TM书审资料

题目： Seasar2迁移Spring MVC

姓名:张吉

部门:DA05

目录

第一章 绪论 1

第二章 Seasar2迁移Spring MVC 2

2.1 Seasar2和Spring MVC简介 2

2.2 Seasar2迁移Spring MVC的实际应用 2

2.2.1 Seasar2实际应用的基盘概要 2

2.2.2 Seasar2到Spring MVC的迁移策略 2

2.2.3 Spring MVC的优点 5

2.3实际问题解决 6

2.3.1业务逻辑层全局变量生命周期的问题 6

2.3.2事务控制注入失败问题 7

第三章 实际应用再构筑后分析 9

3.1 项目的不足 9

3.2 Angular开发建议 9

第四章 总结 11

1. 绪论

Seasar2是一款基于DI和AOP设计理念的日本国产的开源Java框架。其中Struts协作功能“S2Struts” 是Seasar2用DI和AOP技术对Struts进行封装。

Struts是一个热门话题比较多的框架，以漏洞多出名，影响巨大，虽然可以通过打补丁来修复，但是替换成本高，手续繁琐。更何况Seasar2所使用的Struts协作功能已于2016年9月26日停止支持。鉴于Struts不停地出现新漏洞，继续使用Seasar2也是一个很大的风险。因为即使出现新的漏洞也不会提供安全补丁和修复，所以现在有不少Seasar2项目开始往Java EE 或 Spring框架迁移。本文阐述的实际应用是从Seasar2/Struts1迁移到Spring MVC框架。

相比较而言，Spring MVC可以提高应用程序的性能、可扩展性、安全性和开发效率，并且可以提供更好的开发体验和可维护性。

本文将结合自身参与的项目对Seasar2迁移Spring MVC开发过程中遇到的一些技术知识进行阐述和总结。

1. Seasar2迁移Spring MVC

2.1 Seasar2和Spring MVC简介

Seasar2是一款日本国产的开源Java框架，并且曾经在日本非常流行。它通过减少所需的配置文件数量并单独编写依赖项，可以实现高效的程序开发。与传统框架相比，它具有更好的可扩展性、处理规范变更的能力以及单元测试能力，可有效减少整体工时和成本。Seasar2不仅可以在独立环境中运行，还可以在应用程序服务器（例如“Apache Tomcat”等servlet容器）上运行。有DI容器“S2Container”、AOP功能“S2AOP”、数据库访问功能“S2Dao”、Struts协作功能“S2Struts”等各种相关软件和工具。其工作流程就是：1).客户端请求提交到ActionServlet。2).ActionServlet将数据放到相应的form对象的成员变量中。3).由ActionServlet根据配置文件struts-config.xml里的映射关系找到对应的Action，相应的formBean一并传过去。4).Action里的excute()方法里执行相应的业务逻辑(调用其他的业务模块)，完成后返回一个ActionForward对象。5).ActionServlet根据返回的ActionForward对象，找到对应的jsp页面。6).jsp页面将结果展现在客户端。

但是，2016年9月26日，宣布终止对Seasar2的支持。

Spring MVC已经成为目前主流的MVC框架，是一种基于Java的、实现了MVC设计模型的请求驱动类型的轻量级Web框架，顾名思义，其思想是将Web应用程序分为M（模型）、V（视图）和C（控制器）。

Spring MVC主要由前端控制器（DispatcherServlet）、处理器映射器（HandlerMapping）、处理器（Controller）、视图解析器（ViewResolver）、视图（View）组成。其工作流程就是：1).客户端请求提交到DispatcherServlet。2).由DispatcherServlet控制器查询一个或多个HandlerMapping，找到处理请求的Controller。3).DispatcherServlet将请求提交到Controller。4).Controller调用业务逻辑处理后，返回ModelAndView。5).DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图。6).视图负责将结果显示到客户端。

除了基础功能外，Spring MVC还提供了许多其他功能。例如文件的下载、文件的上传、拦截器、JSON处理、静态资源处理等，以帮助开发人员更加高效地开发和管理应用程序。

