**Angularjs开发一些经验总结**

前端范qianduanfan.com前端人的快速响应问答社区，汇聚了大量的前端工程师

在去年到今年参与了2个使用Angularjs作为客户端开发框架的项目开发。主要利用asp.net web api作为restfull服务提供框架和angularjs结合。Angularjs作为html的扩展，旨在建立一个丰富的动态web应用，通过Directive建立一套html扩展的DSL模型，利用PM模式变形MVVM（在网上很多称MVC模式，本人认为在angular0.8是属于经典MVC模式，但在1.0把scope独立注入过后，更倾向于MVVM模式，这将会后续随笔中写道）简化前端开发和使得前端业务逻辑得以分离，view和表现逻辑的分离，更便于维护，扩展。Angularjs本来就是采用TDD开发的，提供了一套单元测试组件和End 2 End的测试框架。Angularjs的的强大之处在于提供了一套内似WPF，Silverlight的强大数据绑定和格式化，过滤组件，这也是MVVM模式所必备的条件；再加之IOC的注入机制，使得不能业务逻辑的分离，服务代码的更大程度抽象重用。

     在这节随便中将讨论的angularjs开发的一些基本准则，为什么会有这篇随便呢，因为看见一些项目对于angularjs的乱用。

  1：不要一个page一个God似无所不能的controller包含所有页面逻辑。

        Angularjs ng-controller旨在将业务逻辑的区分，更推荐按照业务逻辑的划分controller，做到业务功能的高内聚，controller的单一原则SRP。

  2：View中包含尽量少的逻辑。

       就像jsp，asp这类服务端模板引擎一样，我们应该把尽量少的逻辑放在view中，因为这样会导致view和逻辑的紧耦合性，view在软件开发中是最易变化的，而表现层逻辑却相对于view是相对稳定的行为。同时也导致的view中的逻辑不能被自动化测试，持续集成所覆盖，这将导致以后修改重构和模块的集成的痛苦。很明显的就是太多的angularjs的ng-switch，ng-when和页面计算表达式等等。

3：注意一些特殊的节点式的angularjs directive，因为在IE7上这是不被认识的，因为IE的严格XML模式。如果你想make ie7 happy，

     1：请注意导入json2或者json3的js

     2：xmlns:ng命令空间和节点element式directive。

<html xmlns:ng="http://angularjs.org">

<head>

<!--[if lte IE 8]>

<script>

document.createElement('ng-include');

document.createElement('ng-pluralize');

document.createElement('ng-view');

// Optionally these for CSS

document.createElement('ng:include');

document.createElement('ng:pluralize');

document.createElement('ng:view');

</script>

<![endif]-->

</head>

  4：在controller和service中绝对不能出现html的DOM和CSS代码。

      这会导致逻辑的混杂耦合，对于angularjs自身的绑定对html操作，很多时候你会分不清是view的影响源，导致修复bug，和新增功能，重构的艰难，常常出现很多的诡异行为。最好的实践模式则是把必须的dom，css操作移向angular的Directive，或者view中。在angularjs模式中只有directive和view才能出现dom和css的逻辑操作。

  5：controller中公用的逻辑推向service（factory，value，config)，采用IOC的注入，提高代码的重用度，修改的单一点，开闭原则。

  6：controller应该只包含业务逻辑，对于数据模型的格式化过滤尽量交给angular框架filter等处理。

7：viewmodel中最好建立一个通用属性比如vm，它承载view渲染的最小量化model，对于model的变形事件则在vm之外scope之上。这才是MVVM推荐方式。事件相当于WPF中的command，负责模型事件的传递修改模型，从而从模型的改变通知view的强制更新（WPF中model必须实现INotifyPropertyChange接口）。同时这样vm属性也便于数据的填充和收集回发服务端。

8：IOC注入优先，注有助于良好的设计，逻辑的可重用和单元的可测试性，面向对象的“开闭原则”，修改的单一点。

9：良好的分层设计，对于服view的交互采用controller通过viewmode（scope）的推送，服务器的交互推向service层次，利用angularjs的$resource或者$http获取更新数据model，与服务端交互。层次属于纵向分割，将一类功能逻辑的接口放在一起，架构层次，而model则从业务的逻辑横向分离。

10：服务端的服务的接口需要考虑表现层客户端的应用提供，这是一个良好的SOA服务设计的准则，这里不用多余的描述，具体请移步[架构篇](http://www.cnblogs.com/whitewolf/category/379884.html" \t "\)。

11：如果你的公司应用了敏捷开发则，TDD的开发是必备的，angularjs本也是解决javascript测试驱动开发项目。

   最后想说说angularjs也不是银弹，并不是万能的，不是所有的项目都适合应用，它适用于CRUD的应用系统，内置了一些默认规则（惯例优先），对于表现层频繁交互的项目不适用，对于一些特殊的项目比如spring hdiv的项目也不是那么友好，或者就是你希望兼容更多的IE8一下的版本的应用系统，同样也不实用。

本文由<http://www.focustar.net> 整理分享。