Spring创始人 rodJobhnson 毕业悉尼大学 博士学位 专业音乐

公司interface21后来改为 springsource

Spring是一个轻量级的框架

背景：为了解决传统的javaee的复杂性 复杂性：jsp+servlet+EJB(enterprise javabean)

作用：主要用于整合其他技术框架，将代码中javabean对象实现低耦合管理，提高系统的扩展性和维护性。

Spring分层

Core:核心ioc 工厂设计模式

Aop:面向切面编程 代理设计模式

Context：上下文对象

Jdbc:提供jdbc资源

Orm:hibernate、mybatis、spring-data

Web:整合struts1、struts2、

Spring核心 IOC和AOP

IOC:工作原理

将所有的javabean对象统一放到xml进行管理，在xml当中以bean标签形式展示，并且在xml形成依赖关系。容器启动的时候创建所有对象，并且形成依赖关系，必要的时候注入到程序中

IOC（inversion of control）控制反转

Di(dependency injection)依赖注入

从new对象，到通过控制反转帮我们创建对象。

优点：实现资源管理，实现资源可配置和易管理，降低了对象间的维护性。

缺点：底层大量使用了反射机制，降低效率。

反射：自身调用自身,运行时执行。使用就jdk class类可以获取实体类中的属性、方法等信息。

IOC底层采用了工厂设计模式+反射+解析xml

注入方式3种 1.setter 匹配getbean（id） getbean(.class) 2.构造方法 getbean(id) 3.方法或接口 getbean(.class)

Spring bean 生命周期 1.实例化2.初始化3.Spring管理(执行注入)4.容器关闭，执行销毁

Spring作用域(范围)

1. singleton 单例 默认单例

2. prototype 非单例

http当中使用

3.request

4.session

5.global session

工厂模式

1.静态工厂 类名.静态方法名

2.实例工厂 对象 别名 = new 对象（） 别名.方法名

AOP(aspect oriented programming)

