EMSC 架构剖析

-- 知其然，知其所以然

研发中心 柳伟卫

会议规则

- 原则上不带手机进场
- 手机震动或静音
- 自由发问。沉默代表认同

目标

- 了解公司项目架构的变迁
- 知道 EMSC 的架构原理
- 现有架构的不足、改进

Web（互联网）起源

解决的问题

- 分布式计算
- 信息的发布与共享

- 万维网（那就是著名的 World Wide Web 简称： WWW ）的基础技术，第一项是用于编写通用可读文档的超文本标记语言 （ HTML); 第二项是通过因特网引用其他可访问文档或资源的统一资源定位期 (URL); 第三 项用于发布资源的超文本传输协议（ HTTP) 。
- URL ：统一资源定位符是对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的 URL ，它包含的信息指出文件的位置。
- HTTP: 是一个客户端和服务器端请求和应答的标准（ TCP ）。客户端是终端用户，服务器端是网站。通过使用 Web 浏览器、网络爬虫或者其它的工具，客户端发起一个到服务器上指定端口（默认端口为 80 ）的 HTTP 请求，获取应答的服务器上存储着（一些）资源，比如 HTML 文件和图像。
- 尽管 TCP/IP 协议是互联网上最流行的应用， HTTP 协议并没有规定必须使用它和（基于）它支持的层。 事实上， HTTP 可以在任何其他互联网协议上，或者在其他网络上实现。 HTTP 只假定（其下层协议提供）可靠的传输，任何能够提供这种保证的协议都可以被其使用。

Web Service （ Web 服务）

数据交换的标准

- XML
- JSON

XML 和 JSON 的优缺点对比

- (1).可读性方面。
- JSON和XML的数据可读性基本相同，JSON和XML的可读性可谓不相上下，一边是建议的语法，一边是规范的标签形式，XML可读性较好些。
- (2).可扩展性方面。
- XML天生有很好的扩展性，JSON当然也有，没有什么是XML能扩展，JSON不能的。
- (3).编码难度方面。
- XML有丰富的编码工具，比如Dom4j、JDom等，JSON也有json.org提供的工具，但是JSON的编码明显比XML容易许多，即使不借助工具也能写出JSON的代码，可是要写好XML就不太容易了。
- (4).解码难度方面。
- XML的解析得考虑子节点父节点，让人头昏眼花，而JSON的解析难度几乎为0。这一点XML输的真是没话说。
- (5).流行度方面。
- XML已经被业界广泛的使用，而JSON才刚刚开始，但是在Ajax这个特定的领域，未来的发展一定是XML让位于JSON。
- (6).解析手段方面。
- JSON和XML同样拥有丰富的解析手段。
- (7).数据体积方面。
- JSON相对于XML来讲，数据的体积小，传递的速度更快些。
- (8).数据交互方面。
- JSON与JavaScript的交互更加方便，更容易解析处理，更好的数据交互。
- (9).数据描述方面。
- JSON对数据的描述性比XML较差。
- (10).传输速度方面。
- JSON的速度要远远快于XML。

架构的设计原则

- 避免重复造轮子
- 按需代码
- 科学发展观

避免重复造轮子

- 开源 vs 闭源
- 能力与进度

按需代码

设计应该是根据被设计系统的功能、行为等方面的需求而作出的

- 一位建筑师在面对设计一个城市住宅区的目标时，头脑里所抱有的想法是要使用所有现代屠宰场的组成部分来完成这个设计！这也许是他所构思过的最棒的屠宰场，但是对于预期的居民来说却谈不上舒适，因为他们将不得不战战兢兢地穿行在安装着旋转式屠宰刀的走廊中。摘自《建筑师讽刺剧》（ The Architects Sketch ）（ http://en.wikipedia.org/wiki/Architects_Sketch ）

- 很多软件项目一开始就采用最新最时髦的架构设计，到了后来却发现满足系统的需求实际上并不需要这样一种架构。 design-by-buzzword （按照时髦的词汇来做设计）是一种常见的现象。出现很多此类行为是由于设计者并不理解为何（ why ）架构的约束。

