

# ASP.NET MVC and WURFL

Dino Esposito  
<http://software2cents.wordpress.com>  
@despos



**pluralsight**   
hardcore dev and IT training

# Key Points

**Display modes** in  
ASP.NET

WURFL Nuget package

**WURFL** in ASP.NET

# Taking the Definition of a Web Page Further



## The multi-device perspective

- Page `default.aspx` explodes into a slew of pages: `default.smartphone.aspx`, `default.tablet.aspx`, ...
- Logical page might display in various ways according to predefined rules



## Built-in infrastructure for view routing in ASP.NET MVC

- One action, multiple views
- Display modes

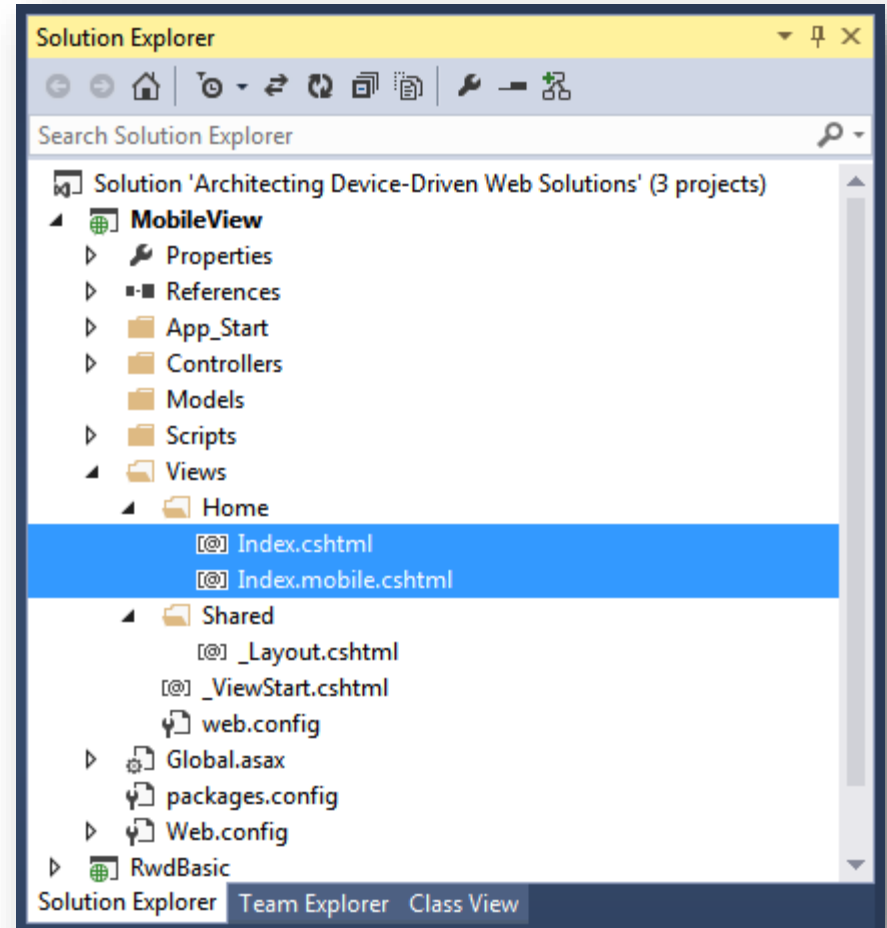
# Display Modes (Simple Experiment)