面向切面编程 延续面向对象思想oop（object oriented programming）

Aop：

非核心业务逻辑：处理日志、认证权限、事务、异常。

核心业务逻辑：增删查改

Aop工作原理：aop分为非核心业务和核心业务，通过代理模式将非核心业务封装成aop组件。

反射：动态代理设计模式

1.jdk 依赖接口 newProxyInstance方法

2.cglb 实现MethodInterceptor

Aop 四种通知

之前通知 前置通知 before

之后通知 后置通知 after

环绕通知 around

异常通知 throws

事务：事务是访问并更新数据库各种数据项的一个单元执行程序

创建事务的方式一共2种

1. 声明式事务（在配置文件配置事务管理）

2. 编程式事务（通过代码来实现）

事务4个特点：

1.原子性：不可分割，要么全部成功，要么全部失败

2.一致性：多个查询，返回的结果必须保持一致

3.隔离性：事务与事务之间是透明

4.持久性：最后就是保存在到数据库中了

tx:advice aop通知表示要给哪些切入点方法加入事务

<tx:method name=""/>给切入点加入事务

timeout="-1"表示超时 超时时间按秒来计算，值是-1表示用不超时

read-only="false"默认false 设置true只读，一般用于查询语句中

rollback-for="java.lang.Exception"表示遇到什么异常回滚

no-rollback-for="java.lang.Exception"表示遇到什么异常不回滚

事务传播特性（一共7个）

1.required 默认

支持当前事务，如果事务存在，就使用当前事务，

如果事务不存在，就创建一个新的事务

2.supports

支持当前事务，如果事务存在，就使用当前事务，

如果事务不存在，就以非事务执行

3.mandatory

支持当前事务，如果事务存在，就使用当前事务，

如果事务不存在，就抛异常

4.requires\_new

支持当前事务，如果事务存在，挂起当前事务，创建一个新事务

5.not\_supported

以非事务执行，如果事务存在，挂起当前事务

6.never

以非事务执行，如果事务存在，就抛异常

7.nested

支持当前事务，在嵌套事务中执行事务

isolation属性表示隔离级别

脏读

事务A

1）查询

2）读取事务B未提交数据

事务B

1）修改数据未提交

不可重复读

事务A

1）查询 select \* from emp where 1=1 结果 1 2 3

2）查询 select \* from emp where 1=1 1 2 4

把事务B修改的数据查出

事务B

1）修改数据，已提交 update emp set id = 1 where 1=1

幻读

脏读 修改 未提交

幻读 新增 已提交

事务A

1）数据库10条记录

2）查询

3）11条 读取事务B

事务B

1）新增

2）已提交

从低到高

DEFAULT 使用什么数据库 脏读 不可重复读 幻读

1.READ\_UNCOMMITTED 否 否 否

未提交读

2.READ\_COMMITTED oracle 是 否 否

已提交读

3.REPEATABLE\_READ mysql 是 是 否

可重复读

4.SERIALIZABLE 是 是 是

可序列化

Jdbc工作流程

1. 加载驱动

2. 创建连接

3. 编译sql，执行sql

4. 返回结果集

5. 从后往前关闭

Jdbc优点

使用jdk提供的类java.sql，执行效率高，灵活性强，（手动写sql，方便优化sql语句），批量处理，连接不同的数据库。

Jdbc缺点：

代码冗余（重复代码多），

频繁创建连接和关闭资源，

使用不好，容易出现sql注入（or 1=1），

支持面向对象不好，返回结果集需要手动拼装。

Hibernate工作原理

1. 在hibernate.cfg.xml配置数据源信息

2. 获取SessionFactory

3. 获取Session

4. 开始事务

5. 持久化

6. 提交事务

7. 关闭资源

Hibernate优点

1. 支持面向对象

2. 快速开发

3. 移植性强（hql，qbc）

支持方言dialect

4. 支持一级缓存、二级缓存

Hibernate缺点

1. 底层采用的了大量的反射，

效率低（数据库、表）

2．支持批量不是很好，

3.hibernate要想使用存储过程，

通过getConnection()方法，回到最原始的jdbc方式来调用存储过程

@Transactional(propagation=Propagation.REQUIRED,

rollbackFor=Exception.class,

timeout=-1,

noRollbackFor=Exception.class,

readOnly=true)

hibernate和mybatis区别

orm映射 对象关系映射

相同点：

底层都使用了jdbc+反射

Mybatis底层采用了ibatis

Ibatis采用了jdbc

一级缓存和二级缓存

都有生成sql语句工具

不同点：

1.Hibernate全自动（映射了数据库和表）开发中无需关注sql语句和返回结果集，只需要关注业务逻辑即可，快速开发

Mybatis半自动（映射表中返回的结果集）需要手动写sql语句，返回的结果集自动映射，需要使用resultType、resultMap，

2Hibernate效率低，映射数据库和表 底层采用了大量的反射机制。优化难度大点

Mybatis效率高，可以手动写sql语句，优化方便

3.Hibernate集成了日志（show\_sql、format\_sql）

Mybatis需要集成第三方日志插件log4j，要想打印sql语句，需要在log4j配置文件中设置debug模式

4.Hibernate懒加载策略（普通查询、子查询、关联查询）

Mybatis 设置懒加载只能设置全局性的，settings

<setting name=”lazylodingenabled” value=”true”/>

5.hibernate移植性好 使用hql qbc oid sql 反射 不需要使用sql 配置方言

mybatis移植性不好

手写sql语句（1.数据库迁移工具2.手改sql语句）

6.hibernate 的session一级缓存 hashtable

mybatis 的session一级缓存 hashmap

Springmvc是spring的分支（层级）

Spring和springmvc区别

Spring是springmvc父容器，

子容器可以继承父容器，反之

父容器不能使用子容器信息，

主要用于整合其他技术框架

将代码中javabean对象实现低耦合管理。提高系统的扩展性和维护性。Springmvc依赖spring的ioc容器，springmvc提供客户端（jsp）和服务端（springmvc）之间的通信，

处理http请求。

Springmvc工作原理

1. 客户端发送请求（login.do）,找到

DispatcherServlet并解析xml信息。

2. 找到处理映射器（<mvc:annotation-driven>），然后找到对应的handler（@RequestMapping("/login")）

3. 找到handler之后返回给DispatcherServlet，DispatcherServlet带着handler找对应的适配器（<mvc:annotation-driven>），也就是找后台方法（login），然后接收参数，并对参数进行转换处理，处理对应的业务逻辑。

4. 业务逻辑处理完之后，返回视图（ModelAndView），

将ModelAndView交给DispatcherServlet，DispatcherServlet拿到视图之后，

进行视图解析器解析（InternalResourceViewResolver），解析视图之后，转交给DispatcherServlet，DispatcherServlet拿到解析好的视图，交给view类，然后跳转真正视图，并返回给客户端。