案例

- 数据模拟器（ ECS ）： Java JDBC
- 数据采集服务器 (DAS): Java+Spring+c3p0

公司项目架构的发展

0G ： Servlet+JSP+JavaBean

- Servlet 是在服务器端执行的 Java 程序，一个被称为 **Servlet 容器**的程序（其实就是服务器） 负责执行 Java 程序。而 JSP(Java Server Page) 则是一个页面， 由 **JSP 容器**负责执行。
- Servlet 和 JSP 两者最大的区别就是， Servlet 以 Java 程序为主， 输出 HTML 代码时需要使用 out.println 函数，也就是说 **Java 中内嵌 HTML** ； 而 JSP 则以 HTML 页面为主，需要写 Java 代码时则在页面中直接插入 Java 代码， 即 **HTML 中内嵌 Java**
- JSP 便于输出，而 Servlet 便于进行逻辑处理。因此实际应用中两者常常结合使用，各司其职。

JSP文件

代码如下:

```
<html>
  <body>
   <h1>
   <% out.println("JSP"); %>
   </h1>
  </body>
 </html>
```

Servlet

代码如下:

```
public class MyServlet ... {
  ...
  out.println("<html>");
  out.println("<body>");
  out.println("<h1>");

  out.println("Servlet);

  out.println("</h1>");
  out.println("</body>");
  out.println("</html>");
 }
```

问题

- 分层混乱。 Servlet 用来写 business layer 是很强大的，但是对于写 presentation layer 就很不方便。 JSP 则主要是为了方便写 presentation layer 而设计的。当然也可以写 business layer 。所以经常会不自觉的把 presentation layer 和 business layer 混在一起。甚至，把数据库处理信息放到 JSP 中
- 对于前端开发技术要求高（服务端语言）
- HTML 与 JSP 标签混杂，代码可读性差
- 对服务器性能要求高（需在服务端编译执行，常驻与内存）
- 用户体验差

1G ：
Flex+AS3+Cairngorm+ BlazeDS+Spring+Hibernate

Flex

- RIA 富客户端技术，异步调用，界面无刷新，高用户体验
- 跨平台：跨浏览器、桌面和操作系统（包括移动端）
- 基于 XML 的 MXML 语言 和 遵循 ECMAscript 的 Actionscript 3 语言
- 界面布局、样式和逻辑处理分层清晰
- MVC
- 控件丰富
- 对服务端语言无依赖

BlazeDS

- BlazeDS 是一个基于服务器的 Java 远程调用（ remoting ）和 Web 消息传递（ messaging ）技术，使得后台的 Java 应用程序和运行在浏览器上的 Flex 应用程序能够相互通信。当客户端 RPC 控件调用远程服务时，该控件就会把服务端返回的数据保存在一个 ActionScript 对象中，这样，在程序中就能够很轻松的获取想要的数据，而这些客户端控件包括 HTTPService 、 WebService 、 RemoteObject 控件。
- AMF （ Action Message Format ）是 ActionScript对象序列化后的二进制流。用于 Adobe Flash 应用和远端服务的通讯。由于它是基于二进制的数据传输，所以相对于 XML SOAP ， json ， WebService 等基于字符串的数据格式，有数据体积小和效率高的特点。
- 提供了真正的数据流实时模式。

案例

- 东莞企业能管中心 (DGEMS)
- 政府级能管中心 (GEMS)
- 数字陶瓷 (EEMS)
- EagleOS
- EagleDesinger

问题

- 编译漫长。。。
- Flex 4 实现上未完成，仍包含了部分 Flex3 的组件
- Flex 4 设计上过于先进，现有硬件不足以支撑
- 受到了乔布斯的阻击，节节败退
- 私有协议 ，技术转换困难
- 人才匮乏
- 用人成本高

2G:
JSP/FreeMarker + Structs+Spring+MyBatis

FreeMarker

- 模板引擎
- 不能编写 java 代码，可以实现严格的 mvc 分离
- 对 jsp 标签支持良好
- 内置大量常用功能，使用非常方便
- 宏定义（类似 jsp 标签）非常方便
- 使用表达式语言

Structs

- MVC
- 分层清晰

MyBatis

- 使用简单的 XML 或注解用于配置和原始映射
- sql 写在 xml 里，便于统一管理和优化
- 解除 sql 与程序代码的耦合。
- 提供映射标签，支持对象与数据库的 orm 字段关系映射
- 提供对象关系映射标签，支持对象关系组建维护
- 提供 xml 标签，支持编写动态 sql 。

