WEB框架比较

WEB框架的实质

阿里巴巴中国 alibaba.com.cn

研究院

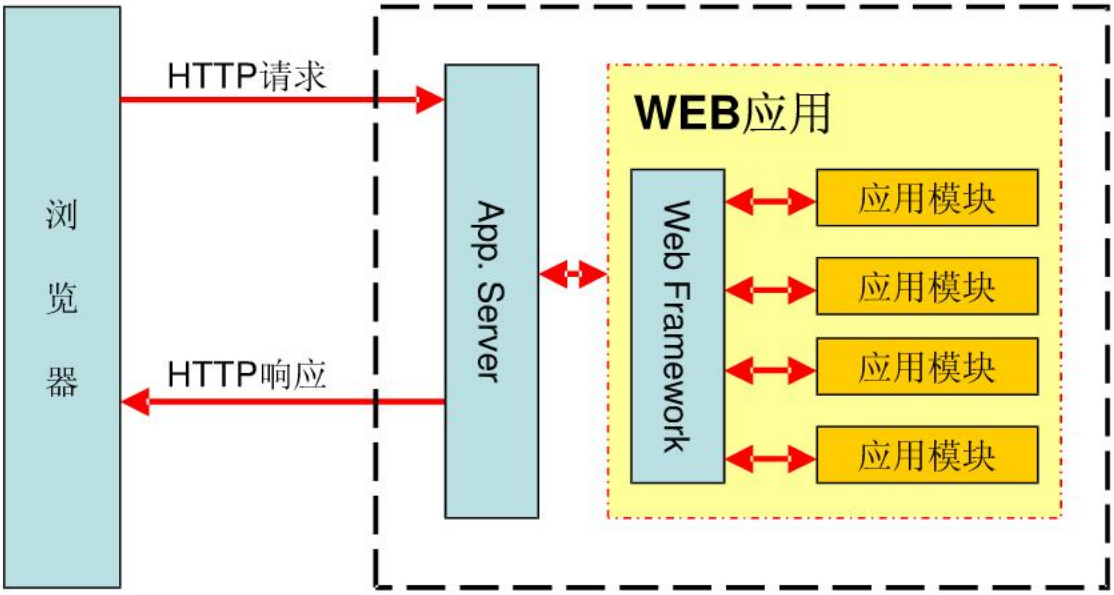
今天分享的目的

- 尝试透过纷繁的表象，分析各种Web应用框架的本质。
  - Spring MVC
  - Tapestry
  - Webwork
- 粗略了解各框架的优缺点。
- 考虑进一步发展我们的框架的方向。