Springmvc底层采用了restfull

，restfull一种风格，提供了一个原则和约束条件，建立在分布式系统上，客户端和服务端之间的通信，具有高延迟，高并发等

Restfull：

Post：insert（添加）

Get：select（查询）

Put：update（修改）

Delete：delete（删除）

http://www.tuicool.com/articles/67vyIbv jersey

springmvc和struts2区别

1. web.xml配置不一样，springmvc使用servlet元素

，struts2使filter元素

Filter是过滤器，初始化加载，

Servlet是服务器请求，初始化不加载，发送第一次请求时，初始化。

2. 接收请求参数，springmvc是局部变量，struts是成员变量

3. Springmvc不依赖拦截器，struts2依赖拦截器，

4. springmvc自己封装json，ObjectMapper，使用的注解

接收json参数注解@requestbody

响应json参数注解@responsebody，

Struts2集成了第三方json插件。

5. springmvc默认单例，也可以设置非单例@scope

struts2每次发送请求都会new一个新对象，struts2和spring整合也可以设置单例@scope

httpclient浏览器

http请求http协议

是一种跨工程 A工程调用B工程 B工程调用A工程

组成：http

参数:地址，和请求参数

创建浏览器：CloseableHttpClient client = HttpClients.createDefault();

定以发送的请求：

HttpGet get = new HttpGet(uri.build());

HttpPost post = new HttpPost(url);

get判断是否为空的参数

if(params != null){

Set<String> keys = params.keySet();

for(Iterator<String> i = keys.iterator();i.hasNext();){

String key = i.next();

String value = params.get(key) == null? "": params.get(key).toString();

uri.addParameter(key, value);

}

}

post定义传来的表单，定义请求参数

List<NameValuePair> list = new ArrayList<>();

if(params != null){

Set<String> keys = params.keySet();

for(Iterator<String> i = keys.iterator();i.hasNext();){

String key = i.next();

String value = params.get(key) == null

? ""

: params.get(key).toString();

list.add(new BasicNameValuePair(key,value));

}

}

设置参数编码

HttpEntity entity = new UrlEncodedFormEntity(list, Consts.UTF\_8);

将表单存放到 post body体中隐藏

post.setEntity(entity);

执行发送get或post请求地址 并响应信息

CloseableHttpResponse response = client.execute(get);

CloseableHttpResponse response = client.execute(post);

获取响应状态码 200为成功

int statusCode = response.getStatusLine().getStatusCode();

定义返回参数 默认null String result=null

判断状态码是否为200

if(statusCode == HttpStatus.SC\_OK){

result =

EntityUtils.toString(response.getEntity(), Consts.UTF\_8);

}else{

System.out.println("statusCode:" + statusCode);

}

关闭资源

resposne.close();

client.close();

返回结果；

web service:

是一种跨平台、跨语言、客户端和服务端之间通信的一种技术。

跨平台：a服务器（调用服务接口）可以访问b服务器（提供服务接口）

跨语言：java提供webservice服务，c语言调用java提供的webservice服务

Webservice组成部分：xml组成

http + schema（xsd）+ xml

http:soap(简单对象访问协议)，

soap提供约束条件（规则），使用soap约束条件来调用webservice方法

schema：xsd提供xml数据类型规范。

数据类型规范：

例如：

<xs:element minOccurs="0" name="arg0" type="xs:string"/>

Xml：wsdl提供一种文档规范

wsdl:types：表示定义方法参数、返回值

wsdl:message：表示通信信息

wsdl:portType：表示客户端调用接口入口点

wsdl:binding：表示服务通过binding找到protType入口点

wsdl:service ：暴露地址和实现类

Webservice提供的框架：

Xfire、cxf

Xfire升级成cxf

Cxf是document

Cxf定义服务端，方法参数不要使用map，返回值也不要定义map

Axis、axis2

Axis升级成axis2

Axis是rpc

cxf-core-3.1.11

cxf-rt-frontend-jaxws-3.1.11

cxf-rt-transports-http-3.1.11

Spring整合cxf

第一步：

http://localhost:8080/Spring-cxf-s/ws/user?wsdl

web工程：http://ip地址:tomcat端口号/工程名/ws/\*?wsdl

第二步：

执行web.xml

<!-- cxf start -->

<servlet>

<servlet-name>cxf</servlet-name>

<servlet-class>

org.apache.cxf.transport.servlet.CXFServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>cxf</servlet-name>

<url-pattern>/ws/\*</url-pattern>

</servlet-mapping>

<!-- cxf end -->

第三步：

<import resource="classpath:META-INF/cxf/cxf.xml"/>

<import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

这两句话解析访问url地址

http://localhost:8080/Spring-cxf-s/ws/user?wsdl

第四步：

解析地址之后，找到哪个服务

<!-- jdk发布服务 -->

<jaxws:endpoint

implementor="#userServiceImpl"

address="/user" />

假设工程只是客户端，没有服务端

Web.xml需要配置

<!-- cxf start -->

<servlet>

<servlet-name>cxf</servlet-name>

<servlet-class>

org.apache.cxf.transport.servlet.CXFServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>cxf</servlet-name>