案例

- I-CMA 监控平台
- 电机能效提高系统（ EMEP ）

问题

- JSP 的问题仍存在，用户体验差
- FreeMarker 与 HTML 混杂，代码可读性差
- sql 工作量很大，尤其是字段多、关联表多时，更是如此。
- sql 依赖于数据库，导致数据库移植性差。
- 编写动态 sql 时，不方便调试，尤其逻辑复杂时。
- XML 配置繁琐、复杂，易错
- 权限管理不灵活

3G:
HTML5+JS+Ajax+EasyUI+
Structs+Spring+MyBatis

与 2G 比较

- 在 2G 基础上，重构了前端
- 模块化
- 模块无需载入主应用的 header ，大大节省带宽
- 简化了 Spring 的 bean 管理
- 富客户端技术应用，提升了用户体验
- 加入压缩与混淆

案例

- 现场信息采集子系统（ SIC ）
- 油品信息管理系统（ OEM ）

问题

- 针对不同的客户端 (浏览器、安卓) ，需要提供不同的 api
- 服务端 xml 配置仍然繁琐

3.5G ：
HTML5+ AngularJS+Bootstrap+
Jersey+Spring+Hibernate

- RESTful api
- 富客户端
- 响应式布局
- 前端 MVC 分层清晰
- 支持数据绑定机制
- 避免 DOM 的操作， js 代码优美

扩展阅读

- http://www.waylau.com/dont-write-angularjs-app-using-jquery-thinking/

数据绑定

数据绑定作为 AngularJS 的特性之一，更新视图，无需操纵 DOM，剩下很多代码（想想 Flex）。在jQuery更新视图的步骤是这样的，将设有如下视图

```
<ul class="messages" id="log">
</ul>
```

我要在 `ul` 插入 `li`,需要操作 DOM 来添加节点元素。

```
$.ajax({
  url: '/myEndpoint.json',
  success: function ( data, status ) {
    $('ul#log').append('<li>Data Received!</li>');
  }
});
```

假设，此时，我要删除 `log` 视图，那我不得不去 操作 DOM,把ajax 里面的添加方法修改了。太蛋疼了。其实在 js 里面写 html 本身就是一件困难的事，因为 html 包含尖括号、属性、双引号、单引号、方法，在 js 需要对这些特殊符号进行转义，代码将会变得冗长易出错且难以识别。 如下面的例子：

```
var str = "<a href=# name=link5 class="menu1 id=link1" + "onmouseover
=MM_showMenu(window.mm_menu_0604091621_0,-10,20,null,\'link5\');"+ "s
el1.style.display=\'none\';sel2.style.display=\'none\';sel3.style.dis
play='none\';"+" onmouseout=MM_startTimeout();>Free Services</a> ";

document.write(str);
```

看下 AngularJS 是怎么干的，先定义一个界面模板

```
<ul class="messages">
    <li ng-repeat="entry in log"></li>
</ul>
```

而后更新数据

```
$http( '/myEndpoint.json' ).then( function ( response ) {
    $scope.log.push( { msg: 'Data Received!' } );
});
```

视图就会根据数据自动刷新了。而界面改成什么样，并不需要去修改更新数据的接口，比如改成下面的样子

```
<div class="messages">
    <div class="alert" ng-repeat="entry in log">

    </div>
</div>
```

