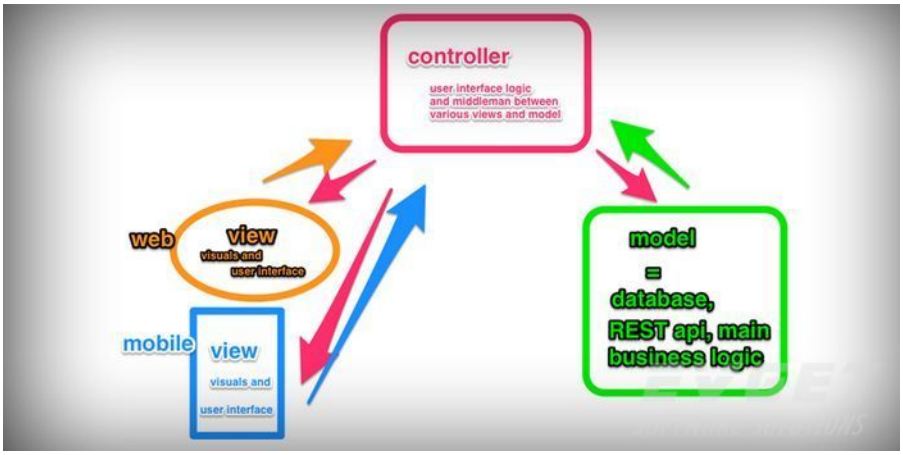
Angular1.x入门与精通 第2章：Angular MVC介绍

> 作者：徐礼文 2016/7/18 21:27:53

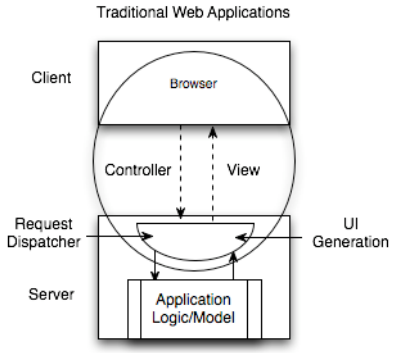
> AngularJS使用MVC架构来开发Web应用程序。要实现MVC，你只需拆分你的应用程序，其余全部由AngularJS管理即可。 MVC架构代表模型视图控制器，其中：



> 1. 在AngularJS中出于内存占用和性能的考虑，控制器只会在需要时被实例化，并且不再需要就会被销毁。这意味着每次切换路由或重新加载视图时，当前的控制器会被AngularJS清除掉。
>
> 2. 做为一个SPA应用，有时我们希望能够在路由更新后仍能保留前一个页面的部分数据， 这时我们可能的做法是将需要保留的数据放置到$rootScope中或者放到Cathe中， 这样做并非最好的方式，下面来看一种更具扩展性、更方便的方式即service。
>
> 3. 服务提供了一种能在应用的整个生命周期内保持数据的方法，它能够在控制器之间进行通信，并且能保证数据的一致性。服务是一个单例对象，在每个应用中只会被实例化一次（被 $injector实例化），并且是延迟加载的（需要时才会被创建）。服务提供了把与特定功能相关联的方法集中在一起的接口。

为什么需要使用JS-MVC框架？

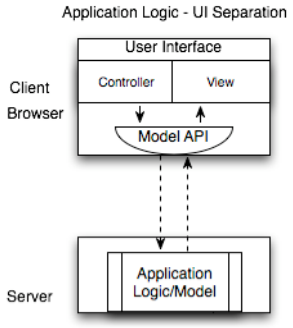
传统Web应用开发



> 传统模式处理业务请求全部放在服务端，前段只是页面交互（瘦客户端，胖服务端）. 这会有以下问题：

1. 分布式处理能力弱 – 服务器处理大量业务，性能堪忧.
2. 相应压力 – 传统应用的响应速度是个硬伤.
3. 开发复杂度 –C/S结构的应用开发是比较复杂的. 由于每次请求响应都涉及到交互设计，很容易出错。未解决该问题的框架也是层出不穷，可惜易用性有待考究.
4. 被攻击危险 – 混编业务代码和交互代码，增加了代码受攻击的概率.在复杂度很高的应用中更是不容易控制安全性。
5. 服务端的负载过大 – 所有客户端的请求都需要经由服务端处理，这意味着所有的session都要等待30分钟后才能被释放，这时客户请求早已处理完毕，但还在占用系统资源，大大降低了系统性能和伸缩性.
6. 离线处理 – 拥有离线处理能力是web应用的竞争力，尤其在处理大量客户端请求的应用中，离线处理部分业务更是不可或缺.
7. 互操作性弱– 由于混杂编写，代码逻辑很难分割，扩展功能变得复杂.