<url-pattern>/ws/\*</url-pattern>

</servlet-mapping>

<!-- cxf end -->

Spring配置文件中applicationContext也不要配置

<import resource="classpath:META-INF/cxf/cxf.xml"/>

<import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

上传jsp页面设置enctype="multipart/form-data"

xml配置上传解析器

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

修改上传编码 默认编码是ISO-8859-1

<property name="defaultEncoding" value="UTF-8"></property>

设置文件大小 单位是b(字节)

<proeprty name="maxUploadSize" value="参数">

下载：

拦截器xml配置

<mvc:interceptors>

<mvc:interceptor>

需要拦截的请求地址可以使用通配符

<mvc:mapping path="/\*\*"/>

不需要拦截的请求地址直接不走拦截器

<mvc:exclude-mapping path="/js/\*\*"/>

拦截器

1）<bean class="com.etoak.common.AuthInterceptor"/>

需要@Component实例化对象注入

2）<ref bean="authInterceptor"/>

</mvc:interceptor>

</mvc:interceptors>

实现HandlerInterceptor

1.发送请求进入后台之前先触发preHandle方法

\* false：拦截请求 true：不拦截请求

2.跳转视图是之前触发postHandle方法

3.跳转视图成功之后触发afterCompletion方法

Redis

底层 c语言 单线程

Redis默认端口号：6379

Redis 优点：快速查询

缺点：缺少结构化

为什么要用缓存

替数据库分担一部分压力

缓存使用在数据库之前

If（缓存存在数据）{

取缓存数据

}else{

查询数据，查询返回的结果集

再放入缓存中

}

Redis是nosql（not only sql）

非关系型数据库 key-value

提供了5种数据类型

默认 key

string：存储最大长度512M

hset：hashmap

list：链表 效率低

set：无序

sorted set：有序

redis特点：

持久化：默认持久化rdb、aof

支持事务、不支持事务回滚

一共16个数据库 默认0

单线程、原子性

redis.io 英文

www.redis.cn 中文

任务调度（定时任务、定时器）

运行时在指定的时候执行业务逻辑。

Maven

M2\_HOME

D:\server\apache-maven-3.3.9

%M2\_HOME%\bin

maven：基于多模块开发，管理jar包

在pom.xml配置jar包，第一次需要网络下载，

远程在apache服务器下载

（默认服务器apache，服务器在国外）

阿里提供了国内服务器

mirrors镜像

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>

http://maven.aliyun.com/nexus/content/groups/public/

</url>

<mirrorOf>central</mirrorOf>

</mirror>

Maven工程

Jdk默认1.5

Group Id 当前工程存储本地仓库的结构 com.etoak

Artifact Id 工程名 et

com.etoak.et

version 版本号

packaging

pom：继承聚合

jar ：普通的java工程 java project

war ：web工程 web project

默认是jar

Maven目录结构

src/main/java java代码

src/main/resources 资源配置文件（例如 applicationContext.xml）

src/test/java junit代码（单元测试）

src/test/resouces junit资源配置文件

src/main/webapp 存在web.xml 、jsp、css、js等文件

继承聚合

主工程（pom）聚合子工程

子工程继承主工程

dubbo:阿里巴巴公司提供的一个高性能分布式服务框架，通过rpc进行输入和输出功能,以及治理soap方案和spring进行整合

分布式：一个项目工程分多个模块，对不同的模块进行拆分

将拆分的模块部署到不同的服务器上，提高性能

解决并发、高延迟 解决单一性

rpc：远程过程调度协议 远程访问计算机

通信 sokect nio

spring boot:

1. 创建独立的Spring应用程序

白话：每个独立的工程都可以独立运行

2. 嵌入的Tomcat，无需部署WAR文件

白话：springboot嵌入tomcat

不管是不是war工程都可以独立去运行

3. 简化Maven配置

白话：微服务（v服务） 每个模块独立运行

4. 自动配置Spring

白话：springboot集成大多数框架 mybatis springmvc等

result风格 提高高并发 高延迟

5. 提供生产就绪型功能，如指标，健康检查和外部配置

6. 绝对没有代码生成和对XML没有要求配置

白话：不是无配置 只是注解化 不建议去使用xml配置 使用注解简化配置