案例

- 油品信息管理系统（ OEM_0.1 ）：半成品

问题

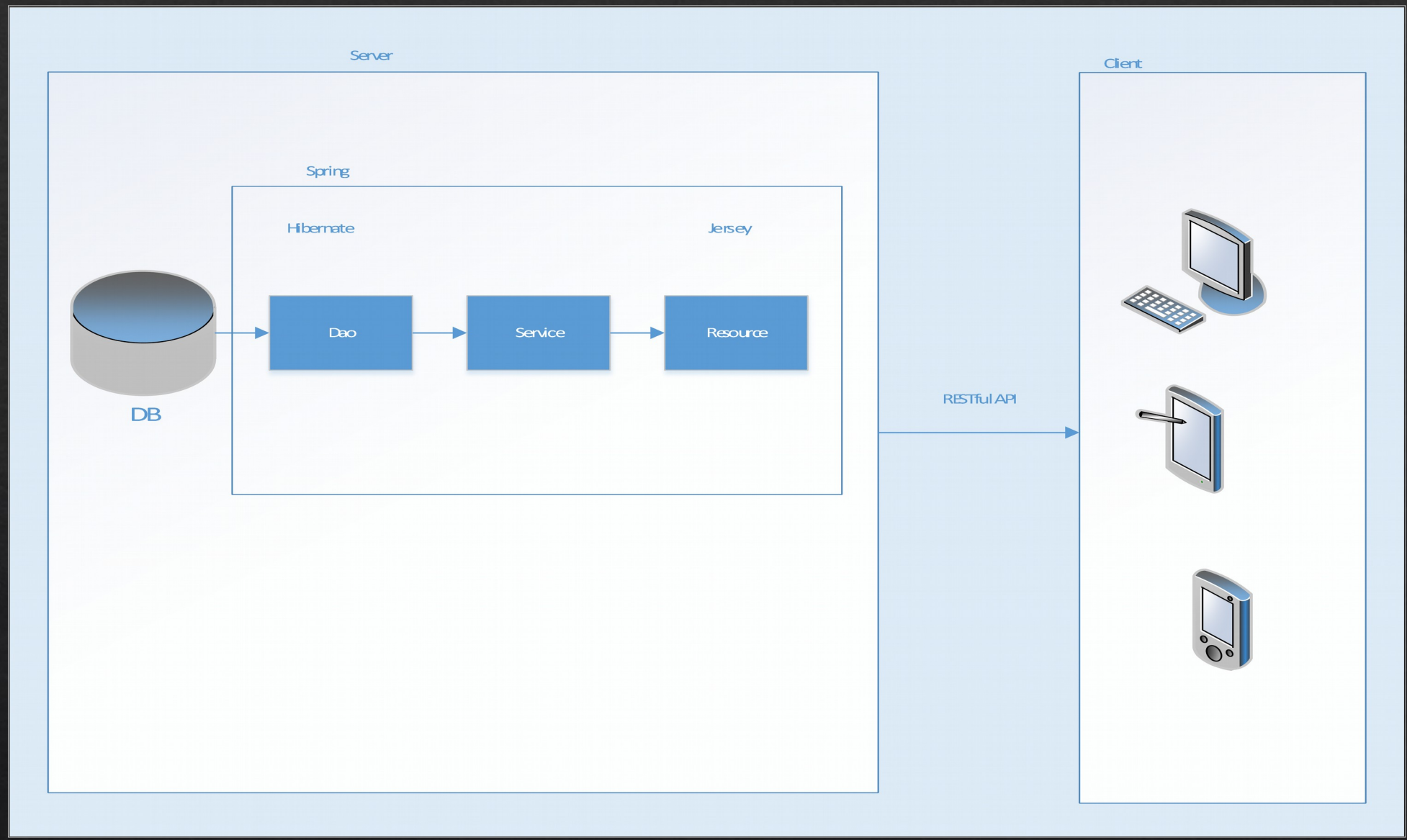
- 未完待续
- 对于前端编码要求高，特别是 js
- AngularJS 上手难
- AngularJS 后期版本变化大

4G:
HTML5+ Ajax+JS+Bootstrap+EasyUI +Shiro+Jersey+Spring+Hibernate

与 3G 比较

- RESTful api
- 响应式布局
- 简化了 Spring 的 bean 管理
- 简化了 ORM 的配置
- 引入了 Maven 管理理念
- 强化了权限模型（基于资源的授权模式）
- 完全采用开源技术

B/S or C/S


Server
Spring
Hibernate
Jersey
Dao
Service
Resource
DB
RESTful API
Client

案例

- EMSC

问题

- 权限管理还有待改进
- 分析平台缺少有效的数据绑定机制
- 大量的 DOM 的操作，使 JS 的可读性差
- 分析平台未能支持表格的友好支持

Why Java

- 面向对象：易学，符合人类思维
- 跨平台： **Write once, run anywhere 。节省编码、适配工作。**
- 安全稳定： Java 的强类型机制、异常处理、垃圾的自动收集等是 Java 程序健壮性的重要保证。对指针的丢弃是 Java 的明智选择。 Java 的安全检查机制使得 Java 更具健壮性。 Java 对通过网络下载的类具有一个安全防范机制（类 ClassLoader ），如分配不同的名字空间以防替代本地的同名类、字节代码检查，并提供安全管理机制（类 SecurityManager ）
- 人才济济：项目开发成本低，维护风险小

