Day2 2月28日

`软件说明` `概要` `日志` `开发模型` `文档`

以原型法为基础，归纳用户需求，快速原型，迭代开发，用户参与

Topsun软件开发模型 SaaS PaaS

`快速原型` `同行评审`

过程管理 技能管理 团队管理

（1）过程管理： KUP、XP、CMMI

（2）技能管理： 工具、架构、代码

（3）团队管理： 文化、制度、价值链

`SPP` **精简过程管理** （RUP 简化）

重要三点

过程管理 开发过程 开发计划 时间、成本、质量

（1）需求分析->需求规格说明书<-需求设计（概要设计、详细设计）

（2）开发计划 时间节点怎么分配的

（3）测试报告 上线准备 支持与服务 **闭环控制软件开发整个过程管理** 配置管理

JAVAEE

N-Tier Architecture

```
表现层 DTO (Object bean)

service/business 服务层/业务层 DTO

Persistence JDBC

DB
```

JDBC-API

展现层 JSP-Spring boot

数组注意事项

```
String[] args;
int []arg1=new int[3];
int[][]arg2=new int[3][];

dataType[] arrayRefVar;   // 首选的方法
或
dataType arrayRefVar[];  // 效果相同，但不是首选方法
```

EasyUI 过时

Bootstrap响应式布局

基于百分比缩放布局

```
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span2">
      <!--Sidebar content-->
    </div>
    <div class="span10">
      <!--Body content-->
    </div>
  </div>
</div>
```

Handlerbars.js Mustach.js

数据可视化

iCharts Google Charts ECharts D3.js

Maven

项目构建工具

Jar包依赖关系管理

Others:

Structs 地基、构建

Hibernate 面向对象与传统数据库间映射

```
hibernate是一个框架，是用来操作数据库的。它把数据库中的表，转换成java类，通过xml文件来实现类和表之间的映射。这样的好处在于，可以面向对象的思想来操作数据库
struts是一个框架，它在网站中起到了控制层的作用。例如表单提交、获取数据、进行一些业务操作等，都是在struts里实现的
spring是一个框架，是用来把struts和hibernate连接在一起的。通过它，可以配置struts中的action要调用哪个业务逻辑层的service类，service类要调用哪个数据操作层的dao类
```

Spring 管理对象

Life Portal 门户

Mybatis 对JDBC封装 简化数据库开发

Web services —— Restful 网上资源调取

> 采用REST架构风格，对于开发、测试、运维人员来说，都会更简单。可以充分利用大量HTTP服务器端和客户端开发库、Web功能测试/性能测试工具、HTTP缓存、HTTP代理服务器、防火墙。这些开发库和基础设施早已成为了日常用品，不需要什么火箭科技（例如神奇昂贵的应用服务器、中间件）就能解决大多数可伸缩性方面的问题。

全文检索 Lucene、Sair

Spring框架

Spring Framework Overview

> 它定位的领域是许多其他流行的framework没有的
> Spring致力于提供一种方法管理你的业务对象
> Spring是全面的和模块化的
> Spring有分层的体系结构，可能选择仅仅使用Spring来简单化JDBC的使用，或用来管理所有的业务对象
> 它的设计从底部帮助编写易于测试的代码。Spring是用于测试驱动工程的理想的framework

DI

Dependency Injection 依赖注入是用于实现控制反转的最常见的方式之一

依赖注入

```
class C{
  J j = new J(10) ;
}
解耦方式：
 (1) 构造方式注入
class C{
  J j
  public c (J j) {
 this.j = j;
    };
}

 (2) 接口注入
public interface InjectFinder {
    void injectFinder(MovieFinder finder);
}

class MovieLister implements InjectFinder {
    ...
    public void injectFinder(MovieFinder finder) {
      this.finder = finder;
    }
    ...
}

 (3) 框架注入
本质是第三方依赖注入，但是这个第三方可以脱离类。将对象依赖映射信息存储在容器一般为.xml 或者特定的对象中，
```

```
并实现动态的注入。
get/set
```

为什么要有依赖注入（一种设计代码模式），因为我们要控制反转（设计代码的思路）。为什么控制反转。因为我们软件设计需要符合软件设计原则依赖倒置（设计代码原则），单一职责原则

MVP模式就是解耦比较全面的设计模式模型

Inversion of Control

控制反转是Spring容器的内核，AOP、声明式事务等功能在此基础上形成

```
get/set
<property name="id"   value="001"/>
<property name="name" value="002"/>

类型
<property name="school"  ref ="school"/>

构造器赋值
 <constructor-arg index="0">
        <value></value>
 </constructor-arg>

 构造器类
 <constructor-arg index="0">
   <bean class="类">
 </constructor-arg>
```