**Create a plain ASP.NET MVC application**

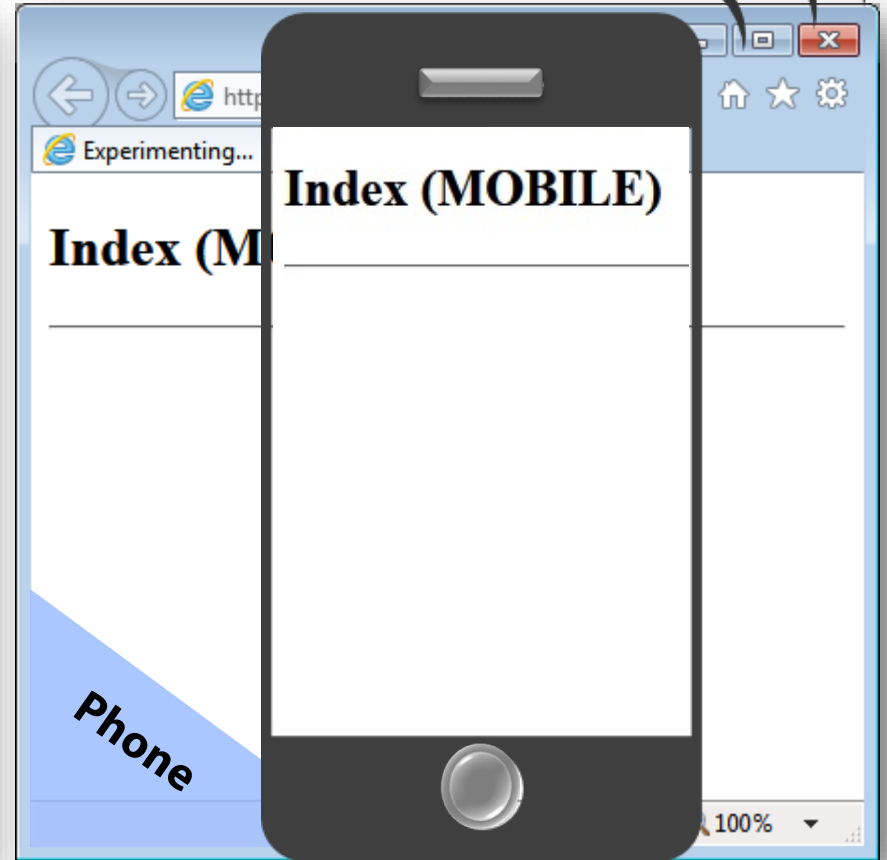
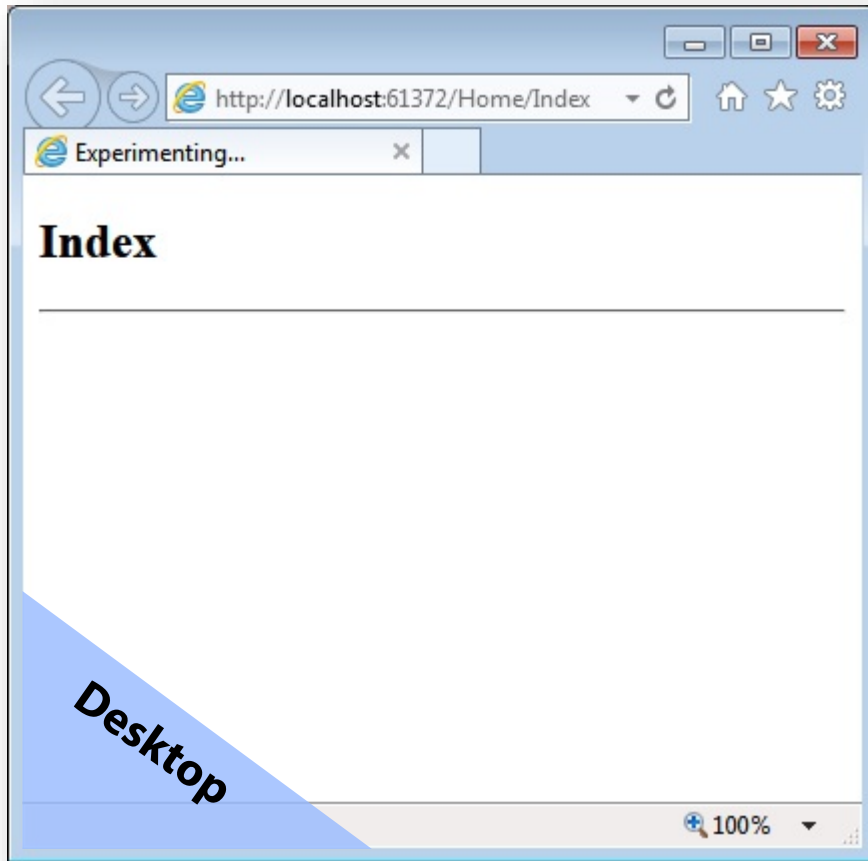
**Add an additional Razor view file**  
(index.mobile.cshtml)