总之，Spring MVC是一个功能强大、非常灵活、易于学习使用的MVC框架，可以帮助开发人员构建出高质量、可维护的应用程序。

2.2 Seasar2迁移Spring MVC的实际应用

2.2.1 Seasar2实际应用程序的基盘概要

1. 目的

应用程序的基盘主要提供以下功能，以减轻开发负担、均衡质量、提高可读性和可维护性。

·定义各层流程的模板（基底class），如：表现层、业务逻辑层。

·访问DB的功能

·业务应用程序的共通处理（utility工具类）

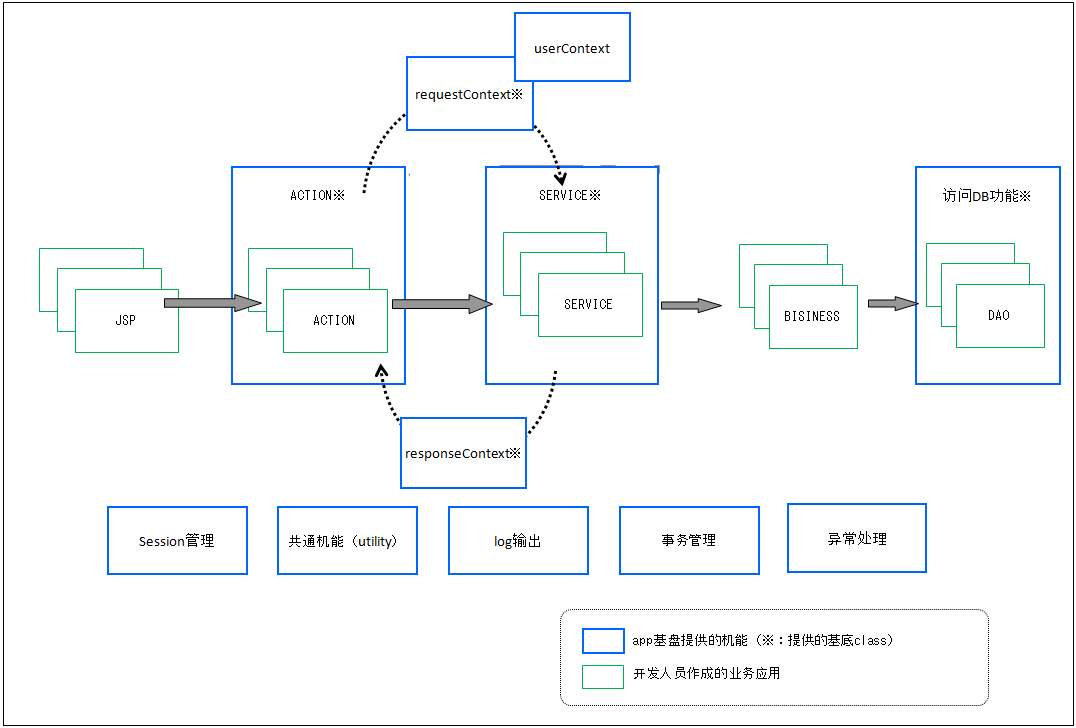
1. 前提条件

应用程序的基盘的各种组件以及使用的扩展框架。

主要组件以及版本如下。

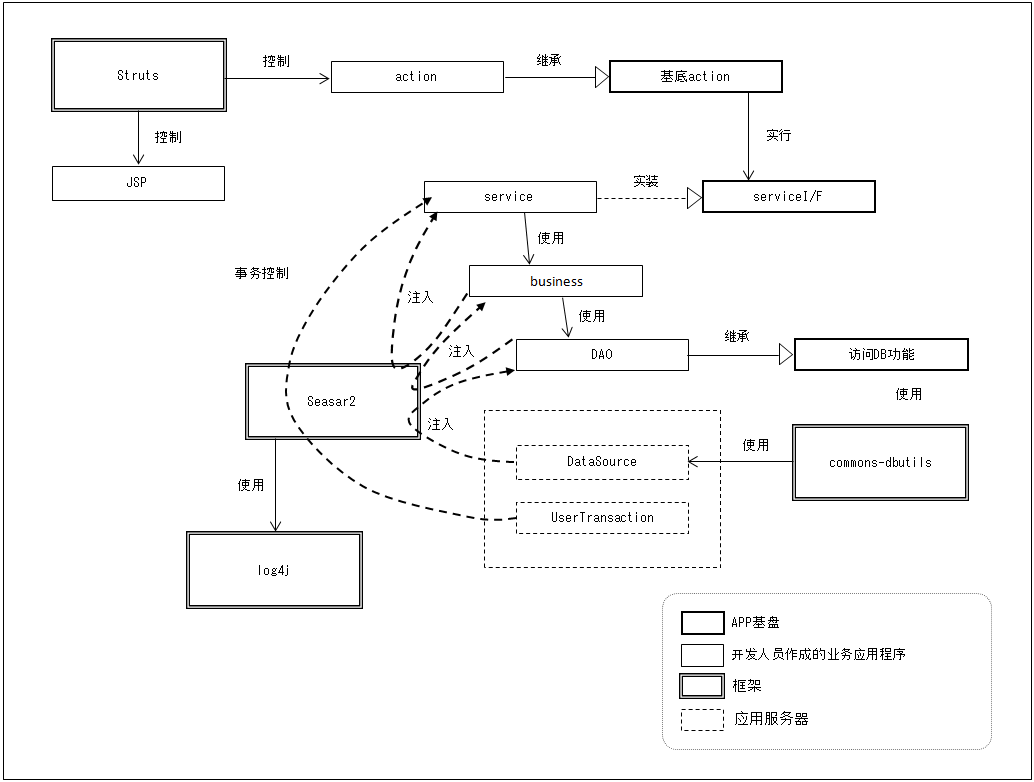
|  |  |  |
| --- | --- | --- |
| 组件 | 版本 | 概述 |
| Struts | 1.3.10 | WEB应用程序的集成框架 |
| Seasar2 | 2.4.17 | DI（依赖注入）容器框架 |
| commons-dbutils | 1.1 | 方便访问JDBC的lib |
| log4j | 1.2.15 | 日志输出的工具类 |

1. 系统构成

整体的一个应用程序结构图

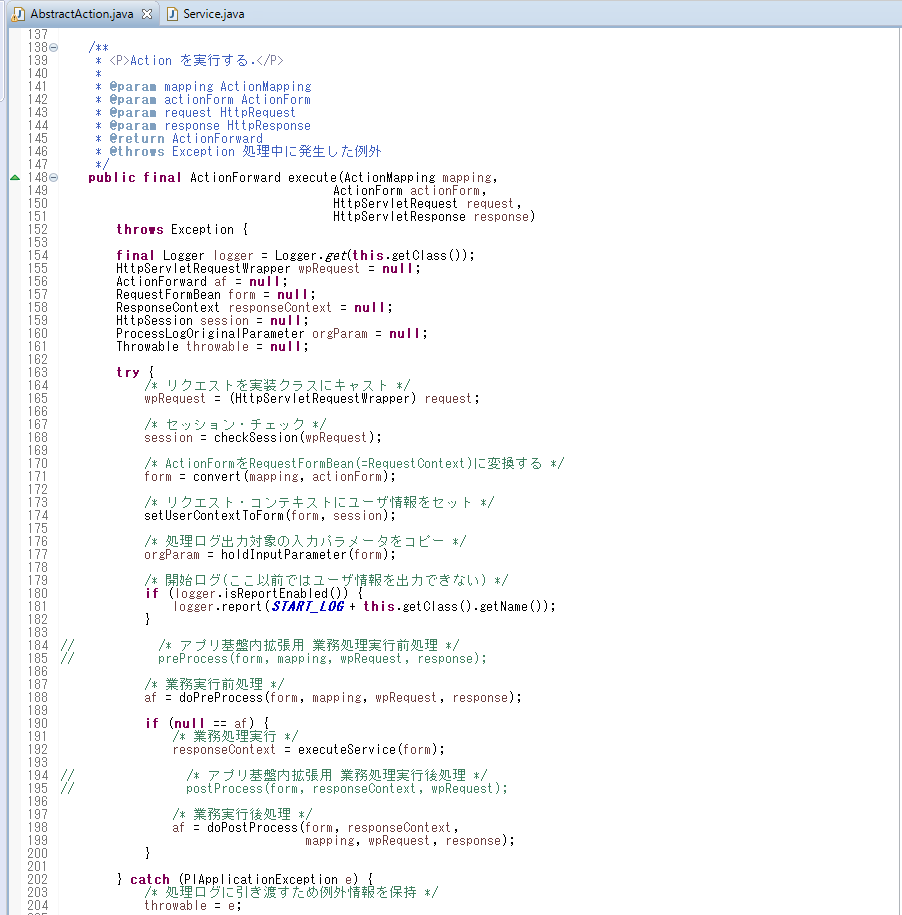
|  |  |
| --- | --- |
| 组成 | 说明 |
| JSP | 描述画面构成的文件。主要用HTML和 JavaScript编写。 |
| ACTION | 作为服务器端最先处理的class。用于画面表示的class，描述了处理画面表示用项目的值的过程。不涉及业务逻辑的东西。 |
| SERVICE | ACTION之后要处理的class。为后续业务处理挂上事务而创建的一个类。 |
| BISINESS | 描述业务逻辑处理的class。 |
| DAO | 为 BISINESS 类访问DB而创建的一个类。 |