IOC是Spring中最核心的思想

AOP 贯穿在系统中重用的模块：切面

一组 Java Class：Jar包

Spring就是Jar包容器

内层区域有各种对象 单例模式对象

Context：理解成Windows注册表（上下文环境）

核心容器：可重用的类 DTO、DIO 数据访问对象（CURD增删改查）

核心类：上下文、表达语言

进行Maven配置过程

（1）下载Maven，修改setting、xml

（2）Eclipse里Maven install和User setting

（3）打开Maven视图，更新索引

（4）建Maven项目，添加依赖 Add indenpency

（5）run Maven install

注解@

```
@Autowired

@Autowired顾名思义，就是自动装配，其作用是为了消除代码Java代码里面的getter/setter与bean属性中的property。当然，getter看个人需求，如果私有属性需要对外提供的话，应当予以保留。

@Autowired默认按类型匹配的方式，在容器查找匹配的Bean，当有且仅有一个匹配的Bean时，Spring将其注入@Autowired标注的变量中。

//是一种最简单的，spring会自动扫描xxx路径下的注解。 必须要告诉Spring在此处要使用注解了
<context:component-scan base-package="xxx" />

<bean id="zoo" class="com.spring.model.Zoo" />
<bean id="tiger" class="com.spring.model.Tiger" />
<bean id="monkey" class="com.spring.model.Monkey" />

public class Zoo {
    @Autowired
    private Tiger tiger;

    @Autowired
    private Monkey monkey;

    public void test(){
    }
}
这里@Autowired注解的意思就是，当Spring发现@Autowired注解时，将自动在代码上下文中找到和其匹配（默认是类型匹配）的Bean，并自动注入到相应的地方去。
去掉了属性的getter/setter并使用@Autowired注解标注这两个属性 ： Spring会按照xml优先的原则去Zoo.java中寻找这两个属性的getter/setter
```

```
@Controller 声明Action组件
@Service    声明Service组件    @Service("myMovieLister")
@Repository 声明Dao组件
@Component   泛指组件，当不好归类时.
@RequestMapping("/menu")  请求映射
@Resource  用于注入，( j2ee提供的 ) 默认按名称装配，@Resource(name="beanName")
@Autowired 用于注入，(srping提供的) 默认按类型装配
@Transactional( rollbackFor={Exception.class}) 事务管理
@ResponseBody
@Scope("prototype")   设定bean的作用域
```

常见注解

包扫描

```
<context:component-scan base-package="entity"></context:component-scan>
```

可以大大减小代码量

```
public class SpringTest {

    @Test
    public  void testOne(){
        ApplicationContext context=new ClassPathXmlApplicationContext("applicationContext.xml");
        StudentInfo stu=(StudentInfo)context.getBean("info");
        System.out.println("");
    }
```

Autowriting

```
public class TextEditor {
    private SpellChecker spellChecker;
    private String name;
    public TextEditor( SpellChecker spellChecker, String name ) {
        this.spellChecker = spellChecker;
        this.name = name;
}
public SpellChecker getSpellChecker() {
    return spellChecker;
}
public String getName() {
    return name;
}
    public void spellCheck() {
        spellChecker.checkSpelling();
    }
}
```

```
public class SpellChecker {
    public SpellChecker(){
        System.out.println("Inside SpellChecker constructor." );
    }
    public void checkSpelling()
    {
        System.out.println("Inside checkSpelling." );
    }
}
```

```
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
        new ClassPathXmlApplicationContext("Beans.xml");
        TextEditor te = (TextEditor) context.getBean("textEditor");
        te.spellCheck();
    }
}
```

日志：事务切面 cross-cutting

cross concern 横切关注

Join Point 临接点

Advice 通知 方法前 方法后 方法前后 异常

Pointcut 切点 所有..方法才做..操作

Introdution 导入

Weaving 编制

依赖注入 自动装配 AOP

Navicat 可以管理 mysql、oracle 支持特殊字符，编码utf8ml4

Spring webMVC Framework

View Model Control

> 前最优秀的MVC框架，自从Spring 2.5版本发布后，由于支持注解配置，易用性有了大幅度的提高。Spring 3.0更加完善，实现了对Struts 2的超越

DispatcherServlet

DispatcherServlet是一个Servlet,所以可以配置多个DispatcherServlet

DispatcherServlet是前置控制器，配置在web.xml文件中的。拦截匹配的请求，Servlet拦截匹配规则要自已定义，把拦截下来的请求，依据某某规则分发到目标Controller(我们写的Action)来处理

```
<web-app>
    <servlet>
        <servlet-name>example</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>example</servlet-name>
        <url-pattern>*.form</url-pattern>
    </servlet-mapping>

</web-app>
```

```
<web-app>
    <servlet>
        <servlet-name>example</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>example</servlet-name>
        <url-pattern>*.form</url-pattern>
    </servlet-mapping>
</web-app>
// <url-pattern>*.form</url-pattern> 会拦截*.form结尾的请求
```

在DispatcherServlet的初始化过程中，框架会在web应用的 WEB-INF文件夹下寻找名为[servlet-name]-servlet.xml 的配置文件，生成文件中定义的bean。

当映射为@RequestMapping("/user/add")时，为例：

1、拦截*.do、*.htm， 例如：/user/add.do

这是最传统的方式，最简单也最实用。不会导致静态文件（jpg,js,css）被拦截。

2、拦截/，例如：/user/add

可以实现现在很流行的REST风格。

弊端：会导致静态文件（jpg,js,css）被拦截后不能正常显示。想实现REST风格，事情就是麻烦一些。后面有解决办法还算简单。