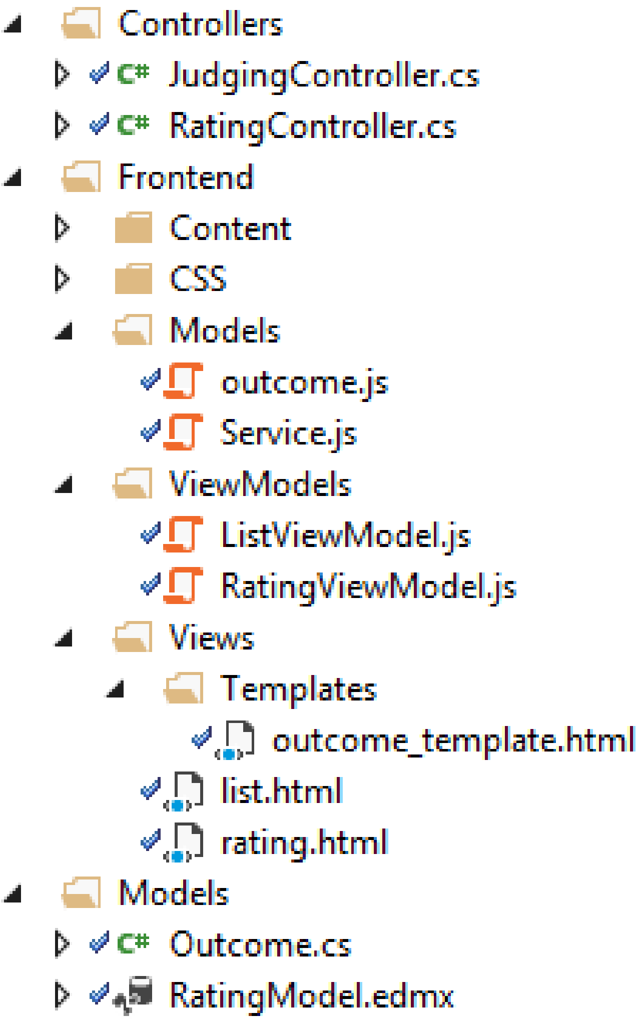
Tech Demo: Workflow

- View all average ratings
- Sorted by rating
- All competitors with equal top rating get “gold star”
- All competitors with equal second rating get “silver star”
- Competitors with no ratings don’t show a score (counted as 0)



Competitor	Average Score
Person A	9.0 (Great) ★
Person B	9.0 (Great) ★
Jane Tester	7.2 (Good) ★
Tina Tester	5.3 (Fair)
Jackie Tester	3.0 (Poor)
Person C	N/A

Let’s Dig In!



```
◢ Controllers
    ▷ ✓ C# JudgingController.cs
    ▷ ✓ C# RatingController.cs
◢ Frontend
    ▷ Content
    ▷ CSS
    ◢ Models
        ✓ outcome.js
        ✓ Service.js
    ◢ ViewModels
        ✓ ListViewModel.js
        ✓ RatingViewModel.js
    ◢ Views
        ◢ Templates
            ✓ outcome_template.html
        ✓ list.html
        ✓ rating.html
◢ Models
    ▷ ✓ C# Outcome.cs
    ▷ ✓ RatingModel.edmx
```

Code: Server Side

- **Controllers**
  - **Return data, not HTML**
  - **Web API handles JSON mapping**
  - **Uses HTTP verbs and response codes**
  - **Only an API**
- **POCO Models**
  - **Perform business logic**
  - **Fully testable**

Server side is just API Service.
The "Amazon.com" Way:

All teams will henceforth expose their data and functionality through service interfaces.

Teams must communicate with each other through these interfaces.

There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

It doesn't matter what technology they use.

All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

Jeff Bezos, ~2002

NO HTML specific behavior

Code: Client Side

- Plain HTML (View)
  - Could make header/footer with MVC views
  - Uses knockout data binding (like razor in MVC)
  - Uses knockout templates (like partial views in MVC)
- View Model
  - Methods hit abstract “service”
  - Stores data
  - Testable
  - Many more KO features exist!
- Model
  - Connects to server API
  - Performs logic specific to this client implementation
    - Testable

In Production

MISSING from this demo

- **Error Handling (see “fail” functions in ViewModel)**
- **Unit Tests**
- **Data Validation (Client and Server)**
- **JS Bundling (ASP.NET feature)**
- **Page Templates (Header/Footer)**

Conclusions

- **Web Applications are "applications" in every sense of the word**
- **1st class frameworks make it**
  - **Tolerable**
  - **Maintainable**
  - **Testable**
- **Separation of concerns**
- **Remember: In the grim darkness of the ~~far~~ future, there is only JavaScript**

not-so-far

Knockout.js

Tech Demo

Questions? Comments?

Get the code: http://www.kolls.net/wsdot/ko2014

Steve Kollmansberger
Lead Developer, Bridge Preservation Applications

WSDOT Developers Conference
January 14, 2014

Washington State
Department of Transportation