Developer Introduction to Infusion Framework

Introduction

The Infusion framework has a lot going on, so here's an attempt to boil it down to some basic concepts with you, the developer, in mind. You might be at a place where Infusion sounds like it will be useful to you, and you're ready to start trying it out, but need a bit more direction on where to start and how to approach it. There are a few things to keep in mind before you dig in - hopefully this will help set the stage for getting into the code.

Infusion's Objectives

To illustrate some of the strengths of component-based development, let's consider building a bar-graph widget for a website that displays poll data and think about some of the big picture goals.

Re-using code - Perhaps this isn't the first bar graph for poll data you've ever made, and certainly not your last. It would be nice to be able to code just one bar graph and easily customize it to fit a particular situation. Maybe just send your bar graph a hunk of data, a colour scheme, a size, and voila, create a new bar graph that fits the situation.

Re-purposing functionality - Maybe you're tired of bar graphs and you want data to be displayed as a pie graph or treemap instead. It would be nice to be able to use some of the functionality within our bar graph to help create other types of charts.

Relating data - Our bar graph is obviously tied to the poll's voting system - when the data changes, our graph should automatically change too. Perhaps we want other parts of our site to change as the data is updated as well.

Using events - It's possible we would want to know when the bar graph was finished drawing so we could trigger another function within the site to begin, or listen for another component's event before starting functions of our own.

Adding Accessibility - By creating components that are modifiable to different situations, with output styled accessibly, we can create an experience usable by all. For our bar-graph, we would make sure the data could be read comprehensively by screen readers for example. This may require modifying options to alter display choices.

Together these objectives inspire **flexible, inclusive design** through modular, highly customize-able, accessible code.

Infusion's Building Blocks

Components

Infusion is built around the idea of components, and components working together to get something cool happening. They'll be explained in a bit more detail here, but for starters think of a component as highly configurable object that does some stuff. It typically has a few key pieces: options, events, a model, and functions.

Options are choices. Most of the out-of-the-box components that ship with Infusion are pre-configured with default options so the component works in a certain way, but you as the developer are able to easily over-ride these defaults with your own options.

Events are happenings. If your component starts doing something, finishes doing something, needs something, an event can be sent out to notify other components of these occurrences. Likewise, your component is able to listen to events coming from other components, and trigger functionality accordingly.

The model keeps track of data and its state. It is the master record keeper, able to populate components with live information. It also means the component can focus on functionality without worrying about data management.

Functions take options, events, and the model, and do stuff. The meat of your component will lie in its functionality, possibly using options to affect output, firing or listening for events before running actions, reading and/or modifying the model, and so on.

Sub-components

Many components are built up of, or make use of, other components called subcomponents. There's nothing too special about a subcomponent - it's basically a component that is used by another component. Therefore it's defined by its context, not its makeup.

Inversion of Control - a new feature in Infusion to make using components together easier and cleaner. Instead of a parent component having to directly initialize subcomponents, IoC lets you declare subcomponents and their options through its instantiator. The instantiator will add them to a "resolution stack" where then a component can ask IoC to pick out information from components in the stack as it needs, without having the responsibility of creating or even knowing too much about them.

Pre-built components

The Infusion package ships with several pre-made reusable, accessible user interfaces such as Uploader, Progress Bar, Inline Edit, etc. We call these components as well. It may seem to muddy the waters when defining what a component is, but now that you know about what makes up a component, and what are subcomponents, it's easy to see that these interfaces are simply a combination of functionality into bigger components with options, events, models, and functions.

It might be helpful to think of a component as a kit that comes pre-assembled as one thing, but is also build-your-own. You can use the pre-set defaults as is, or change them to work how you would like - the nature of a component let's you modify anything and everything about it to get it the way you want.

Infusion's Framework Features

In addition to Infusion's basic component management abilities, it ships with several important and useful core features you can use in your components:

ChangeApplier - handles the function of data binding and coordinates model updates. This controls and mediates the modification and access of data among different components.

DOM Binder - allows a component to quickly locate named elements in the DOM. Instead of dealing with class or id names of HTML elements in your code, the DOM Binder lets you locate associated selectors specified in your component's options.

Renderer - allows users to create user interface templates in pure HTML, and render the pages entirely on the client side given a data model.

Fluid Skinning System - using context-rich class names in your templates, FSS not only helps to style your page, but adds flexibility to your layout, allowing it to be modified by the UI Options component for example, or for mobile or print versions.

Events - a system of firers and listeners

Types of components

Different combinations of component pieces

Little Component - just options

View Component - options, dom binder, events

Renderer Component - view stuff + renderer

Evented Component - little + events

Start Coding

Now it's time to dig in. The best place to start will probably be to create a component from scratch. Let's try making the bar-graph component mentioned above over at Creating a Component.