1. 选择题

1.单选(2分)如下关于AOP的描述中错误的是:**（B）**  
A. AOP可以对代码进行解耦;  
  
B. AOP只能作用于public函数;  
  
C. AOP可以作用于函数执行;  
  
D. AOP可以简化代码;  
  
2.单选(2分)下面哪个不属于AOP的Advice类型:**（C）**  
A. Before;   
  
B. After returning;  
  
C. Before returning;  
  
D. Around;  
  
3.多选(3分)关于Pointcut表达式，说法错误的是:**（A C D）**  
A. 必须得声明函数抛出异常的匹配模式;  
  
B. 必须得声明函数的参数列表匹配模式;  
  
C. 必须得声明函数所在类的包的匹配模式;  
  
D. 必须得声明函数的modifiers;

1.单选(2分)下列关于loC容器的描述错误的是:**（C）**

A. 容器会根据配置创建对象;

B. 应用可以从容器中获取所配置的对象;

C. 容器必须由应用代码主动创建;

D. 根据会根据依赖关系组装对象;  
  
2.单选(2分)如下关于自动装配(Autowiring)说法中，错误的是:**（A）**

A. 不可以根据Bean的类型通过构建函数注入;

B. 可以根据Bean的类型通过Setter方法注入;

C. 可以根据Bean的类型自动注入;

D. 可以根据Bean名称自动注入;  
  
3.单选(2分)关于依赖注入方式，说法错误的是:**（C）**

A. 可以通过Setter方法注入;

B. 可选依赖可以通过构造函数注入;

C. 强依赖必须通过构造函数注入;

D. 可以通过构造函数注入;  
  
  
4.多选(3分)关于Bean作用域的singleton和prototype说法正确的是:**（A C）**  
  
A. singleton只在容器中创建一个对象;   
  
B. singleton在每次通过容器获取实例时创建一个对象;  
  
C. prototype在每次通过容器获取实例时创建一个对象;  
  
D. prototype只在容器中创建一个对象;  
  
5.多选(3分)关于Bean的生命周期回调，说明正确的是:**（B D）**  
  
A. 可以通过销毁时回调进行资源回收的操作  
  
B. 可以通过创建(初始化时)回调进行申请资源的操作;  
  
C. 生命周期回调发生在编译时;  
  
D. 销毁回调发生在对象回收前

1. 判断题

6.判断(2分)loC容器在初始化时，必须要加载名称为application-context.xmI

的配置文件;**（A）**  
  
A. ✖

1. ✔

三、简答题

1.我们在示例中通过context.getBean(Performance.class);方式来获取容器管理的对象，查找并学习Spring的API文档，列出getBean函数的其它形式接口，并解释各个接口的差异。

2. 假设我们有个应用（Application），它包含一个写文件的服务（FileWriterService），该服务在创建时会创建并打开一个文件，销毁时关闭该文件，并且该服务包含一个write(String content)方法：会将content写入到文件中，要求：

（1） 使用Spring IoC的相关内容组织该程序；

（2） 文件路径通过配置文件（properties）指定；

（如果不清楚或者忘记如何打开，关闭以及写入文件，请回顾一个翁老师的 Java基础 课程）

基本要求：必须附加一个项目说明文档，说明每个功能点对应程序的运行结果（截图），项目的接口说明或者关键代码（不要把全部代码贴出来）等可以反映项目结果的内容。

3. 完成一个基本的应用，提供若干个Service（>=2），每个Service提供基本的增删查改的接口（实现随意，比如输出一行信息），通过AOP保证所有所有的Service接口在正常调用返回后以及抛出异常时（Service接口模拟），打出如下信息：函数名称，函数参数，并说明发生的事件：正常返回或者抛出异常。

（注：需要查找资料，了解JoinPoint的API）

基本要求：必须附加一个项目说明文档，说明每个功能点对应程序的运行结果（截图），项目的接口说明或者关键代码（不要把全部代码贴出来）等可以反映项目结果的内容。

1. 配置Spring配置文件，加入AOP相关的xsd文件。并配置AOP。

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p" xmlns:aop="http://www.springframework.org/schema/aop"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-2.0.xsd">

<!-- 引入AOP -->

<aop:aspectj-autoproxy />

<!-- 自动扫描annotation配置 -->

<context:component-scan base-package="site.zhanjingbo" />

<!-- 自定义的日志切面声明 -->

<bean id="logAspect" class="site.zhanjingbo.aspect.LogAspect"></bean>

</beans>

1. 编写Service。以UserService为例。

package site.zhanjingbo.service;

import org.springframework.stereotype.Component;

@Component

public class UserService {

public void add(String username) throws Exception {

System.out.println("添加用户:" + username);

throw new Exception("测试");

}

public void edit(String username) {

System.out.println("添加用户:" + username);

}

public void del(String username) {

System.out.println("删除用户:" + username);

}

public void find(String username) {

System.out.println("查询用户:" + username);

}

}

1. 编写Aspect，完成切面方法。

package site.zhanjingbo.aspect;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Aspect;

@Aspect

public class LogAspect {

@AfterReturning("execution(\* site.zhanjingbo.service.\*.\*(..))")

public void logAfterReturning(JoinPoint jp){

String className = jp.getSignature().getDeclaringTypeName();

String methodName = jp.getSignature().getName();

System.out.print("正常返回Log:调用"+className+"的"+methodName+"方法,参数为：");

for (Object obj : jp.getArgs()) {

System.out.print(obj.toString());

}

System.out.println("");

}

@AfterThrowing(pointcut="execution(\* site.zhanjingbo.service.\*.\*(..))",throwing="ex")

public void logAfterThrowing(JoinPoint jp,Exception ex){

String className = jp.getSignature().getDeclaringTypeName();

String methodName = jp.getSignature().getName();

System.out.print("异常Log:调用"+className+"的"+methodName+"方法,参数为：");

for (Object obj : jp.getArgs()) {

System.out.print(obj.toString());

}

System.out.println("。抛出异常:"+ex.getMessage()+"!");

}

}

1. 编写模拟调用的Application类，来模拟业务。

public class Application {

public static void main(String[] args){

//加载Spring相关配置文件

ApplicationContext applicationContext = new ClassPathXmlApplicationContext("application-context.xml");

//获取UserService实例

UserService userService = applicationContext.getBean(UserService.class);

//调用方法

try {

userService.add("ZhangSan");

} catch (Exception e) {

}

//获取ProductService实例

ProductService productService = applicationContext.getBean(ProductService.class);

//调用方法

productService.del("Pen");

((ConfigurableApplicationContext)applicationContext).close();

}

}