Sep 2014	Sep 2013	Change	Programming Language	Ratings	Change
1	1		C	16.721%	-0.25%
2	2		Java	14.140%	-2.01%
3	4	^	Objective-C	9.935%	+1.37%
4	3	v	C++	4.674%	-3.99%
5	6	^	C#	4.352%	-1.21%
6	7	^	Basic	3.547%	-1.29%
7	5	v	PHP	3.121%	-3.31%
8	8		Python	2.782%	-0.39%
9	9		JavaScript	2.448%	+0.43%
10	10		Transact-SQL	1.675%	-0.32%

Spring Framework

- IOC ： *Inversion of Control,* 低耦合。
- AOP : Aspect Oriented Programming ，作为面向对象编程的一种补充，广泛应用于处理一些具有横切性质的系统级服务，如事务管理、安全检查、缓存、日志、对象池管理等。

扩展阅读

- http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/
- https://www.gitbook.com/book/waylau/spring-framework-4-reference/details

Hibernate ORM

- 对象关系映射框架，它对 JDBC 进行了轻量级的对象封装，使得 Java 程序员可以使用对象编程思维来操纵数据库。
- 不仅负责从 Java 类到数据库表的映射（还包括从 Java 数据类型到 SQL 数据类型的映射），还提供了面向对象的数据查询检索机制，从而极大地缩短的手动处理 SQL 和 JDBC 上的开发时间。

RESTful Web Services

- 系统中的每一个对象或是资源都可以通过一个唯一的 URI 来进行寻址， URI 的结构应该简单、可预测且易于理解，比如定义目录结构式的 URI 。
- 以遵循 RFC-2616 所定义的协议的方式显式地使用 HTTP 方法，建立创建、检索、更新和删除（ CRUD ：Create, Retrieve, Update and Delete ）操作与 HTTP 方法之间的一对一映射：
  - 若要在服务器上创建资源，应该使用 POST 方法；
  - 若要检索某个资源，应该使用 GET 方法；
  - 若要更改资源状态或对其进行更新，应该使用 PUT 方法；
  - 若要删除某个资源，应该使用 DELETE 方法。
- 资源多重表述 :URI 所访问的每个资源都可以使用不同的形式加以表示（比如 XML 或者 JSON ），具体的表现形式取决于访问资源的客户端，客户端与服务提供者使用一种内容协商的机制（请求头与 MIME 类型）来选择合适的数据格式，最小化彼此之间的数据耦合。
- 无状态 : 对服务器端的请求应该是无状态的，完整、独立的请求不要求服务器在处理请求时检索任何类型的应用程序上下文或状态。无状态约束使服务器的变化对客户端是不可见的，因为在两次连续的请求中，客户端并不依赖于同一台服务器。一个客户端从某台服务器上收到一份包含链接的文档，当它要做一些处理时，这台服务器宕掉了，可能是硬盘坏掉而被拿去修理，可能是软件需要升级重启——如果这个客户端访问了从这台服务器接收的链接，它不会察觉到后台的服务器已经改变了。

HTTP 请求方法在 RESTful Web 服务中的典型应用

资源	GET	PUT	POST	DELETE
一组资源的URI，比如 `http://www.waylau.com/resources/`	**列出** URI，以及该资源组中每个资源的详细信息（后者可选）。	使用给定的一组资源**替换**当前整组资源。	在本组资源中**创建/追加**一个新的资源。该操作往往返回新资源的URL。	**删除** 整组资源。
单个资源的URI，比如 `http://www.waylau.com/resources/142`	**获取** 指定的资源的详细信息，格式可以自选一个合适的网络媒体类型（比如：XML、JSON等）	**替换/创建** 指定的资源。并将其追加到相应的资源组中。	把指定的资源当做一个资源组，并在其下**创建/追加**一个新的元素，使其隶属于当前资源。	**删除** 指定的元素。