研究院

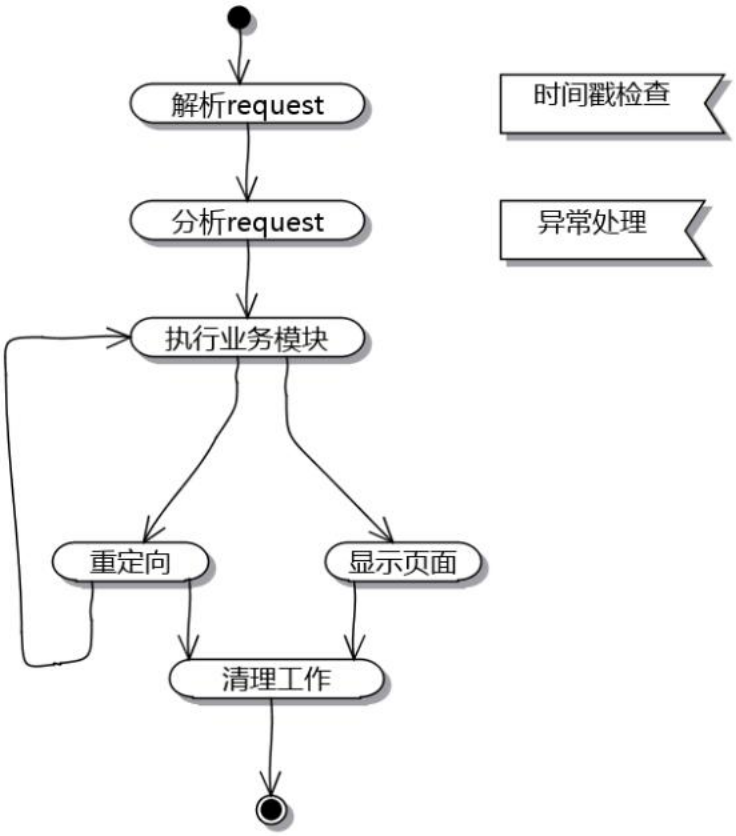

浏览器
HTTP请求
HTTP响应
App. Server
黑 箱

打开黑箱


浏览器
HTTP请求
HTTP响应
App. Server
WEB应用
Web Framework
应用模块
应用模块
应用模块
应用模块

研究院

一般WEB请求处理过程



解析request

- 解析request包括:
  - 解析URL
  - 解析HTTP headers
  - 解析query string
- 解析request工作主要是由**servlet container**完成的:
  - 根据URL，调用适当的webapp
  - 将URL匹配web.xml中的servlet-mapping，并调用适当的servlet
  - 根据HTTP method，调用servlet中适当的方法。
  - 当需要时，解析query string
- 应用程序/框架需要做的工作:
  - 告诉servlet container，当以何种locale/编码来解析query string
  - 解析multipart/form-data的数据（upload）

研究院

分析request

- 分析request是指：
  - 根据前一步解析而得的信息，进一步确定应该如何来处理这个request。
  - 最常见的方式，是根据URL中的某部分，或者某参数的值，来取得某个业务模块，并执行之。

研究院

- 这一步是应用程序员需要做的事，主要包括:
  - 分析/取得WEB参数
  - 验证表单
  - 根据请求的内容，执行具体的业务逻辑（例如从数据库中取得数据，或保存数据）
  - 决定要显示的页面，并准备好页面显示所必须的一些参数。
- 除了上述这些内容，还可能需要考虑一些非功能性的需求，例如:
  - 页面的安全特性（是不是必须登录？是否重复提交？等）
  - 页面的流程（页面之间的关系、状态的保存等）

研究院

显示页面

- 这一步涉及到页面的展现:
  - 以何种技术展现页面？（velocity、jsp、freemarker、hardcode，etc.）
  - 显示哪个页面？（找到正确的页面文件）
  - 页面的重用
  - 页面的布局

研究院

重定向

- 有时不一定直接显示页面，而是将控制转发给另一个模块/应用。
  - 内部重定向，对浏览器不可知，将控制转发给另一个模块
  - 外部重定向，通过浏览器定向，将控制转发给其它应用

研究院

清理工作

- 释放资源
- 记录日志
- 释放ThreadLocal

研究院

可选操作

- 异常处理
  - 日志
  - 显示错误页面
  - ……
- 时间戳检查
  - 通过时间戳检查，可以加快响应的时间，减少不必要的业务操作
  - 需要HTTP协议配合：If-modified-since等。
- Portlet支持

研究院

其它需要考虑的内容

- 以上只是列出了一个Web框架要处理的常见内容，事实上还有很多要考虑的事情，例如：
  - 可定制性
    - Search-engine-friendly URL
  - 开发的便利性
    - 页面技术的复杂性
    - 代码的侵入性
    - 和系统资源整合的方便性
    - 模块性
  - 扩展性
    - 创建新功能/替换现有功能的可能性
  - ……
  - ……

Web框架实质——扩展

- 现在有很多open source的Web框架，例如：
  - Spring MVC
  - Tapestry
  - Webwork
- 共同点：
  - 均实现了M-V-C设计模式。
  - 在整个请求处理的流程中，提供多种扩展点，来实现具体的业务逻辑。
  - 尽可能简化开发应用的复杂性。
- 那么不同的Web框架，它们之间有什么不同呢？
  - 提供扩展的方式不同
  - 扩展点的功能和数量不同

