Design

Calvin Xiao edited this page on 4 Feb 2014 · 34 revisions

1. Web

MVC Framework： Spring MVC 3.0 Restful的风格终于回归了MVC框架的简单本质，对比之下Struts2概念太复杂更新又太懒了。

Template：JSP 2.0且尽量使用JSP EL而不是taglib，万一要写taglib也用纯JSP来编写，一向是SpringSide的推荐，Freemarker们始终有点小众, 而Thymeleaf与美工配合度非常高，可惜也是太少用户了。

Layout Decoration： Tiles的配置都太复杂了，SiteMesh2好些，但Sitemesh3烂尾了。

Javascript Library： 随大流用了JQuery。其实Dojo的面向对象语法更优美，但用户数和插件社区差了点。

CSS Framework： 最热火的Twitter Bootstrap，提供了简便的布局能力和基本的页面美化。

JavaScript/CSS Compressor: 还是随便选的YUI Compressor。

Validation: JQuery Validation Plugin这种客户端校验的客户体验更好，而Spring MVC集成Hibernate Valiator的服务端校验则可以避免恶意用户跳过页面直接发送请求，校验规则也更多，需要混合使用。

2. WebService

SOAP WebService： JAX-WS2.0的注解 + Apache CXF 无疑是最成熟的，一说起Axis1/2我都要打冷颤。

Restful Service： JAX-RS 1.0 + Jersey/CXF，够标准。但直接使用Spring MVC能使架构更简单。 如果追求极致的性能指标，直接写Servlet也没啥。

Restful Client: 刚出来的JAX-RS 2.0标准，实际是用Jersey的client api做蓝本的， 而直接使用Spring的RestTemplate可以减少第三方包的引入。

为了隔绝变化影响，隐藏细节，对外暴露的DTO和应用内部的领域对象是不同的类型，用Dozer进行复制。

请求参数的校验，JSR303 Bean Validator的实现Hibernate Validator没太多的竞争对手。

3. Database

数据库设计基本原则: 见DataBase的相关章节

ORM Framework： 快速开发的应用里，领域对象肯定是用JPA标注的。至于API用Hibernate还是JPA，

因为那个极简便的，DAO只要写接口就好了的Spring-Data-JPA，所以选了JPA。 当然，JPA的实现还是用Hibernate。

追求高性能的应用，如各种Web服务，当然就是MyBatis了。如果项目再简单点，Spring JDBC其实也不错。

传统数据库: 无非Oracle与MySQL的选择，如果你恨MySQL依然是Oracle家的东西，可以考虑越来越多人用的，语法和Oracle很像的Postgresql。

NOSQL数据库: 国内用的比较多的还是Redis和MongoDB。Redis更像一个数据结构服务器，暴露各种数据结构的专有API。而MongoDB将数据存成BSON格式，也提供类似SQL的查询语句，更像一个schema-less的数据库。

数据库连接池: Apache DBCP本来一统江湖，现在被人批评又慢又复杂，所以有了Tomcat JDBC，另外温少的Druid也是一个选择。

Cache： 在JVM里的缓存，最老牌最多人用的依然是Ehcache，一些更强大的DataGrid方案如HazelCast，JBoss的Infinispan反而没什么人用。另外最简单的JVM内缓存是Guava的Cache。

而中央式的缓存，Memcached已经成为了事实标准。而且当主创撒手不管后，社区现在反而有着稳定的更新。 Client方面，比较稳健选择的还是Spymemcached。

3. Services

安全系统

Security Framework： 选择Apache Shiro是因为SpringSecurity的代码复杂度已经超过了它的实际需要，扩展困难痛苦。另一个原因是SpringSecurity的基本API居然只支持基于角色的判断，e.g. hasRole("Administrator")，而Shiro同时还支持我们其实更常用的基于Permission的判断，e.g. hasPermission("User:Edit")。