外部框架与应用程序基盘的关系图



* 1. Action基底类

被业务应用程序的各个Action类所继承，为这些Action类定义了一个共通流程。





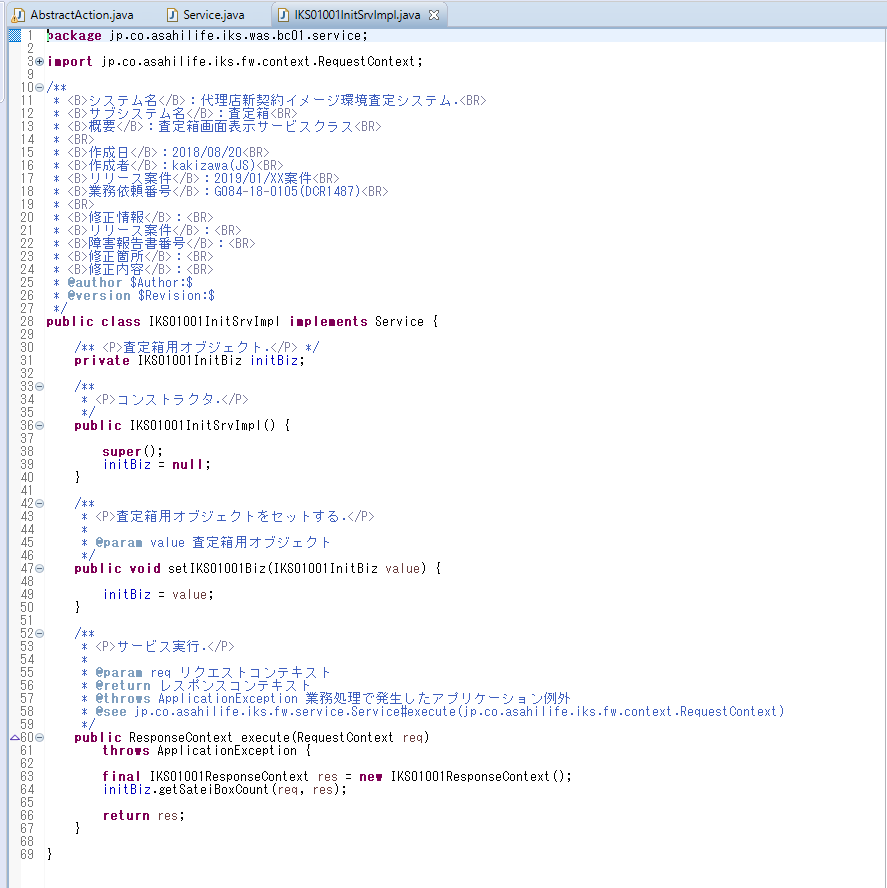


其中包含了requestContext/responseContext和ActionForm之间的填充、前处理、执行service处理、后处理、异常处理等。

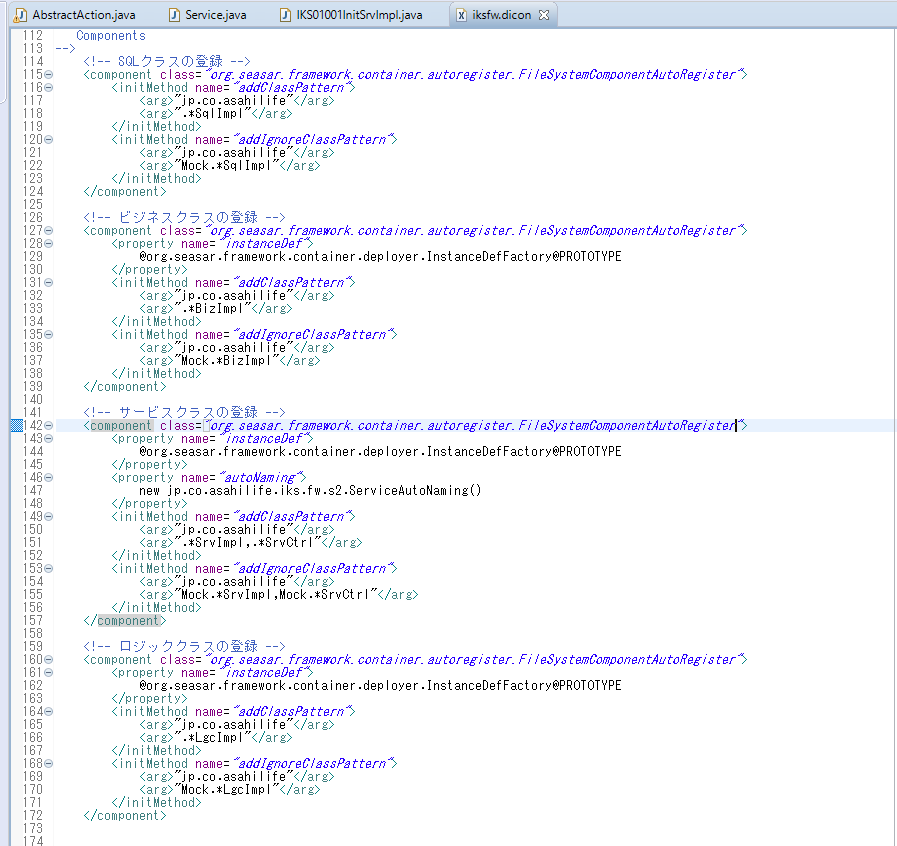
* 1. serviceI/F

定义了一个统一的service接口。

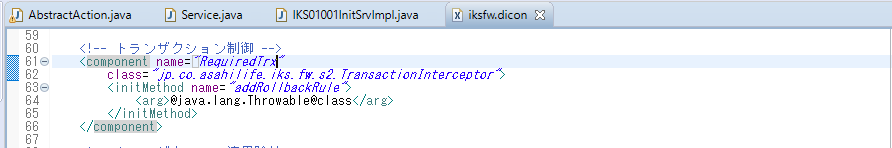
* 1. service

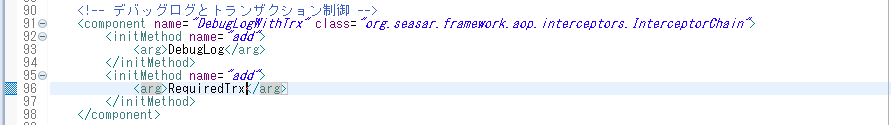


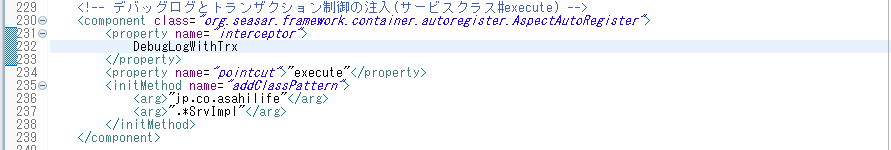