**Configure the test browser to look like a mobile browser**  
(F12 in Internet Explorer)

**Invoke URL**  
(/home/index)



# Display Modes (Simple Experiment)

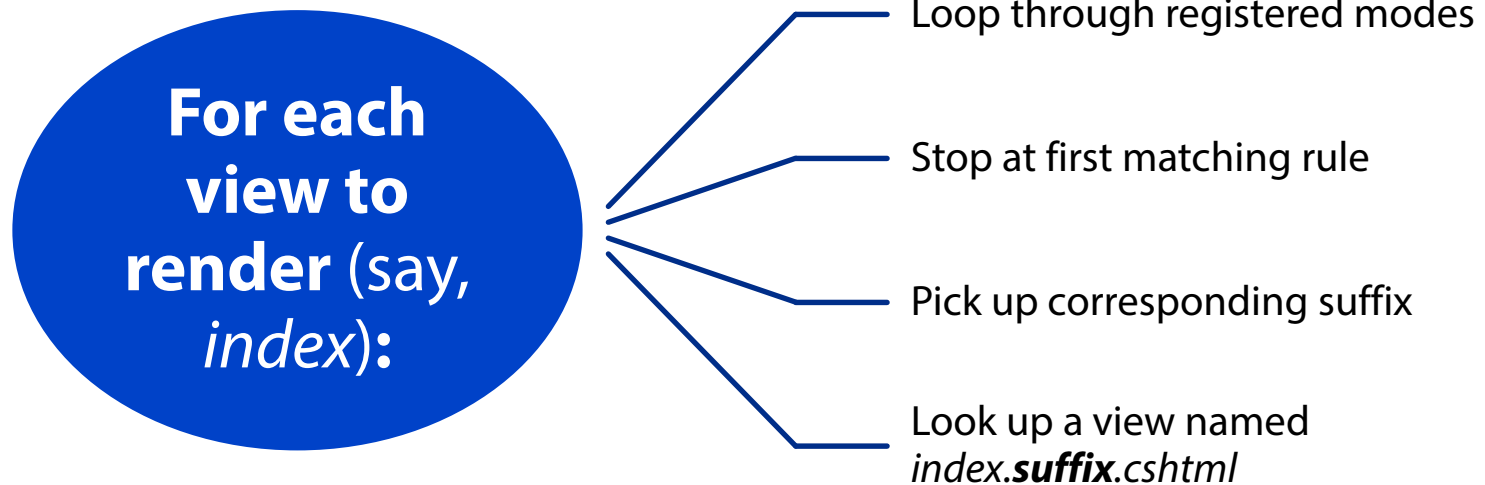


**Out-of-the-box**

# No Tricks and No Illusion

## ■ Display Modes

- Made of **suffix name** and **matching rule**
- Ad hoc provider to extend the ASP.NET MVC view engine



# Display Modes: Implementation

- **Class** `DefaultDisplayMode`
  - `Property ContextCondition`
  - Define matching rule that would trigger the mode
- **Matching rule delegate**
  - `Func<HttpContextBase, Boolean>`
  - Purpose is analyzing the HTTP context of the current request
  - Return a Boolean answer to the question: should this display mode be used to serve the current request?

# Display Modes: Implementation

```
var mobileMode = new DefaultDisplayMode("mobile")
{
    ContextCondition =
        context => LogicThatDeterminesWhetherThisIsMobile(context)
};
```

```
// Clear all predefined modes
DisplayModeProvider.Instance.Modes.Clear();

// Add your own display modes
DisplayModeProvider.Instance.Modes.Add(mobileMode);
:
```



