Routing and Templating Feedback

This document collects some feedback from a private demo of the router and templating engine. The term “they” refers to the individuals to which the demo was given. “I” refers to Rob Eisenberg and “we” refers to the Angular 2.0 team.

# Router Feedback

## router-view-port

The *router-view-port* directive doesn’t follow the ng1 element naming convention, where all Angular directives begin with *ng-.* They thought this name should be changed to something like *ng-view-port* or *ng-router-port* for consistency and familiarity to existing Angular users.

## route config settings

When configuring the router, they wanted to be able to include arbitrary data as part of the route config in a way that they knew would not conflict with the route config data structures now or in the future. They recommended adding a “settings” or “bag” property that the router would not touch and simply make available as part of the data model for developer use. Examples of uses for this included custom security data used to validate routes as well as icon, caption and category information used in rendering a custom navigation ui.

## route config property names

They felt that the term “pattern” was too generic and should be changed to “route”. They also felt that “componentUrl” should be shortened to “component”.

# Templating Feedback

## ngRepeat

I mentioned that we didn’t have the full directive implemented yet so we didn’t support the microsyntax for iterator variable naming. In place of that we had temporarily hardcoded the “row” variable. Feedback was that this was a *good* idea. They liked having the ability to just specify the collection in the binding with a fallback iterator variable if one was not supplied. They recommended giving it a bit more special name though, such as “$row” or “$item”.

## bind-

There was **very strong** pushback against this syntax. They felt that this made the HTML difficult to read. (Also, they didn’t like the extra typing.) I’m not sure if this was a deal breaker for them, but they really did not like this at all. I mentioned that we had considered a bind operator such as “:=” early on but that that would not work in IE. However, we discovered later that other symbols would work fine, such as “&=” and that I thought we should revisit this issue. They said they would greatly prefer this to “bind-” and if we could choose a symbol that related to the scope symbol from ng1, that would be great too (though not that big of a deal).

## component model

The current state of the component model and/or requirements of the templating system on developers was seen as a complete deal breaker. Both individuals indicated they would not use Angular 2.0 if this wasn’t fixed. They were very concerned. The primary problem was due to the fact that there didn’t appear to be any way to use different views with the same controller or to use the same view with multiple controllers. Both of these use cases were critical “we cannot live without” issues for them. Related to this, they wanted a way to name their files according to different patterns, but without having to provide the templating engine with extra information for every component. They also wanted to be able to programmatically determine the view for components at runtime. I mentioned that I hadn’t studied the new templating in depth since the last set of big changed, but that I think this could be solved by allowing an extensibility point for the developer that would pass the component token (and possibly some context) and allow the developer to return an object with a controller url and a view url. This would allow writing custom conventions or changing views based on the user, device, screen context or other runtime data.

*PERSONAL NOTE: It would be even better if we could load the controller first and then determine the view. This would even allow for annotations placed on the controller or custom callbacks to control the view, if desired.*

*PERSONAL NOTE 2: I didn’t get the impression that this view/controller location flexibility was needed for Angular exported Web Components, but that it was critical for application-specific components.*

*FROM WARD:*

*Folks have their own ideas about their “standards” for naming and locating controllers and views. They might put them in separate folders: viewModels/views or controllers/views or … maybe (like me) they like them together in the same folder. The folders might be relative to something else. The file names might differ only in the extension (foo.js and foo.html). The View might have a “tmpl” particle in the file name to distinguish templates from full HTML (“foo.tmpl.html”). The variations are endless. And that’s before we get to the exceptions (“shell” is often unlike the others).*

*People love conventions … especially when THEY get to define those conventions. That facility seems altogether missing in the example we saw.*

*I have very specific examples wherein I must determine the view dynamically on the basis of the information in the route. [client] has a VM to display pizza parlor products. The VM’s responsibilities and surface are pretty much the same across the product line but the views are different for Pizzas, Salads, Drinks, etc. I have no need of a PizzaVM or SaladVM and if I was trying for a fully general approach to product presentation in which there might be 100s of product categories (think Health Care), I’d not be keen on having to write a new VM for each category.*

## double directive attributes

Currently, when databinding a directive, you must add the directive twice. Once with no binding expression and once with the *bind-* prefix and the binding expression. I believe this is just a bug in the templating engine. They stated that if it wasn’t this was a real big problem. Not sure if it was a deal breaker, but it made them very, very unhappy.

## component export conventions

A minor annoyance was that ComponentDirective annotations had to be placed on classes which were the only export from a module. They thought that if the router requested the component from the module, and that there was only one export then that should assume that export was the component without the developer having to write more code. Also, having to provide a selector for a component that would never be explicitly used in HTML seemed like more meaningless work to them. I mentioned that I thought this wouldn’t be hard to do.

## ng-config

They immediately noticed that an *ng-config* attribute had to be placed on every single template. They didn’t like this at all and felt that they should be able to configure a general set of directives at the application level and only need ng-config to bring in directives which were not declared there. They saw this as a violation of DRY.

## bootstrapping

There were a lot of questions about how to bootstrap the app and set up initial configuration. I mentioned this wasn’t all worked out yet. They didn’t feel that the current implementation was entirely adequate.

FROM WARD:

Moreover, the product should ship with out-of-the-box config or some kind of startup component that would make it easy for people to jump on board with a minimum of fuss. I know you’re trying to componentize everything. Fine. But don’t minimize configuration for the first time experience.