Java消息系统

JMS： ActiveMQ是最多人选用的应用服务器无关的JMS实现，JBoss的HornetQ同样只是JBoss的用户在使用。Spring自带的JMS封装很好用。但还有更高级的如支持跨平台的AMQP协议的RabbmitMQ。

任务调度

Schedule： 对于固定时间间隔的任务，JDK自带的Executor已足够好。Cron式定时执行，Spring的Scheduler也能满足。而且Spring提供的纯XML配置也让Scheduler变得很简单，Quartz更大的优势体现在保证集群中有且仅有一台服务器执行任务。另外，SpringSide还演示了基于Redis做了一个适合海量的只需单次触发的任务。

Java监控系统

JMX: Jolokia能将JMX中的MBean以Restful+JSON的方式暴露出来，使JMX这个古老的，在平台互通中显得有点封闭的协议重新焕发了青春。而Spring-Jmx将普通POJO注释一下就变成MBean也非常方便。

产品功能

其他Production Feature: 用Hystrix对访问资源进行并发、延时、短路控制，防止系统雪崩。而监控方便包括自己写的Metrics Reporter和Graphite。

4. Utilizes

常用

General： Apache Commons Lang说是伴着我们长大的也不为过，3.0版连package名也改了，全面支持泛型。 Guava 是Google新鲜推出的优雅产品。但说它会一桶天下又不定，因为它有时候太新潮了，反而用不惯。比如StringUtils我还是喜欢用Apache的，IO也同样是Apache Commons IO的好使。

XML： 用JDK自带的JAXB就算了，不折腾。

JSON： GSon虽然系出名门而且接口优雅，但Jackson的功能更加丰富到匪夷所思，而且比GSon快很多。

Email： Spring自带的Email封装挺好用的。

Logging: Slf4j作为入口，早就替代了Apache Common Logging了，下面的实现Log4j 1.x 被批判太多同步方法太慢，Log4j作者的后作Logback就好很多了，但社区似乎不甘心log在一家QOS公司手里，又在推动log4j2.0的发展，目前还是beta版。另外选择Logstash做日志的中央式处理。

最后，Freemarker虽然不用来做页面Template，平时用来生成点东西也不错的。 JodaTime这种要直接加入JDK的就不多说它了。HttpClient建议用Apache HttpClient好过JDK自带。

5. Test

单元测试

Unit Test： JUnit始终是正统，TestNG的功能如测试用例分组它也慢慢支持了。AssertJ 是目前最好的Assert语句库。

模拟框架

Mock： Mockito的API比老牌的EasyMock更为优雅，而PowerMock则能配合Mockito完成static方法，函数内部new 出来的对象这些Mockito做不了的mock。

功能测试

Functional Test：Selenium与WebDriver的合并后，最大改进是原来基于javascript的方案， 变成了直接调用浏览器的核心API，性能好了。

性能和扩展性测试

Performance/Stability Test: Jmeter作为测试工具里最成熟的，Gatling还需要时间成熟。

6. Development Environment

JDK6这样没什么兼容性问题又成熟得一塌糊涂的版本建议大家都升级吧。JDK7也不错，有G1垃圾收集器和Try-Catch新语法的语法糖。

用Jetty7是因为它的嵌入式版本做得好，集成测试不用部署直接就开跑了。开发时一般也不用Eclipse插件，直接自己在代码里启动了，省下打包拷贝War文件的时间。Tomcat现在也有嵌入式版本了，而Jetty最新版要JDK7。

用H2 Database，既是嵌入式的，又可以持久化到文件。用Web Console查看，性能还是嵌入式中最好的。

用Maven，在项目构建脚本不复杂的时候的首选，否则就只能ant+ivy了，或者像hibernate和spring一样，用gradle.

另外，用Log4jdbc在开发时查看实际执行的SQL。

最后，用Jenkins做持续集成, Sonar做代码质量检查，是大部分好项目的共同爱好。

7. 參考資料

- RoadMap