使用Seaser2的DI（依赖注入）机制将要执行的业务层class（IKS01001InitBiz）提供给service层。



使用Seasar2的AOP（面向切面的编程）将事务切入在service层的execute()。



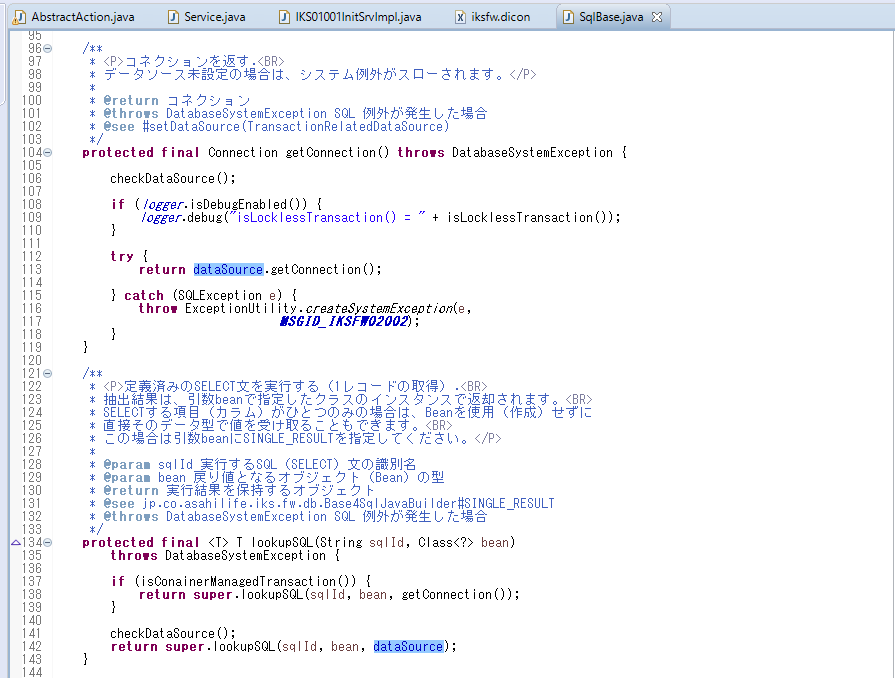




* 1. 访问DB的基底类

提供实现SQL的执行逻辑的基底class，并且定义了访问DB的方法。

它还负责隐藏事务控制，例如获取和关闭数据库的连接。

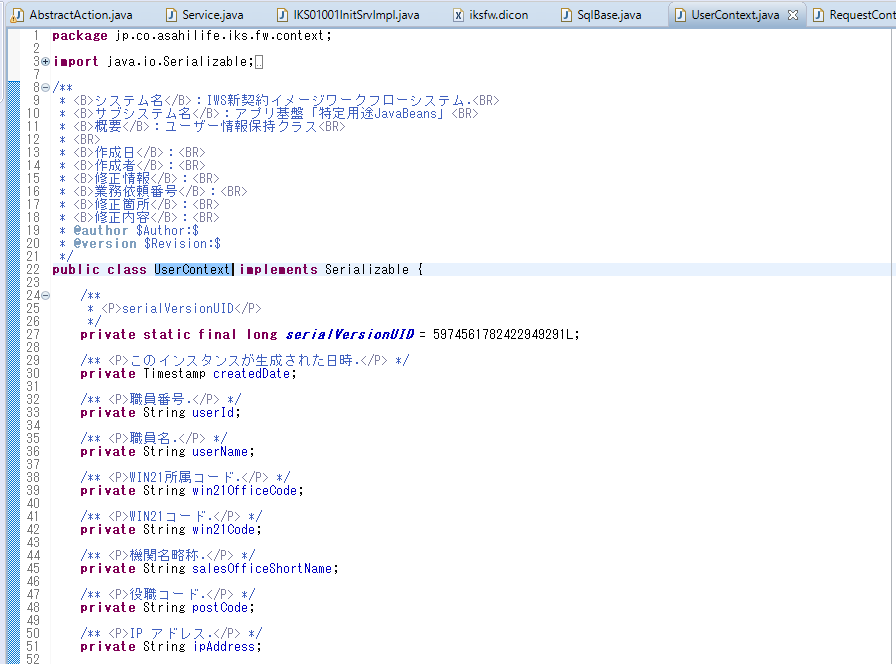


* 1. Context

requestContext：将在表现入力和设定的参数传递给业务层。

responseContext：将在业务层执行的结果传递给表现层。

userContext：保存用户ID、登录信息等的JavaBean。 由用户认证函数生成。在事务中userContext保持在requestContext中，并且根据需要在表现层和业务层中引用它。