研究院

Web框架的实质——组合

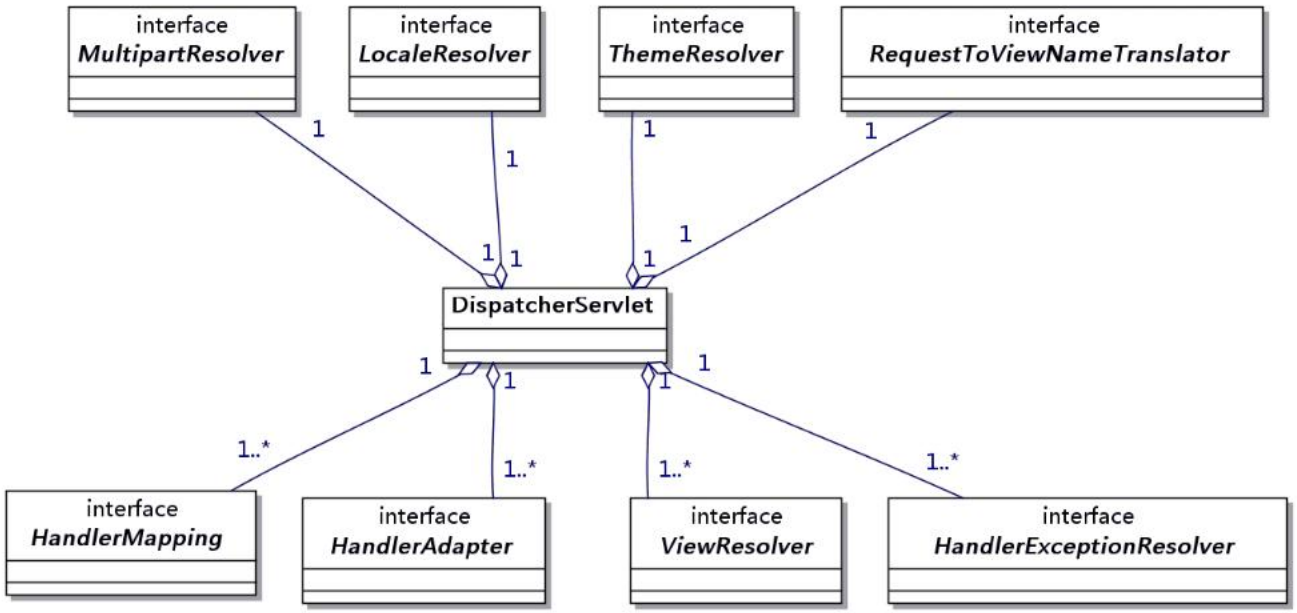
- 从另一个角度看
  - 其实一个框架就是一个模型
  - 模型由很多部分组合而成
    - 有哪些部分、如何组合？这些答案的不同就构成了各种不同框架的不同。
    - 但不同的实现，为的是完成类似的任务
- 我们将从分析每个框架的模型来入手

现代“组合”技术的基础

- 不约而同，今天我们要分析的三个Web框架，都为自己建立了一个与Web无关的“组合”对象的平台。
  - Spring MVC —— 以Spring framework为基础
  - Tapestry —— 以Hivemind为基础
  - Webwork —— 以Xwork为基础
- 这个基础平台的不同，很大程度地影响了Web框架本身的风格。
  - 平台的优点，直接带给Web框架独特的价值。
  - 平台的缺点，也带给相应Web框架有时是难以克服的缺陷。