Java REST

- 针对 REST 在 Java 中的规范，主要是 JAX-RS（ Java API for RESTful Web Services ）。Java EE 6 引入了对 JSR-311 的支持。 JSR-311（ JAX-RS ： Java API for RESTful Web Services ）旨在定义一个统一的规范，使得 Java 程序员可以使用一套固定的接口来开发 REST 应用，避免了依赖于第三方框架。同时， JAX-RS 使用 POJO 编程模型和基于标注的配置，并集成了 JAXB ，从而可以有效缩短 REST 应用的开发周期。
- JAX-RS 定义的 API 位于 javax.ws.rs 包中。
- 伴随着 JSR 311 规范的发布， Sun 同步发布该规范的参考实现 Jersey。 JAX-RS 的具体实现第三方还包括 Apache 的 CXF以及 JBoss 的 RESTEasy 等。未实现该规范的其他 REST 框架还包括 SpringMVC 等

Why Jersey

- 在 Java 中，既然 规范的制定者和实现者都是 Sun 公司（现在是 Oracle ），那么 Jersey 毫无疑问就是事实上的标准，对于 Java REST 的初学者来说尽量要跟着标准走。当然，所有规范的实现，在用法上基本上没有差别，只是相对来说 Jersey 的实现更全面一些。

扩展阅读

- http://jersey.jave.net
- http://waylau.gitbooks.io/jersey-2-user-guide
- http://waylau.gitbooks.io/rest-in-action

Apache Shiro

干净利落地处理身份验证、授权、企业会话管理和加密。

Shiro 能做什么呢？

- 验证用户身份
- 用户访问权限控制，比如：
  - 判断用户是否分配了一定的安全角色。
  - 判断用户是否被授予完成某个操作的权限
- 在非 web 或 EJB 容器的环境下可以任意使用 Session API
- 可以响应认证、访问控制，或者 Session 生命周期中发生的事件
- 可将一个或以上用户安全数据源数据组合成一个复合的用户 "view"(视图)
- 支持单点登录 (SSO) 功能
- 支持提供 “ Remember Me” 服务，获取用户关联信息而无需登录

扩展阅读

- http://shiro.apache.org/reference
- http://www.waylau.com/apache-shiro-1.2.x-reference/

Maven

- 提供高级项目管理工具。它包含了一个项目对象模型 (Project Object Model) ，一组标准集合，一个项目生命周期(Project Lifecycle) ，一个依赖管理系统 (Dependency Management System) ，和用来运行定义在生命周期阶段 (phase) 中插件(plugin) 目标 (goal) 的逻辑。当你使用 Maven 的时候，你用一个明确定义的项目对象模型来描述你的项目，然后 Maven 可以应用横切的逻辑，这些逻辑来自一组共享的（或者自定义的）插件。
- 项目的整个生命周期，包括编 译，构建，测试，发布，报告等等

扩展阅读

- http://maven.apache.org
- http://www.waylau.com/categories/#maven

SVN

版本管理工具

SVN 与 Git 比较

- Git 是分布式的， SVN 是集中式的。 Git 每个开发人员从中心版本库 / 服务器上 chect out 代码后会在自己的机器上克隆一个自己的版本库。可以这样说，即使不联网，仍然能够提 交文件，查看历史版本记录，创建项目分支，等
- Git **把内容按元数据方式存储，而 SVN 是按文件：**所有的资源控制系统都是把文件的元信息隐藏在一个类似 .svn,.cvs 等的文件夹里。如果你把 .git 目录的 体积大小跟 .svn 比较，你会发现它们差距很大。因为 ,.git 目录是处于你的机器上的一个克隆版的版本库，它拥有中心版本库上所有的东西，例如标签，分 支，版本记录等
- Git **没有一个全局的版本号，而 SVN 有：** SVN 的版本号实际是任何一个相应时间的源代 码快照

Apache POI

Apache POI 项目的使命是创造和维护 Java API 操纵各种格式的文件 , 其中包括基于 Office Open XML 标准（ OOXML ）和微软的 OLE 2 Compound Document 格式（ OLE2 ）。总之，你可以使用 Java 读写 MS Excel 文件。此外，您可以使用 Java 读取和写入 MS Word 和 MS PowerPoint 文件。 Apache POI 是你的 Java Excel 解决方案（用于 Excel 97-2008 ）。包含了一个完整的 API 用于移植其他 OOXML 和 OLE2 格式。

