1. WEB应用的持续集成和自动化运行

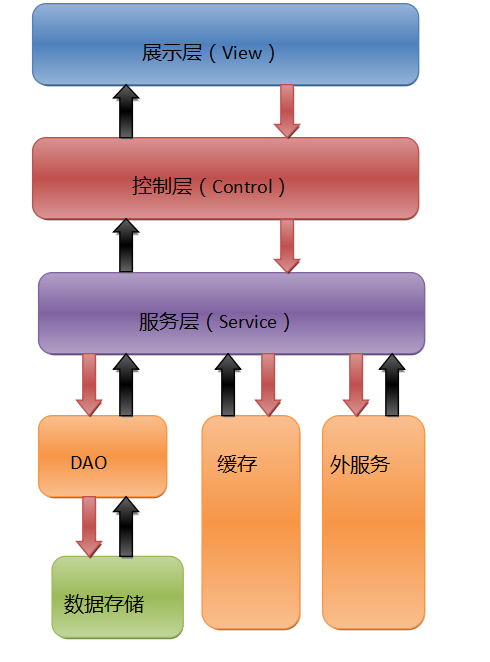
在本项目中，采用了基于linux系统的jenkins的持续集成解决方案，以github作为版本控制工具。在部署端采用SHELL脚本作为运行脚本。

采用持续集成的目的是为了减少风险，提前并多次进行集成测试以期提前发现集成阶段才能发现的错误。并且可以调高开发过程中软件的可见度，并且可以在任何部署软件。

自动化运行采用SHELL脚本，Shell本身是一个用C语言编写的程序，它是用户使用Unix/Linux的桥梁，用户的大部分工作都是通过Shell完成的。Shell既是一种命令语言，又是一种程序设计语言。作为命令语言，它交互式地解释和执行用户输入的命令；作为程序设计语言，它定义了各种变量和参数，并提供了许多在高级语言中才具有的控制结构，包括循环和分支。采用SHELL脚本进行可控的自动单元测试和自动部署运行。

1. WEB的架构设计

当前主要采用的架构是MVC架构，即（模型层-表示层-控制层）。在最初的JSP网页中，像数据库查询语句(SQL query)这样的数据层代码和像HTML这样的表示层代码是混在一起。虽然有着经验比较丰富的开发者会将数据从表示层分离开来，但这样的良好设计通常并不是很容易做到的，实现它需要精心地计划和不断的尝试。MVC可以从根本上强制性地将它们分开。尽管构造MVC应用程序需要一些额外的工作，但是它带给我们的好处是毋庸置疑的。



首先，多个 View 能共享一个 Model 。如今，同一个Web应用程序会提供多种用户界面，例如用户希望既能够通过浏览器来收发电子邮件，还希望通过手机来访问电子邮箱，这就要求Web网站同时能提供Internet界面和WAP界面。在MVC设计模式中， Model 响应用户请求并返回响应数据，View 负责格式化数据并把它们呈现给用户，业务逻辑和表示层分离，同一个 Model 可以被不同的 View 重用，所以大大提高了代码的可重用性。

其次，Controller 是自包含（self-contained）指高独立内聚的对象，与 Model 和 View 保持相对独立，所以可以方便的改变应用程序的数据层和业务规则。例如，把数据库从MySQL移植到Oracle，或者把RDBMS数据源改变成LDAP数据源，只需改变 Model 即可。一旦正确地实现了控制器，不管数据来自数据库还是LDAP服务器，View 都会正确地显示它们。由于MVC模式的三个模块相互独立，改变其中一个不会影响其他两个，所以依据这种设计思想能构造良好的少互扰性的构件。

此外，Controller 提高了应用程序的灵活性和可配置性。Controller 可以用来连接不同的 Model 和 View 去完成用户的需求，也可以构造应用程序提供强有力的手段。给定一些可重用的 Model 、 View 和Controller 可以根据用户的需求选择适当的 Model 进行处理，然后选择适当的的 View 将处理结果显示给用户。

WEB应用目前主流的开发平台是J2EE和.NET framework，同时有部分新兴的开发平也在发展，例如本次项目所采用的NODEJS。

各个平台之间优劣：

|  |  |  |
| --- | --- | --- |
| 平台 | 优势 | 劣势 |
| J2EE | 目前使用最多的WEB应用后台，与各主流数据库兼容性良好。有大量的教程和成熟框架可供使用。 | 需要JAVA语言基础。 |
| .NET framework | 便于开发，可以快速进行小规模开发，拥有mircosoft系列软件的直接支持 | 不具备异常机制，当进行大规模开发时难以进行测试 |
| Nodejs | 非阻塞式API，应用JS语言 | 单线程/单进程 |