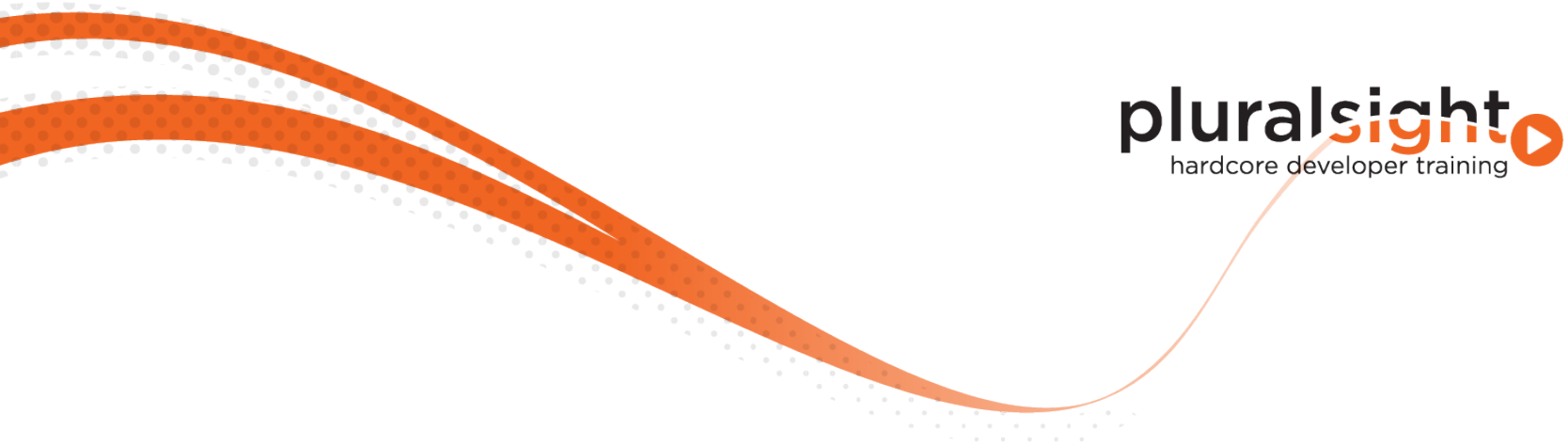
Nancy Views and View Engines

Richard Cirerol

http://codeprogression.com

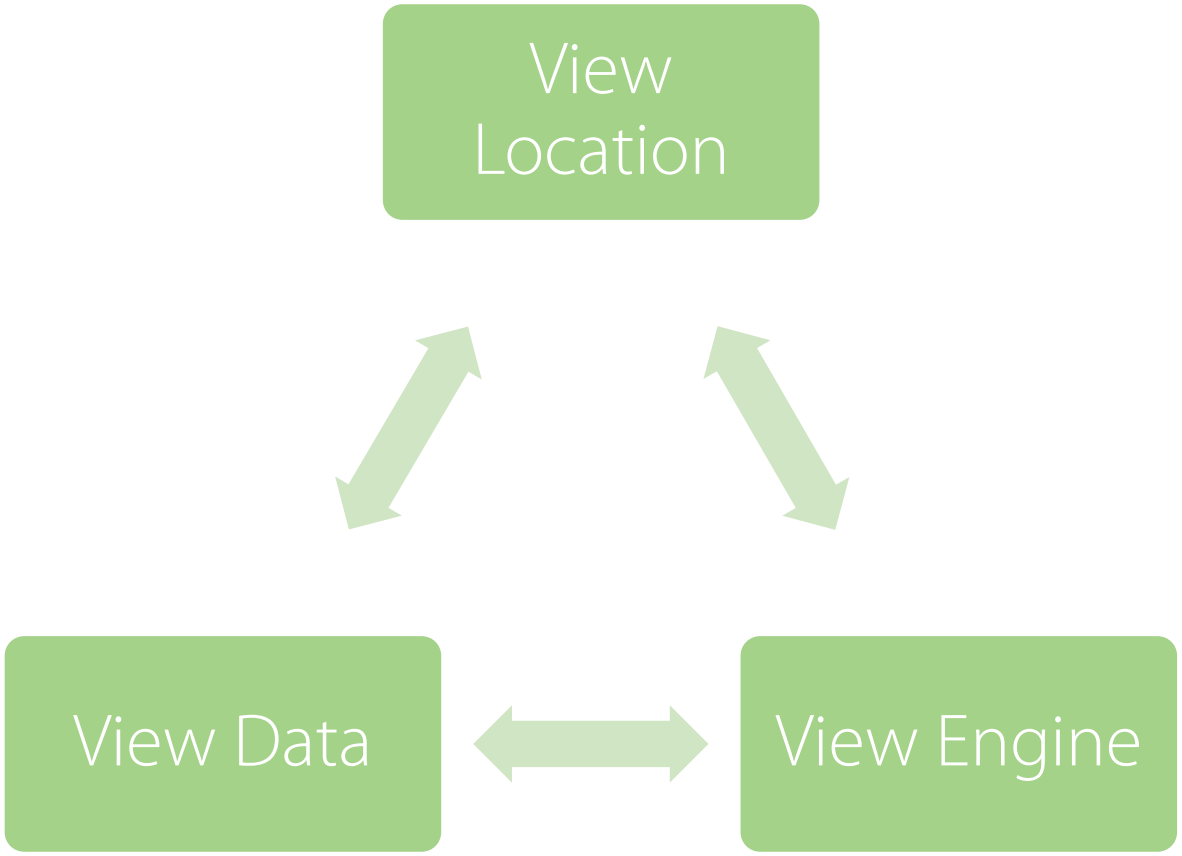


Views and View Engines

- **Recap**
  - Static view
  - Pure data

- **For user-facing web applications, we need:**
  - Dynamic views
  - Contextual data
  - Templated output

Views and View Engines



NANCY

View Location

- **Styles of handling view-data-engine interactions**
  - Tightly-coupled – ASP.NET WebForms
    - The view and code are tightly bound
    - Layered architecture is more difficult
    - One default view engine

View Location

- **Styles of handling view-data-engine interactions**
  - Prescribed convention – ASP.NET MVC
    - Views mirror the code
      - By default, a view location is derived from the controller/action couplet
    - Overrides are possible
      - In the routing tables
      - In the individual actions
    - Data passed via:
      - Strongly-typed object
      - Bag/dictionary object
    - Multiple view engines built in
      - Razor
      - Web Forms

View Location

- **Styles of handling view-data-engine interactions**
  - Multiple conventions – Nancy
    - Separation of logic and presentation
    - Flexible file system structure
    - Flexible view engine conventions
    - Pass data via:
      - Object
      - Bag/dictionary

View Location

- **Built-in conventions:**
  - The Root Convention
  - The View Folder Convention
  - The Views and Module Path Convention
  - The Module Path Convention
  - The Module Name Convention
  - The Views and Module Name Convention
  - Your Convention Here

- **For more information on the default conventions:**
  - *https://github.com/NancyFx/Nancy/wiki/View%20location%20conventions*

Demo: View Location