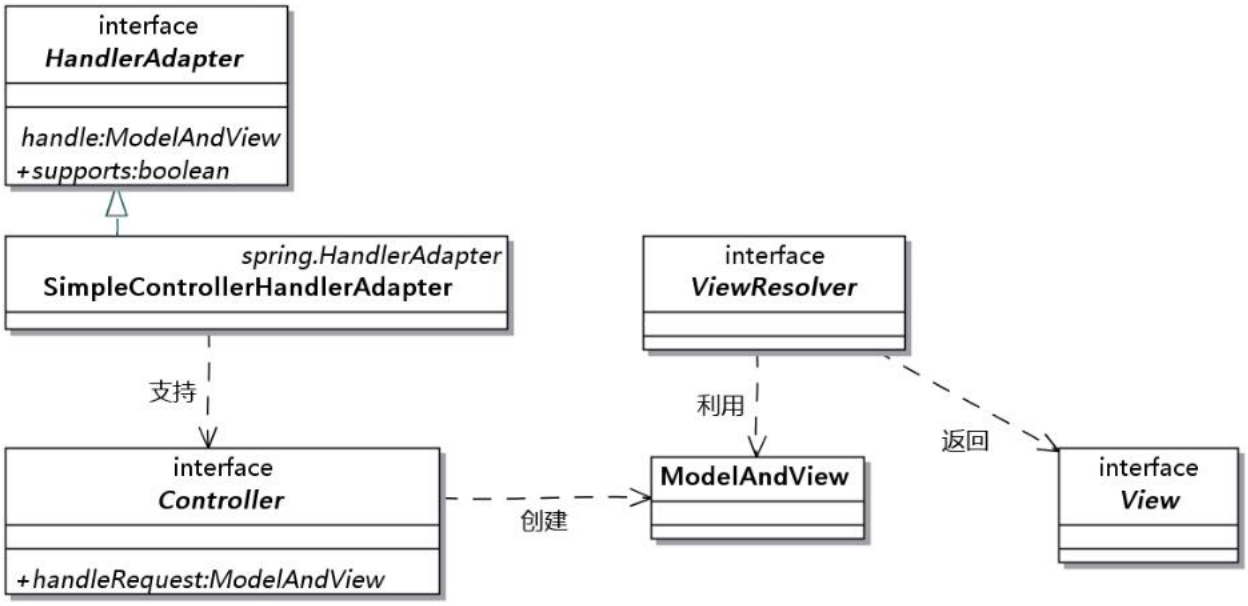
研究院

Spring MVC的模型



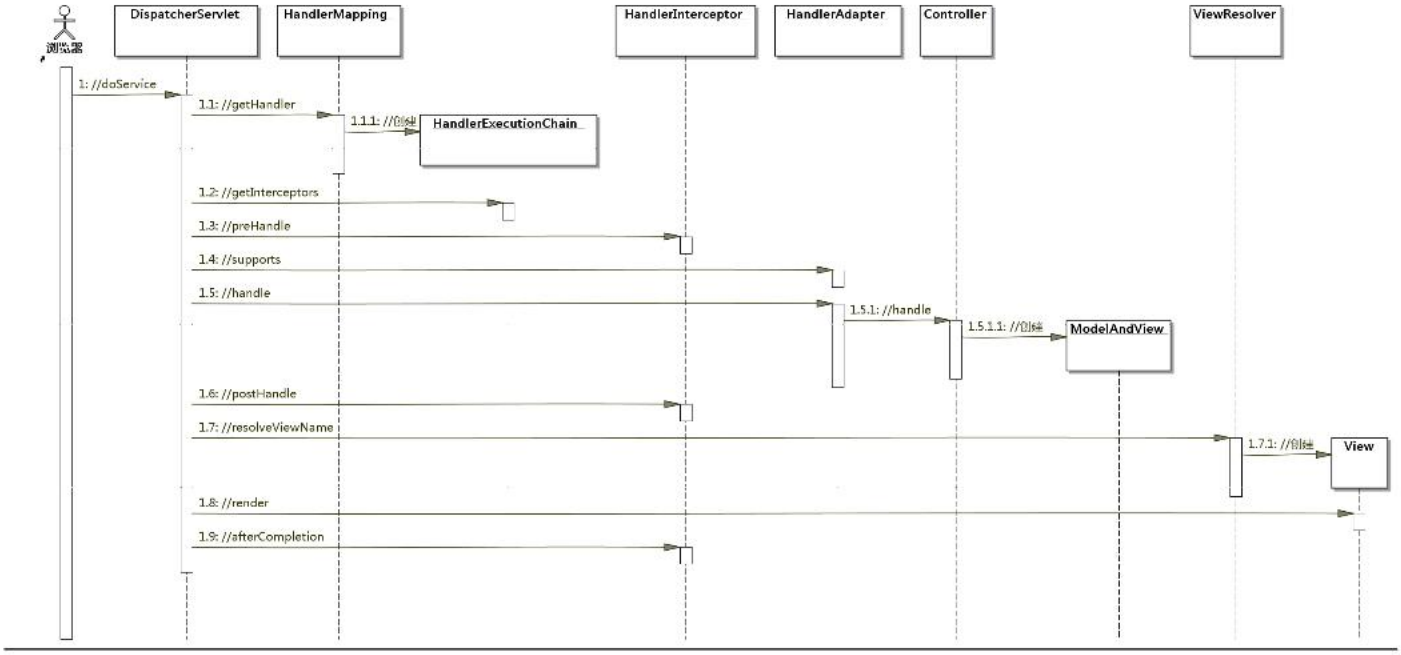
研究院

Spring MVC的模型



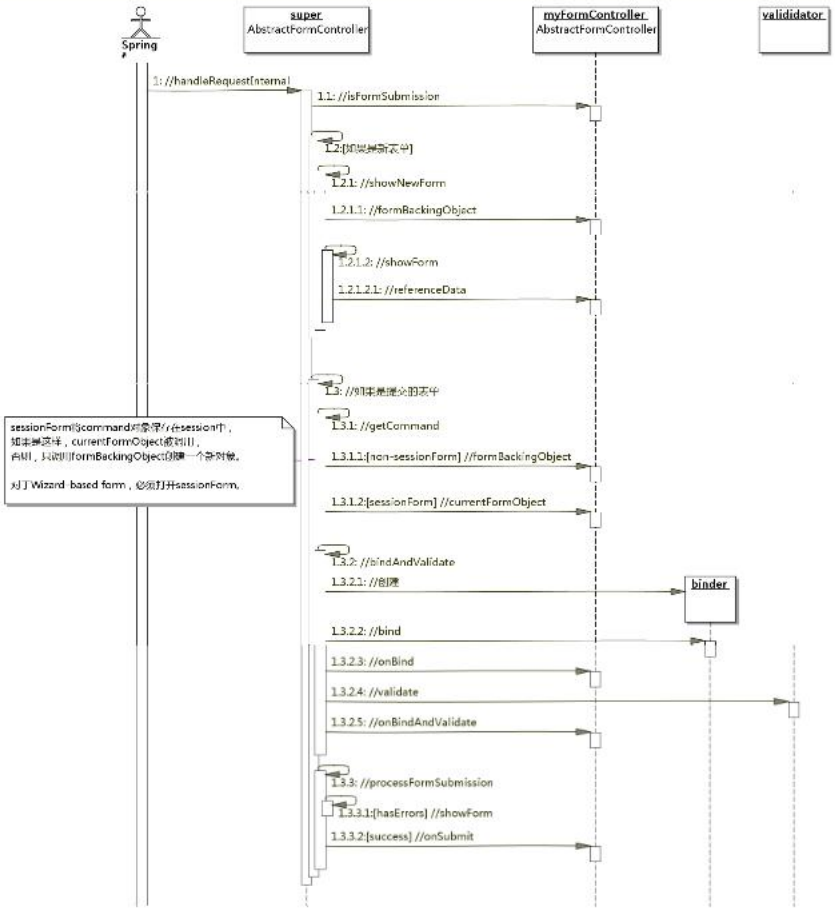
研究院

Spring MVC的request处理流程



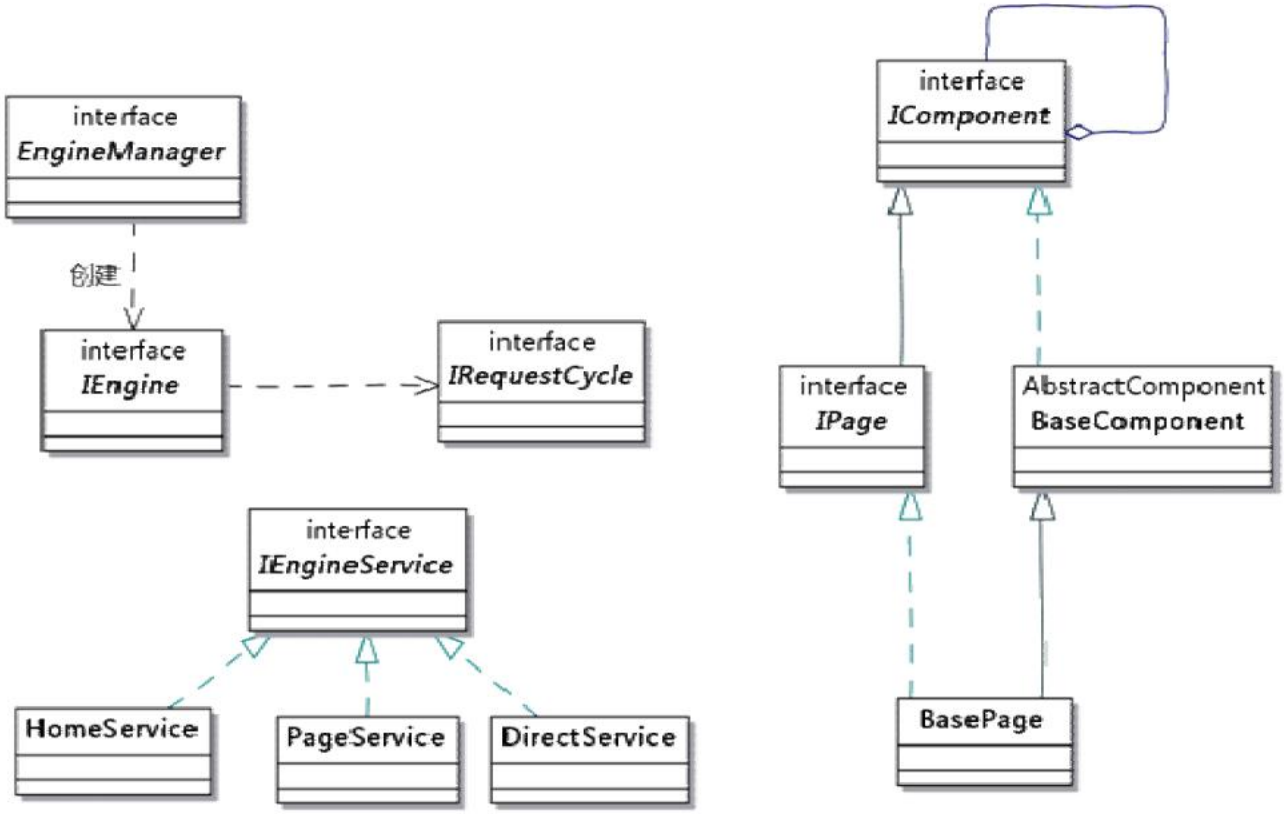
研究院

Spring MVC的表单处理过程

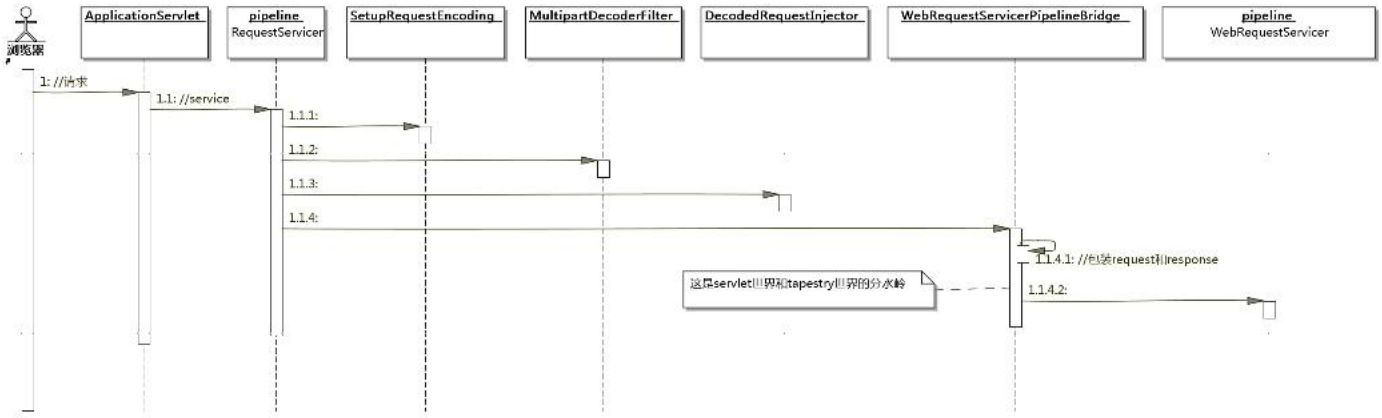


研究院

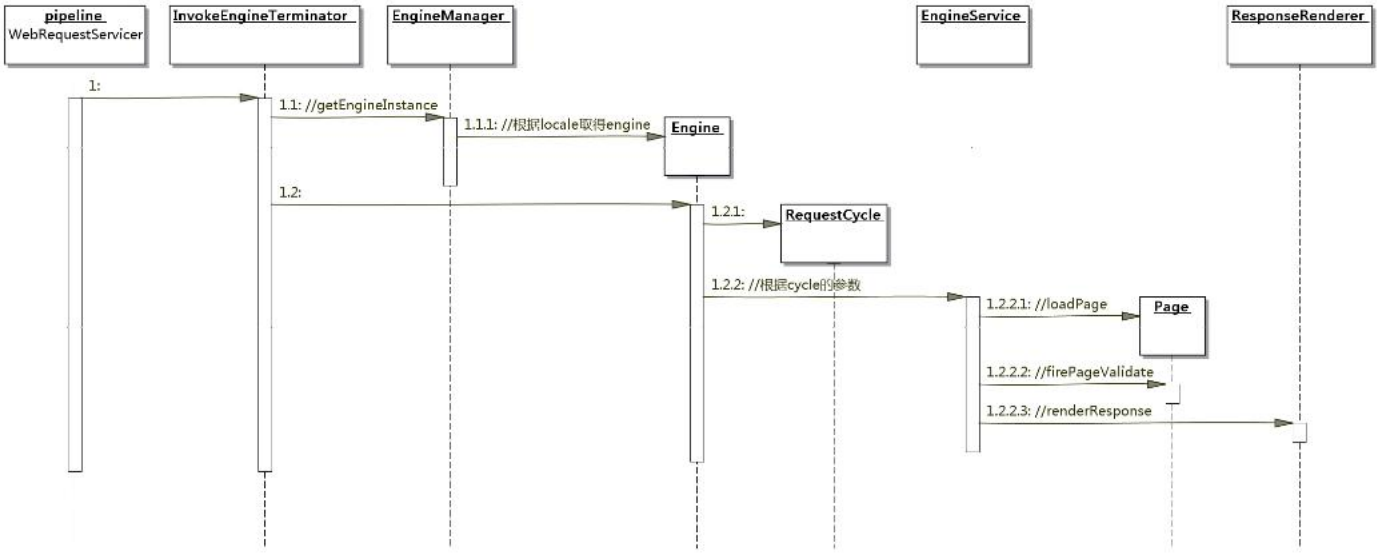
Tapestry的模型



Tapestry的request处理流程

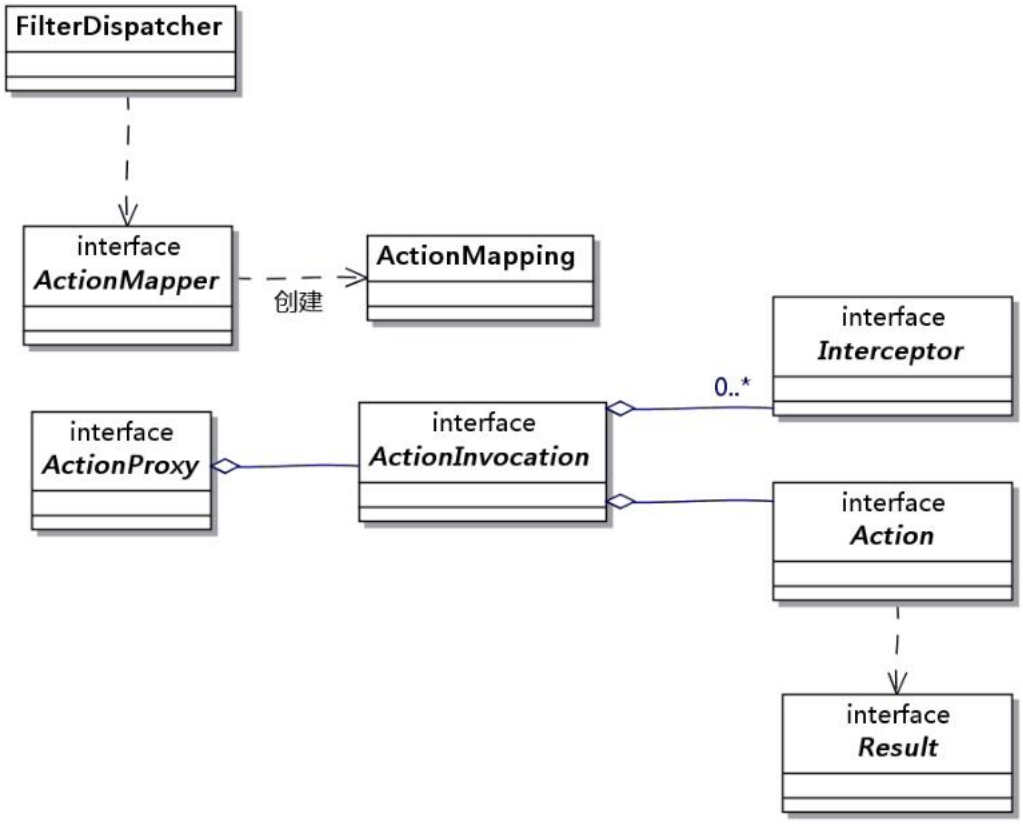


Tapestry的request处理流程(续)



研究院

Webwork的模型


FilterDispatcher
interface