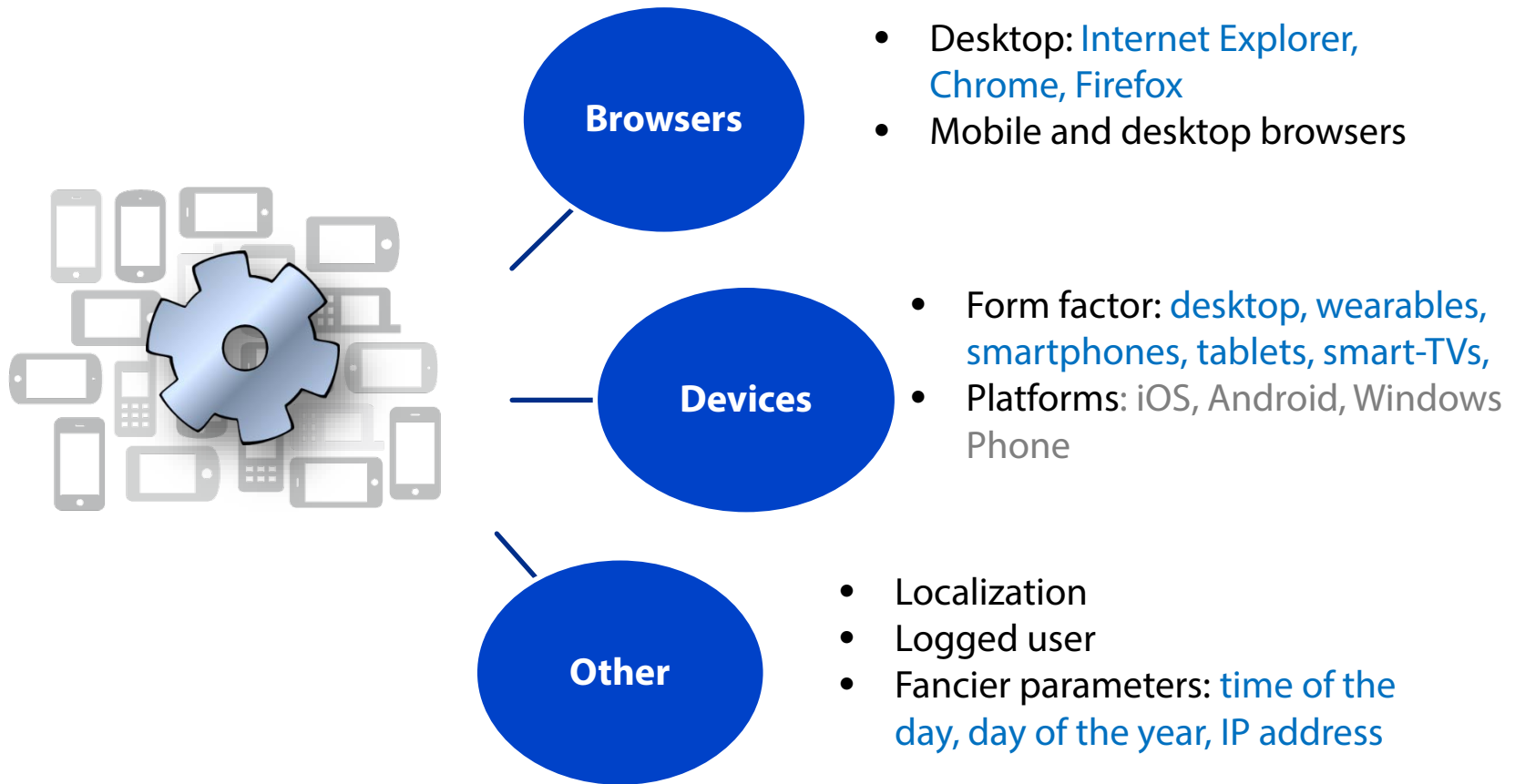
# Display Modes: Implementation

```
protected void Application_Start()
{
    // More configuration steps here
    :

    // Added to make the site device-aware.
    // (Disable this to get back to default behavior.)
    DisplayConfig.RegisterDisplayModes(
        DisplayModeProvider.Instance.Modes);
}
```