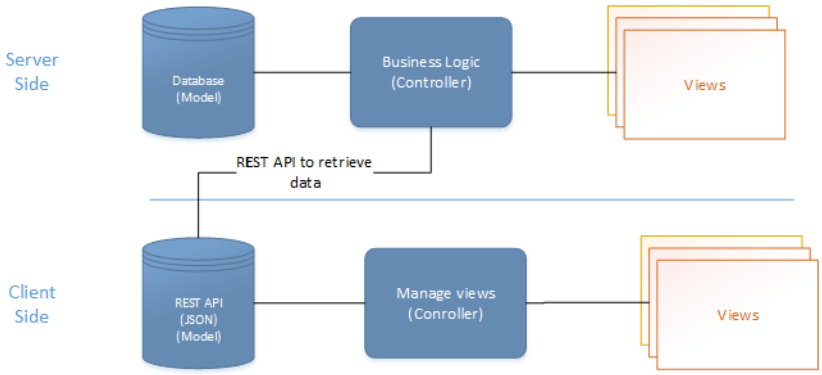
JSMVC Web应用程序

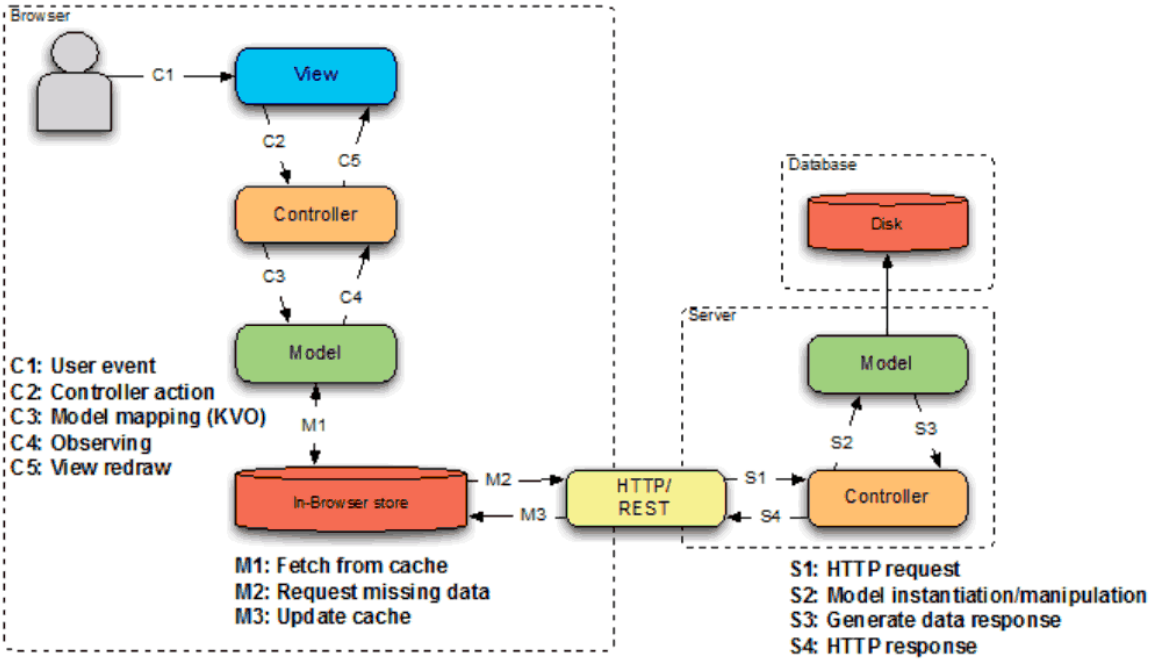


JS MVC web应用程序架构主要致力于将服务端的逻辑处理转移到客户端和实现Rich-Internet-Apllication 客户端web应用程序。client/server模型的处理逻辑和代码被委托给浏览器的好处是：

1. 可扩展性：很容易看到利用客户端处理在可扩展性方面的优势。服务器处理能力保持不变的前提下，应用被越多的客户使用，那么越多的客户端机器可以被使用（直到你购买更多的服务器）。
2. 实时的用户响应：客户端代码可以立即对用户的输入作出反应，而不需要等待网络传输。
3. 结构清晰的编程模型：用户界面可以有效地分离应用程序的业务逻辑。这样的模型为安全提供了一个更加简洁方法。所有通过用户界面的发出的请求，我们可以在数据通过各种接口前进行安全检查。使用复杂的分析流程会让安全分析变得更加复杂。另一方面，用清晰的web服务接口，有明确的网关安全工作和安全分析更简单直观，漏洞可以快速发现并纠正。
4. 客户端状态管理：在客户端维护临时会话状态信息可以减少服务器上的内存负载。这也允许客户利用更多的RESTful交互，可以进一步提高可伸缩性和使用缓存的时机。
5. 离线应用-如果大部分应用程序的代码已经在客户端上运行，那么创建一个离线版本的应用程序可以肯定将会变得更加容易。
6. 互操作性：通过使用结构化数据和最小限度的api进行交互，这样更容易连接额外的消费者和生产者与现有系统进行交互。

目前，较流行的一种包含客户端服务端的模式是 后端RESTful API 通过 JSON发送数据模型 客户端使用MVC模式 处理应用.




Browser
C1
View
C2
C5
Controller
C3
C4
Model
M1
In-Browser store
M2
M3
HTTP/
REST
S1
S4
Database
Disk
Server
Model
S2
S3
Controller
C1: User event
C2: Controller action
C3: Model mapping (KVO)
C4: Observing
C5: View redraw
M1: Fetch from cache
M2: Request missing data
M3: Update cache
S1: HTTP request
S2: Model instantiation/manipulation
S3: Generate data response
S4: HTTP response