* 1. 共通机能

提供共通机能模块。

分为各层的共通机能和特定层的独自机能。

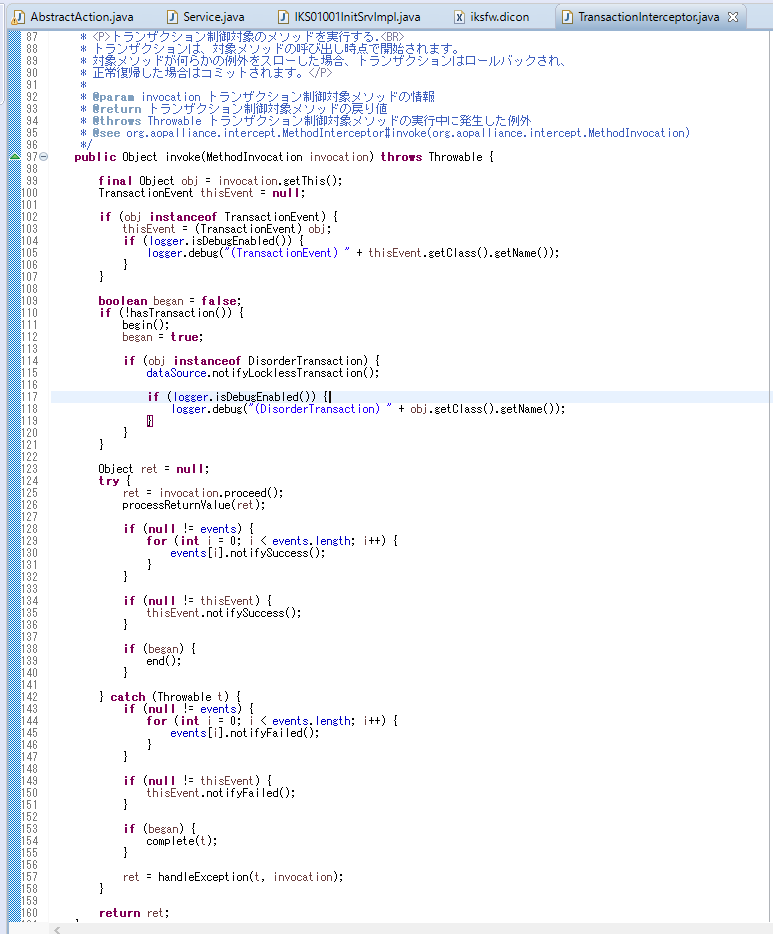
例如：message管理的机能、例外处理机能、各种utility工具类

* 1. 日志输出

提供错误日志、跟踪日志等输出功能。

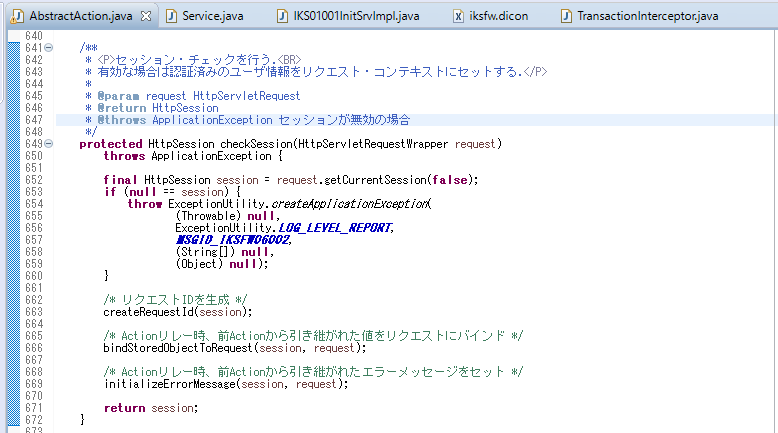
* 1. 事务控制

提供业务逻辑的事务控制功能。



* 1. Session管理

提供管理HTTPSession功能。



* 1. 异常处理

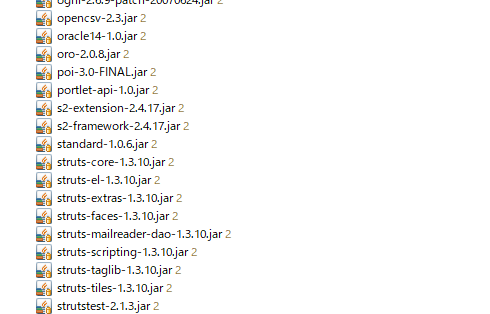
提供集中处理系统所有异常的功能。

2.2.2 Seasar2到Spring MVC的迁移策略

只有了解了Seasar2/Struts1的工作原理和机制，才能针对性的将应用程序迁移到Spring MVC的框架下。所以结合实际项目的开发，我总结了以下几个迁移方针。

1. 替换基本的框架库

Seasar2/Struts1



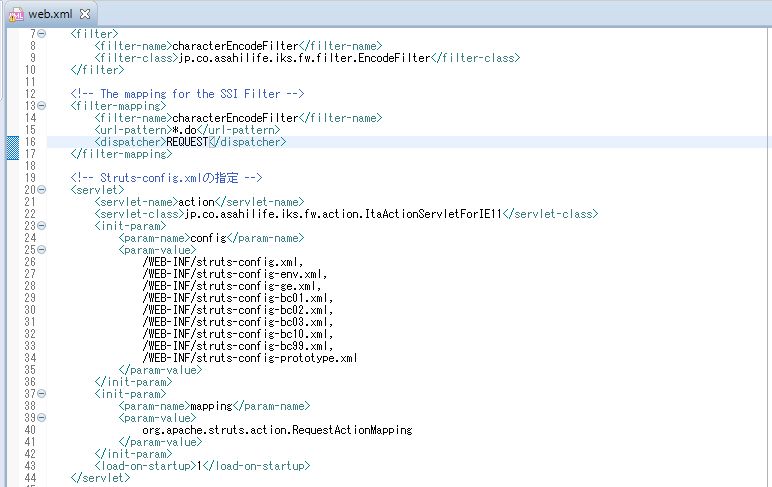
替换Spring MVC



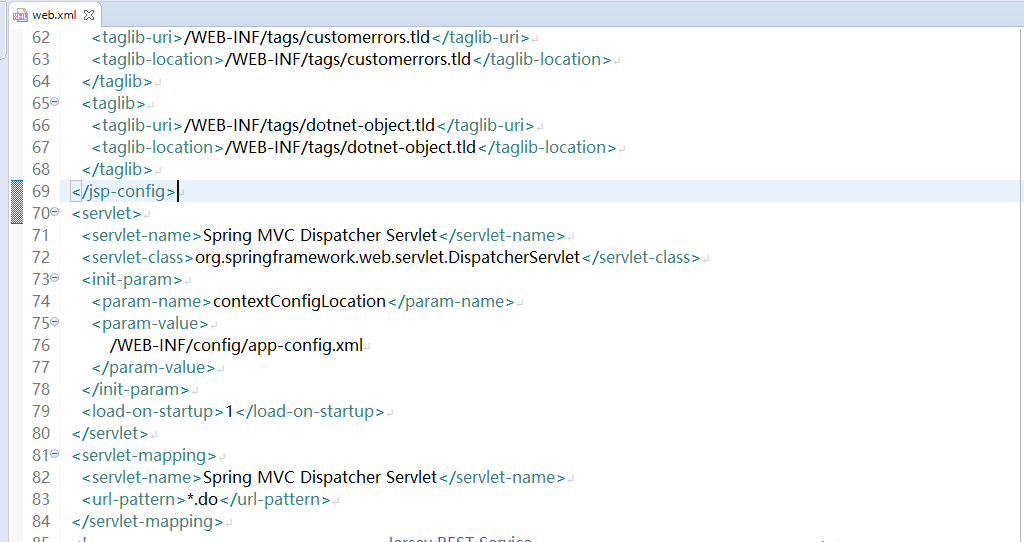
1. 替换web.xml配置文件

在此步骤中，我们得知道DispatcherServlet是Spring MVC的核心组件，所以web.xml里必须将DispatcherServlet作为前端控制器。