Requirements:

- **Revisit the Root Convention as we applied it in the first module**
- **Move the view to a Views folder (the View Folder Convention)**
- **Use the Views and Module Name Convention**
- **Create our own convention**

View Data and Engines

```
// Finds the view by name only
View["index"]

// Finds the view by name and extension
View["index.cshtml"]

// Finds the view by name and passes in the model data
View["index", productsModel]

// Passes in the model data
// Finds the view by using the model's type name
// Removes "Model" from the type name if it exists
// If products type is ProductModel,
//   name will resolve to 'product'
View[products]
```

View Data and Engines

- **The Super Simple View Engine**
  - Default view engine
  - Similar to Razor
  - https://github.com/grumpydev/SuperSimpleViewEngine

  - Limitations:
    - Does not support nested collections

NANCY

Demo: The Super Simple View Engine

Requirements:

- **Display a list of courses using the Super Simple View Engine**
- **Use a specific view name**

View Data and Engines

- **Using the Razor View Engine**
  - Same as the default view engine from ASP.NET MVC 3
  - Available as Nuget package
  - Build project immediately after installing package
    - Intellisense
    - Web.config references
- **Can use multiple view engines**
  - May need to specify file extension explicitly

Demo: The Razor View Engine

Requirements:

- **Display a single course using the Razor View Engine**
- **Use the model as the view name**

Summary

- **How Nancy locates views**
- **How to specify which view to retrieve**
  - View name
  - Model
  - Both
- **View Engines**
  - Super Simple View Engine
  - Razor View Engine
  - DotLiquid
  - NDjango
  - Nustache
  - Spark