ActionMapper
创建
ActionMapping
interface
Interceptor
0..*
interface
ActionProxy
interface
ActionInvocation
interface
Action
interface
Result

对比——解析request

- Spring MVC
  - 根本没考虑编码问题（输入中文会乱码），也没有提供明显的扩展点来做这件事。
  - 利用MultipartResolver来处理upload表单
- Tapestry
  - 通过requestServicer pipeline中的SetupRequestEncoding类来处理locale和编码，编码值从app spec.中注入。
  - 同样利用pipeline中的MultipartDecoderFilter来处理upload表单。

研究院

对比——分析request

- Spring MVC
  - Spring使用HandlerMapping来匹配URL和controller。
  - Spring 2.0支持ControllerClassNameHandlerMapping，就是利用URL命名规范来映射controller，使配置文件被简化。例如：WelcomeController -> /welcome/*。
  - 利用HandlerAdapter分离框架对具体Controller实现的依赖。最常用的是SimpleControllerHandlerAdapter。
- Tapestry
  - 通过ServiceEncoder来定制URL，也就是将URL翻译成参数表，然后利用这些参数来确定service、page等。
  - 类似Spring的HandlerAdapter，Tapestry也支持任意数量的EngineService。最常用来显示页面的service叫做PageService。

对比——执行业务模块

- Spring MVC
  - Spring的业务模块为controller
  - Spring提供了很多种controller，比较有用的有：MultiActionController、CommandController、FormController、WizardController。
  - 大部分controller支持从query data生成command对象。
  - 可以通过注入的方式来装配controller。
  - Spring 2.0支持session和request scope的对象注入。
  - Spring未提供表单验证的功能，需要通过硬编码来完成表单的验证。
  - Controller支持interceptors，可以用它来完成一些特别的功能，例如：页面安全性。
  - 由于Spring本身的功能，在controller里调用业务逻辑非常容易。

研究院

对比——执行业务模块

- Tapestry
  - Tapestry的业务模块为page和component。
  - 利用ognl的功能，可以在页面模板中直接将用户的输入值注入到page properties中。
  - 同样，通过注入（page spec.或annotation），page可以获得容器中的任意对象。
  - 没有直接的方法可以对page和component创建interceptor，如果要实现诸如页面安全的功能，必须通过基类，或通过pipeline来做。
  - Tapestry的表单验证是通过页面控件来做的。