Seasar2/Struts1



替换Spring MVC



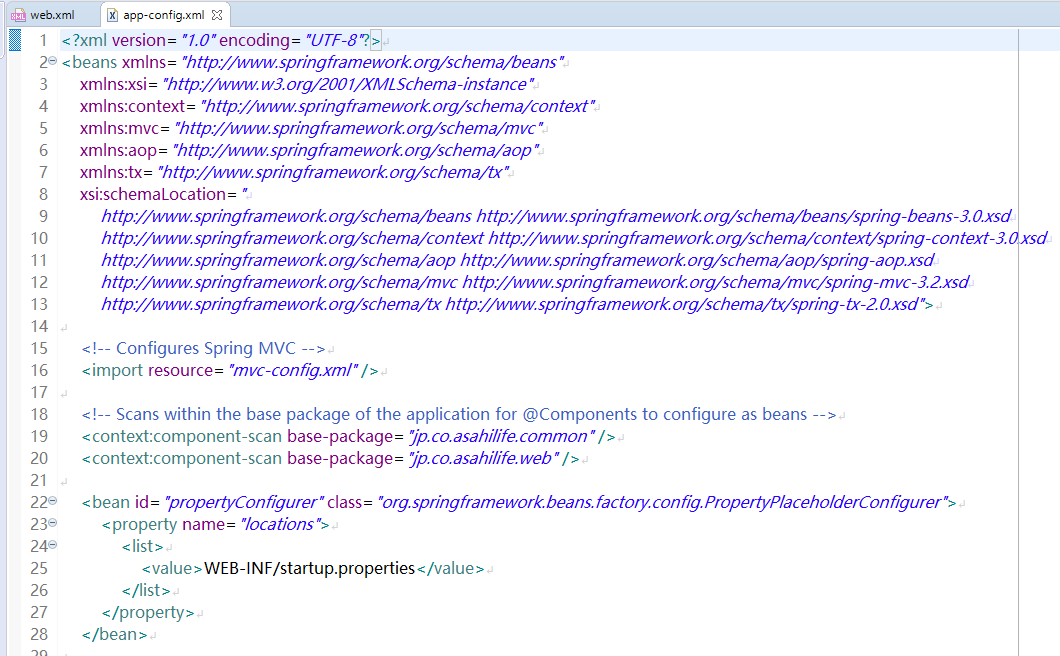
1. 替换Struts本身的配置文件

将所有struts配置文件替换为spring配置文件如下

Seasar2/Struts1：struts-config.xml



替换Spring MVC：app-config.xml



1. Spring MVC框架的拦截器的利用

1.Struts1框架的ActionServlet根据配置文件struts-config.xml里的映射关系找到对应的Action，找不到就报错。而Spring MVC框架的DispatcherServlet查询HandlerMapping时就需要有个拦截器先拦截住请求，对请求的参数进行匹配，匹配不正的场合报错。

2. Struts1框架在ActionServlet将请求送到Action时进行SessionCheck。相应的在Spring MVC中由DispatcherServlet将请求送到Controller前需要添加拦截器对Session进行Check。

1. 修改JSP文件

因为Struts框架和Spring MVC关于JSP的标签也有所不同，所以相对应的Struts标签要转换成Spring MVC的标签，例如下面常见的一些标签。

|  |  |
| --- | --- |
| Seasar2/Struts1 | Spring MVC |
| <%@ taglib uri=*"http://struts.apache.org/tags-bean"* prefix=*"bean"*%>  <%@ taglib uri=*"http://struts.apache.org/tags-logic"* prefix=*"logic"*%> | <%@ taglib uri=*"http://struts.apache.org/tags-bean"* prefix=*"bean"*%>  <%@ taglib uri=*"http://struts.apache.org/tags-logic"* prefix=*"logic"*%> |
| <**logic:equal** name="targetRecord" property="checkCategory" value="21"> | <**c:if test**="${targetRecord.checkCategory == '21'}"> |
| <**bean:write** name="SateibakoCountBean" property="recordCountA101"/> | <**c:out value**="${IKS01001ResponseFormBean.sateibakoCountBean.recordCountA101}" /> |
| etc. | etc. |

1. 整合action、service、business类文件
2. 因为Struts一个请求就需要对应一个Action，所以我们会发现一个画面的不同请求需要对应多个Action，这就大大增加了代码量。而我们可以利用Spring MVC不同的方法只需要各自获取请求中的参数的特点，可以把各个机能的Action合并到一个Controller里。例如：

|  |  |
| --- | --- |
| Struts | Spring MVC |
| IKS03001InitAction | IKS03001Controller  ・init()  ・search ()  ・download () |
| IKS03001SearchAction |
| IKS03001DownloadAction |

1. 因为现行的实际应用程序的事务是在Service层开启，而Spring MVC没有Service层，但是为了事务开启的时节点和现行保持一致，所以迁移后的应用程序保留了Service层。在Service类上添加@Transactional注解。
2. 如上记所说，那么只需要把现行应用程序的Business类合并转换成Logic类就可以了，这样省去了将Service类和Business类一起合并到Logic类所带来的麻烦。
3. 转换的过程中需要给Controller层加上@Controller注解，Service层加上@Service注解，Logic层加上@Component注解，这样就可以把这些类交给Spring管理。同时我们需要在配置文件app-config.xml里配置扫描注解，就可以在Spring MVC容器中注册带有注解的bean了。通过@Autowired在这些bean被引用的地方实现自动装配。
4. 数据库从DB2换成Postgre表结构再构建（客户的要求）

现行的实际应用程序利用的Tomcat服务器，迁移后的应用程序利用的Jboss服务器，所以在Jboss的配置文件standalone.xml里设置Postgre数据库的驱动。因为数据库的不同，DDL和DML需要根据最新的设计进行构建。