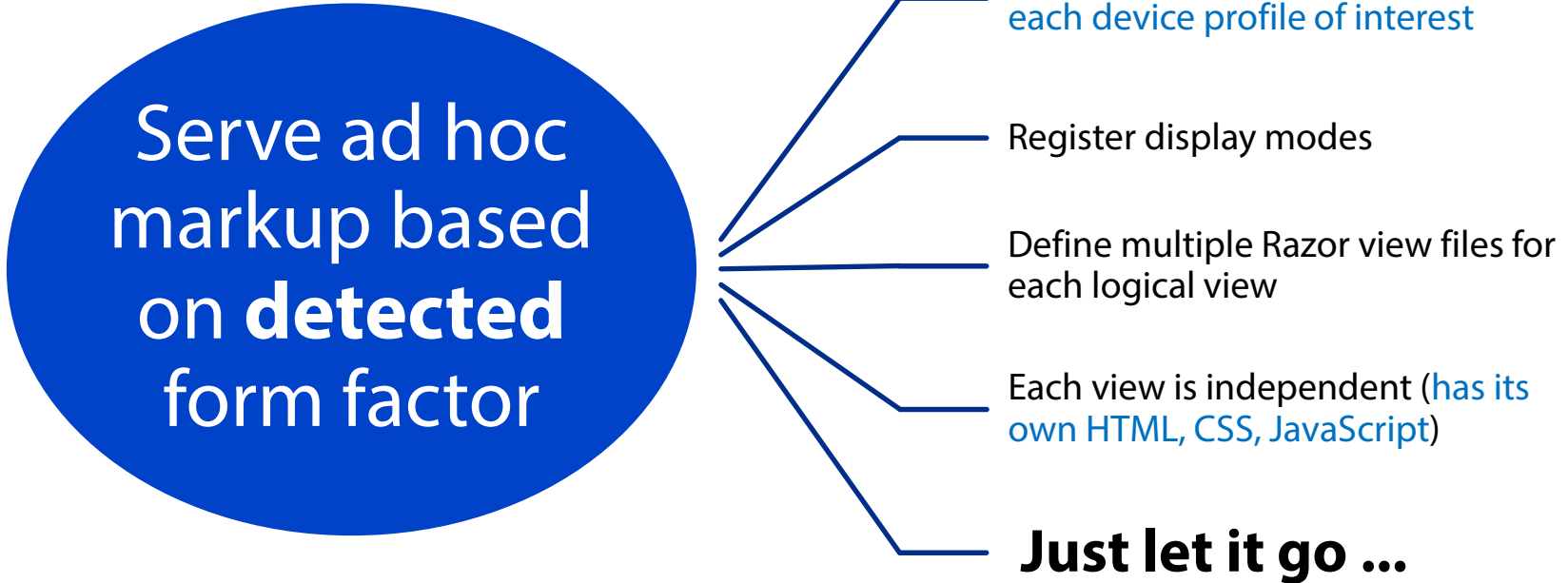
# Display Modes: Usage

Let you to switch between views based on run-time conditions.



# Display Modes: Usage Made Here

Let you to switch between views based on run-time conditions.