扩展阅读

http://poi.apache.org/

http://www.waylau.com/apache-poi-handle-microsoft-documents/

MySQL

- 最流行的关系型数据库管理系统
- 使用 C和 C++编写，并使用了多种编译器进行测试，保证了源代码的可移植性。
- 支持 AIX、 FreeBSD、 HP-UX、 Linux、 Mac OS、 NovellNetware、 OpenBSD、 OS/2 Wrap、 Solaris、 Windows等多种操作系统。
- 为多种编程语言提供了 API。这些编程语言包括 C、 C++、 Python、 Java、 Perl、 PHP、 Eiffel、 Ruby,.NET和 Tcl 等。支持多线程，充分利用 CPU 资源。
- 优化的 SQL查询算法，有效地提高查询速度。
- 既能够作为一个单独的应用程序应用在客户端服务器网络环境中，也能够作为一个库而嵌入到其他的软件中。
- 提供多语言支持，常见的编码如中文的 GB 2312、 BIG5 ，日文的 Shift_JIS等都可以用作数据表名和数据列名。
- 提供 TCP/IP、 ODBC 和 JDBC等多种数据库连接途径。
- 提供用于管理、检查、优化数据库操作的管理工具。
- 支持大型的数据库。可以处理拥有上千万条记录的大型数据库。
- 支持多种存储引擎。
- MySQL 是开源的，所以你不需要支付额外的费用。
- MySQL 使用标准的 SQL数据语言形式。
- MySQL是可以定制的，采用了 GPL协议，你可以修改源码来开发自己的 MySQL 系统。
- 在线 DDL/更改功能，数据架构支持动态应用程序和开发人员灵活性（ 5.6[3] 新增）
- 复制全局事务标识，可支持自我修复式集群（ 5.6[3] 新增）
- 复制无崩溃从机，可提高可用性（ 5.6[3] 新增）
- 复制多线程从机，可提高性能（ 5.6[3] 新增）

扩展阅读

http://www.waylau.com/mysql-tutorial/

Who use MySQL

GNU/Linux

- GNU：Richard Stallman 在 1983 年 9 月 27 日公开发起的。它的目标是创建一套完全自由的操作系统（类 Unix ）。一个理由就是要“重现当年软件界合作互助的团结精神”。为保证 GNU 软件可以自由地“使用、复制、修改和发布”，所有 GNU 软件都有一份在禁止其他人添加任何限制的情况下授权所有权利给任何人的协议条款， GNU 通用公共许可证（ GNU General Public License ， GPL ）。即“反版权”（或称 Copyleft ）概念。
- 到了 1990 年， GNU 计划已经开发出的软件包括了一个功能强大的文字编辑器 Emacs[1] 。 GCC （ GNU Compiler Collection ， GNU 编译器集合），是一套由 GNU 开发的编程语言编译器。以及大部分 UNIX 系统的程序库和工具。唯一依然没有完成的重要组件就是操作系统的内核 (称为 HURD) 。
- 1991 年 Linus Torvalds 编写出了与 UNIX 兼容的 Linux操作系统内核并在 GPL 条款下发布。 Linux 之后在网上广泛流传，许多程序员参与了开发与修改。 1992 年 Linux 与其他 GNU 软件结合，完全自由的操作系统正式诞生。该操作系统往往被称为“ GNU/Linux”或简称为 Linux 系统

自由软件：

- 不论目的为何，有运行该软件的自由（自由之零）。
- 有研究该软件如何运行，以及按需改写该软件的自由（自由之一）。取得该软件源代码为达成此目的之前提。
- 有重新发布拷贝的自由，这样你可以借此来敦亲睦邻（自由之二）。
- 有改进该软件，以及向公众发布改进的自由，这样整个社群都可受惠（自由之三）。取得该软件源码为达成此目的之前提。

扩展阅读

http://www.mysql.com/

http://www.gnu.org/

其他前端框架

- EasyUI : 后台管理界面框架　1.4.1　http://www.jeasyui.com/
- Boostrap ； 前台分析界面框架 3.3.1 http://getbootstrap.com/
- JQuery:　 2.1.3　　http://jquery.com/
- Echarts: 图表分析　2.1.10　http://echarts.baidu.com/
- Kindeditor: 文本编辑器 4.1.10　http://kindeditor.net/

学习 / 工作 方法论

- 避免过度社交
- 专注
- 工具的选择
- 善用文档
- 基本的英文
- 信息检索
- 23:30 前躺下

总结

架构的目的

- 提供可靠谱的系统服务
- 减少开发人员的工作量

Q&A