与此同时，关于batch中执行DB2命令改成psql命令。

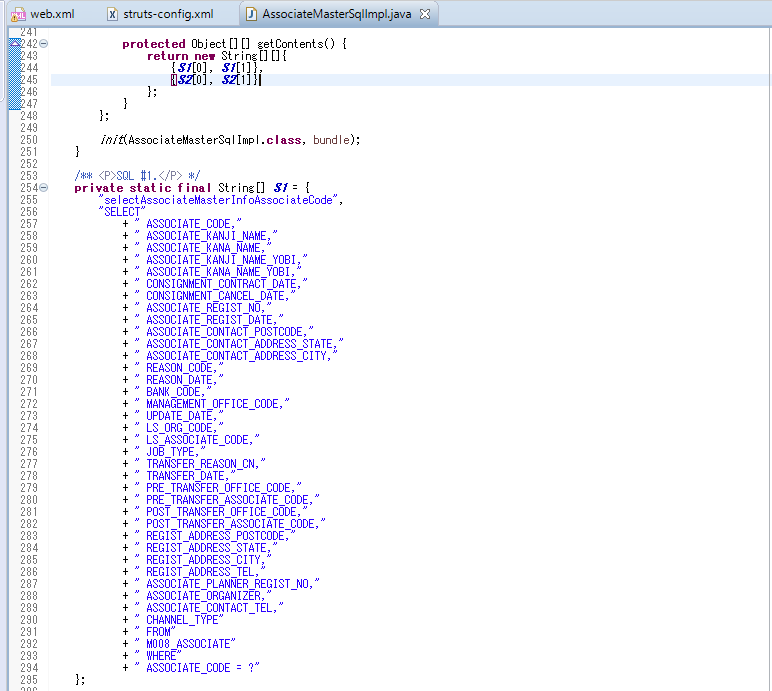
例如：使用SQL文件进行DB操作的场合

db2 -f ${SQL\_FILE}改成： psql –f ${SQL\_FILE}

例如：使用SQL语句进行DB操作的场合

db2 -x "${SQL}"改成：psql –c "${SQL}"

1. 数据库框架换成ibatis（offshare的提案）



现行的实际应用程序把SQL语句写在java源程序里，鉴于维护的便利性，采用了ibatis框架。ibatis把SQL语句从java源程序中独立出来，放到单独的xml文件中编写，给程序的维护也带来了极大的便利。ibatis封装了底层JDBC API的调用细节，并能自动将结果集转换成JavaBean对象，大大简化了Java数据库编程的重复工作。更重要的是，简单易于学习，易于使用。

2.2.3 Spring MVC的优点

随着软件行业的技术日新月异的发展，Spring MVC也逐渐地被Spring boot替代。但是因为客户在SpringMVC框架上做过很多项目积累了许多经验，采取了相对保守稳健的策略。所以将项目迁移到SpringMVC框架上，相比较于Seasar2/Struts1，SpringMVC框架具有很大的优势。

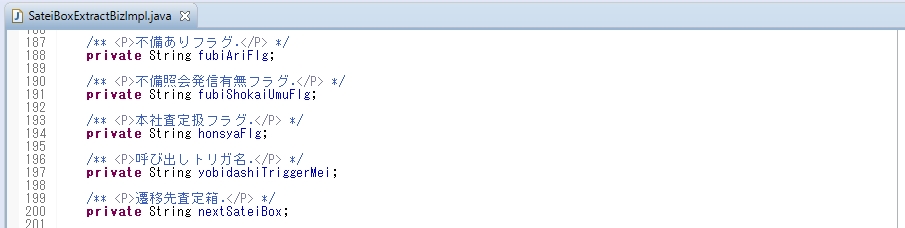
1. Spring MVC安全性更高，Struts1框架是类级别的拦截，每次request请求Struts1都会为之创建一个action，然后将数据注入到实体bean中，所以在Struts1中，一个action对应一个request上下文。Spring MVC是方法级别的拦截，每个request对应一个方法，然后通过注解将数据注入到对应的实体中，所以在Spring MVC中，一个方法对应一个request上下文，粒度更小，所以说Spring MVC更安全。
2. Spring MVC直接使用注解的方式，配置更加的简洁清晰。开发过程中只需要更多的考虑业务的开发。
3. Spring MVC的异常处理机制更加的优秀。Struts1因为在Action的execute方法声明时就抛出了Exception异常，所以我们无需再execute方法中捕捉异常，仅需在struts.xml 中配置异常处理；在Spring MVC中采用统一的异常处理框架，配置异常处理之后直接在执行方法中抛出对应的异常，更加的准确，显示更加友好的用户界面提示。
4. 相对于Struts1，Spring MVC的参数传递更加方便。Spring MVC直接使用对应的实体去接收form提交的表单信息，而Struts1要在form中指定对应的实体，否则Struts1无法区分form中的数据对应哪个实体 。

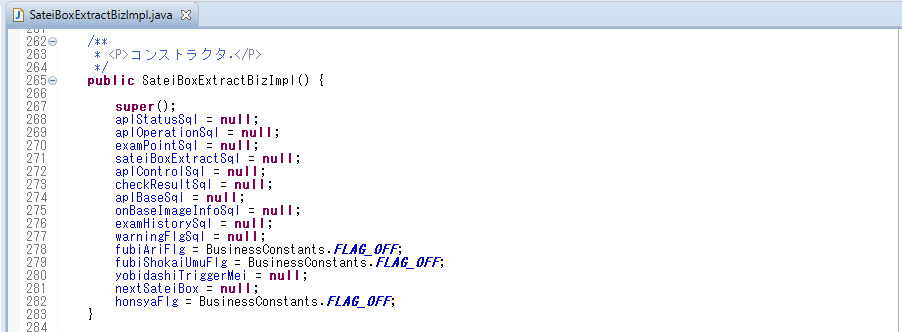
2.3 实际问题解决

2.3.1 多线程使用单例的问题

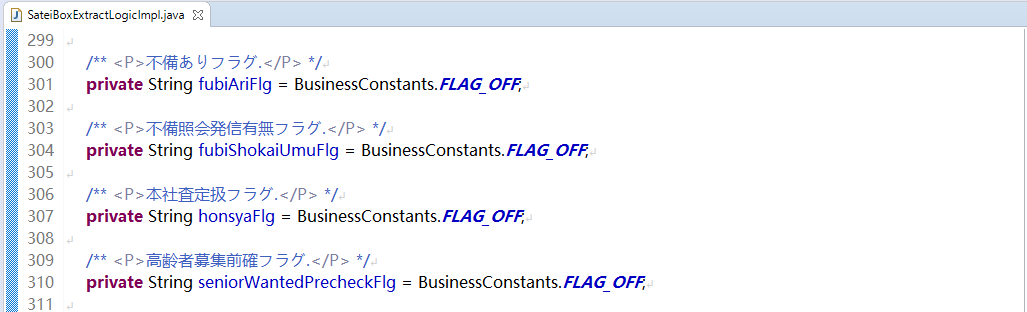
在使用了Spring MVC框架后 ，我们都会选择Spring的IOC功能来管理Bean，而不是用到时去new一个。Spring管理的Bean默认是单例的（即Spring创建好Bean，需要时就拿来用，而不是每次用到时都去new），但有时候单例并不满足要求（比如Bean中不全是方法，有成员），如下所示