Serve ad hoc  
markup based  
on **detected**  
form factor

Define a display mode instance **for each device profile of interest**

Register display modes

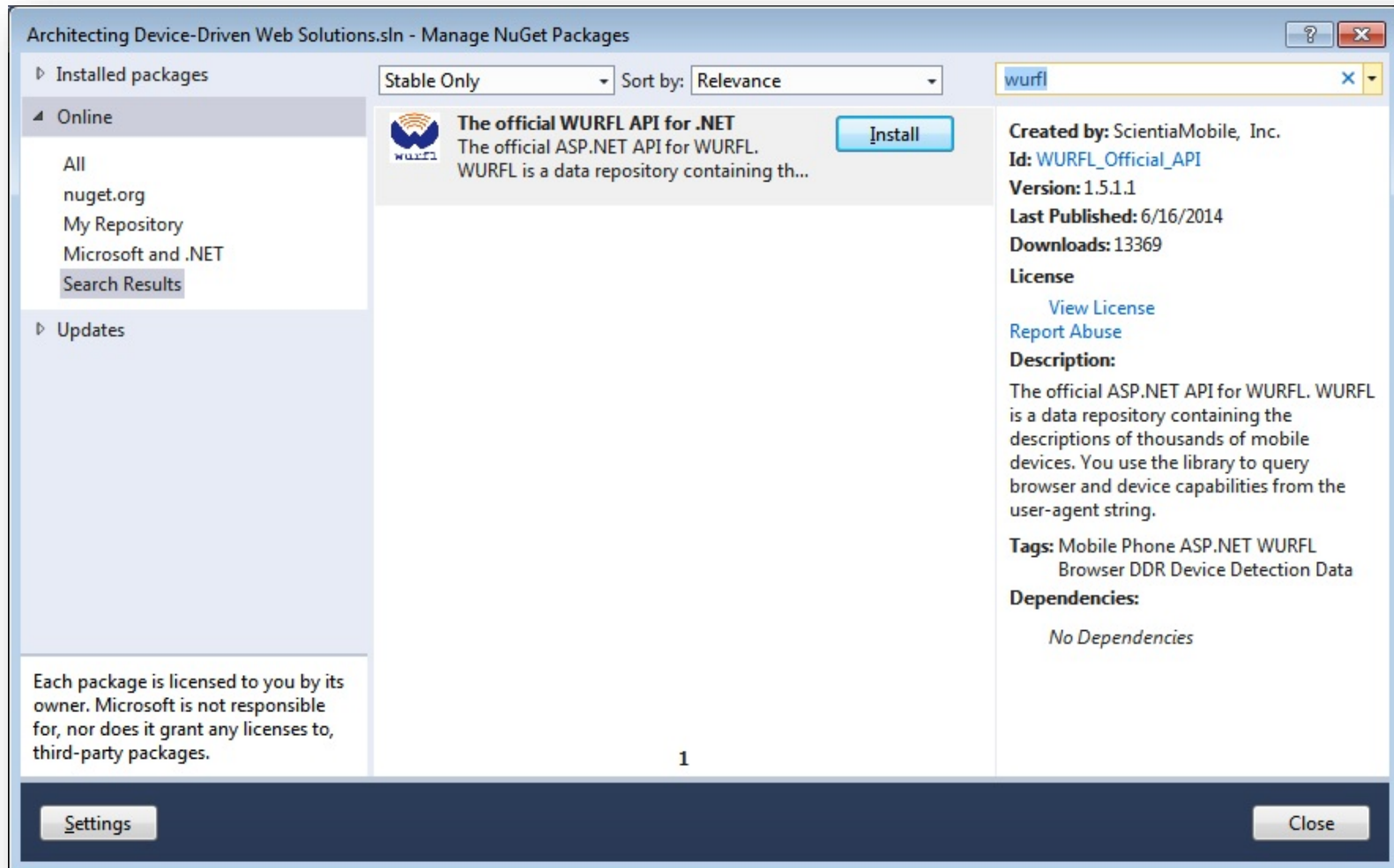
Define multiple Razor view files for each logical view

Each view is independent (**has its own HTML, CSS, JavaScript**)

**Just let it go ...**



# But Before Everything Else...



[http://www.nuget.org/packages/WURFL\\_Official\\_API](http://www.nuget.org/packages/WURFL_Official_API)

# DEMO **About** ASP.NET MVC and WURFL



**Display Modes**

**WURFL on-premise**

# DEMO About ASP.NET MVC and WURFL



WURFL.js

# Summary of WURFL in ASP.NET



## Server-side detection

- Manage DDR yourself
- Fastest response; built-in caching of device profiles
- Each device gets its own markup (ad hoc use-cases and layout)



## Client-side detection

- Free JavaScript service; no maintenance on your own
- Get JSON data about form-factor of the device
- Building the UI is up to you
- Typical scenario: calling WURFL.JS from ready()



# Summary

- **DDR is key to detect the form-factor**
  - Various options
  - May **not be** free, but it's the **cheapest** option
- **Serving ad hoc markup**
  - Not the same as just ad hoc **rendering** of markup
- **One ASP.NET MVC project**
  - Built-in display modes and Razor views
  - One set of controllers and one set of view model objects

... well, sort of ...