研究院

对比——显示页面

- Spring MVC
  - Spring通过ViewResolver将view的名称和view的实现对应起来。
  - Spring理论上可以使用任何展现技术，但主要推荐的还是JSP。
  - Spring 2.0提供了一套form tag，使创建form表单的工作简化很多。
  - Spring没有直接提供页面重用的功能。
  - Spring没有直接提供页面布局的功能，但可以通过TilesView可以将tiles集成进来。
  - 当然，由于webwork的sitemesh基本上是和webwork分离的，所以应该也可以被spring所使用。

对比——显示页面

- Tapestry
  - Tapestry最强的技术就是它的基于控件的模板技术:
    - 和标准的HTML兼容，可以使用dreamweaver等HTML编辑器来编辑它的模板文件。
    - 可重用的控件（例如：DatePicker），支持JavaScript。
    - 页面的出错信息很详细。
    - 通过可重用的控件（例如：常见的border.jwc控件），可以方便地实现页面布局。
  - Tapestry不直接支持其它页面显示技术。

研究院

其它关注点：page driven

- 所谓page-driven，就是在开发应用时，以页面为主导，程序为辅助。这种模式可以比较快和直观地开发应用。
- Spring和webwork都不支持这种模式。Spring2.0虽然增加了CoC的功能（Convention over Configuration），但为了实现它仍然需要相当多的配置。
- Tapestry可以做到page-driven。在tapestry中，只要创建一个普通的html页面，就可以显示出来——即使page的程序还没写。

研究院

其它关注点：侵入性

- 所谓侵入性，就是指应用的代码依赖多少框架的代码。最理想的情况，是没有依赖，但这个很难做到。比较好的情况是只依赖一些特定的接口，而这些接口越简单越好，同时接口本身和框架之间也是松散耦合。
- 这一点Spring做得比较好:
  - 首先，Spring MVC的核心Controller接口非常简单，只有一个方法。
  - 其次，Controller接口和Spring MVC之间的耦合只是通过一个HandlerAdapter完成的，除此之外没有任何关联。
- Tapestry的侵入性最大，但是据说其后续版本将改良这一点，引入POJO编程。
- Webwork的侵入性也比较小，Action接口并不复杂。但Action接口是webwork的核心类，和webwork耦合很紧，所以其侵入性比Spring略大一些。
- 较低的侵入性意味着较好的扩展性、较易于测试、较好的系统结构。
- 过度追求低侵入性，也是有问题的。因为基于接口的编程可以利用编译器的检查，而假设侵入性为零，那意味着连接口也不能用了，这样就只好通过配置或者convention来定义规则，这样不一定比使用接口要好。

其它关注点：基于request vs. 基于对象

- Webwork、Spring都是基于request的Web框架。
  - 优点：简单易用；对request和response有直接的控制。
  - 缺点：和WEB结合太紧；难以实现页面/组件之间的关联、重用等高级功能。
- Tapestry是基于Object的Web框架。
  - 优点：易于实现页面/组件的重用；可创建出非常复杂的可重用组件：DatePicker、Tree等，也可非常方便地实现JavaScript组件；书写页面显得很结构化。
  - 缺点：对request和response没有直接的控制，以至于做一些简单的HTTP操作也显得很麻烦，例如：重定向；页面过于结构化，导致一些页面显得很笨拙。

其它关注点：Service model vs. Bean model

- Service model，就是像Tapestry所基于的Hivemind的模式。
- Bean model，就是像Webwork所基于的xwork，以及Spring的模式。
- Bean model可以看作是简化的Service model，但service model包含更多的内容：
  - Service model最重要的一点是：Service的提供者和使用者，两者权责的分离。
  - 另一个重要点是：Service包含一个service的定义，而bean只是一堆相对无意义的properties而已。
- 两者均支持IoC、AoP。
- 但Service model支持搭建更复杂的对象层次，而bean model只能在一个“平面”的层次中工作。这实际上是Spring MVC和Tapestry最本质的区别。
- 然而HiveMind的思路虽然绝妙，实现上却还没达到完美，以至于有的情况下会很麻烦。

我们框架的发展方向

- 每一个框架都有它的独特优点，也有它的不足。
- 作为一个希望持续发展的公司，一定要有一套完整的开发平台（包括框架、工具、开发流程等）。
- 我希望能够开发出一套集各家之所长，又融入我们现有优点的框架，以及相应的开发工具：
  - Citrus （Service框架）
  - Citrus Web （下一代的Webx框架）
  - Citrus Studio （一组eclipse插件，方便开发）
  - Citrus Xyz （不止是WEB应用，任何其它应用都可基于这一平台）