现行：





迁移后：



这样就会出现了以下的问题。

当多线程时，当前线程就会共享上一个线程的全局变量。就会导致当前线程出现预期之外的结果。这里我们就需要考虑多线程使用单例出现问题的可能性。

在类上加上@Scope(“prototype”)注解，可以通知Spring把被注解的Bean变成多例。这样多线程的时候就会调用不同的Bean互不影响，上述问题就可以得到解决。而这样的情况往往会被忽视，因为开发过程中程序员都是独自进行开发调试。所以多线程使用单例的问题需要值得重视。

2.3.2事务控制开启失败问题

正如之前我们所说的给Service类上添加@Transactional注解来进行开启事务控制。但是往往我们忽视了@Transactional只能作用在public方法上，所以如下Service类上加@Transactional是无效的。



因为batch侧的Service层的父类（AbstractBatchService）定义了**protected**的executeEx方法，所以Service层的方法也是**protected**修饰的，导致@Transactional失去作用。



所以在executeEx方法执行之前，在父类（BatchService）的invoke方法里手动开启事务就可以解决问题。如下：



正如这种情况我们不能一味地添加了注解就以为事务可以开启，必须得需要了解@Transactional注解的作用方式。

第三章 实际应用迁移后分析

前端这个工作目前看来必不可少的。只要不跨出浏览器的限制，在网页中可以实现各种功能。因为未来可能会有更多的Angular项目。经过这个项目我对Angular开发项目进行了简短的分析和总结。

* 1. 项目的不足

1. Angular开发经验者少，因为项目原因，很少会有前端Angular开发的项目，近几年才逐渐接触。经验者少的情况下，在开发时没有办法熟练的运用Angular提供的功能，很多开发人员写代码时还停留在最基础逻辑上，不会调用提供的API，这样就会造成代码量过于庞大，本来一行可以解决却写了很多行，而且在性能上也会有所欠缺。

改进：对初学者进行统一开发前教育，强制规定一些必要的Rule。加强开发人员Angular的学习，在项目中尽可能做到一带一，由经验者引导初学者编码。

1. 虽然Angular的模块化让项目分配管理起来很方便，但是这样往往会造成开发人员只是埋头做自己的事情，忽略了相互之间的关联，代码相互之间的兼容性较差从而导致合并的时候会有一些难于修改的错误，只能通过打补丁的方式强行修复，最终导致代码的可读性很差，也增加了后期维护的难度。

改进：要加强开发人员之间的沟通，尤其是有关联模块之间的开发人员，在开发前商议好如何做到业务上的关联性，降低代码兼容性差的风险。

1. 项目中重复代码较多，虽然在设计阶段已经将一些必要的处理提炼为函数，但是在实际开发作业中会有很多设计外的逻辑和处理存在，因为没有很好的管理每个人的开发程序，导致了这一问题。

改进：组长每天check组员的代码，将功能相似的代码封装成函数，降低代码的耦合度。

1. 设计人员很多不懂技术，导致了开发人员拿到设计书的时候面对设计无从下手。

改进：对设计人员提供基础的开发知识培训，可以让设计者在设计时更多的考虑到开发的可能性。

1. 开发人员对自己需要开发的部分没有做一个计划就匆匆开始编码，导致有的时候写到一半发现串联不起来，需要重新考虑，这样大大降低了开发效率。

改进：加强开发人员教育，培养先做规划在开发的好习惯。

3.2 框架迁移建议

针对公司特点和自身理解对将来框架之间迁移提出一些建议。

1. 加强初学者教育，熟悉Angular框架的基本概念和特性，包括组件、模块、服务、依赖注入等。这些概念是理解和使用Angular的基础。
2. 对设计人员也要提高技术知识储备，从而减轻开发人员的压力。
3. 最好不要在项目中期为了追赶进度加入不熟悉Angular的开发人员，因为可能会事半功倍。
4. 要求开发人员编码之前先进行规划避免出现返工，从而提高开发效率。
5. 在开发阶段可以给每一个领域配置一个技术和经验足够的成员，来把握代码的整体性和统一性，避免出现各写各的以及大量功能相似一样的代码。
6. 将开发过程中出现的统一要求以及特殊问题书面化管理，避免时间长后再犯同样的错误。后期维护的时候，其他开发人员也可以参考，从而避免同样的问题反复发生。
7. 组长每天都check组员的代码，保证每天的代码都是相对规范的。
8. 加强开发人员之间的沟通，以避免出现开发偏离的风险。
9. 将每个Angular项目中用的新技术，新功能整理成书面资料，为以后的Angular开发项目积累经验。
10. 避免使用刚发布的Angular版本，优先考虑相对稳定的老版本。

第四章 总结

随着技术的发展，在以后我们会经历越来越多的框架迁移项目，每一个项目的在开始时都需要了解现行的框架，掌握迁移的框架，以及项目的结构部署。除了网上查阅相关文档，我们也可以借鉴以前成熟的相关系统进行框架搭建，从而节省时间提高效率。同时也要学习储备各种技术知识，增加我们的开发水平，不仅仅拘泥于一个项目的成功开发。

作为一名软件开发人员，通过工作经历和自我学习对Seasar2迁移Spring MVC框架相关开发进行了简单的讲述，希望此文会对公司以后在Seasar2迁移Spring MVC框架还有Spring MVC框架开发项目中有所帮助。

业精于勤荒于嬉，我会继续向公司的前辈学习，不断锻炼自己，提高自己解决实际问题的能力，并在工作中克服各种困难，积极、热情、细致地对待每一项工作。能更多的为同事们解决一些技术问题。