**一、选择(单选和多选，1分/个)**

1. spring框架采用Java的**（ ）**机制进行依赖注入。
   1. 映射
   2. 反转
   3. 反射
   4. 控制
2. spring中Bean的默认作用域为**（ ）**。
   1. prototype
   2. singleton
   3. properties
   4. application
3. SpringMVC通过**（ ）**来对请求进行统一分发处理，从而转发给相应的业务控制类。
   1. DispatcherServlet
   2. initServlert
   3. ConstructerServlet
   4. DisplayServlet
4. 在Spring配置文件中为JDBC配置事务时，需要做四步工作，它们的正确顺序是**（ ）**
   1. **配置数据源，配置事务管理器，配置事务增强切面，配置事务增强**
   2. **配置数据源，配置事务管理器，配置事务增强，配置事务增强切面**
   3. **配置事务管理器，配置数据源，配置事务增强，配置事务增强切面**
   4. **配置事务管理器，配置事务增强，配置事务增强切面，配置数据源**
5. SpringMVC中若要让控制器类被扫描到，需要在配置文件中声明**（ ）**
   1. spring-package
   2. spring-context
   3. spring-component
   4. context-componentscan
6. 在SpringMVC中，下述方法可以用于接收请求参数name值的是**（ ）**
   1. request(@Param String name)
   2. regist(@RequestParam name)
   3. regist(@RequestParam(“name”) String name)
   4. regist(@RequestParam String name)
7. MyBatis通过映射文件进行输入参数映射时，输入参数的类型可以是**（ ）**(多选)
   1. 基本数据类型
   2. Pojo类型
   3. List类型
   4. Map类型
8. Spring的主要功能有：**（ ）**(多选)
   1. 依赖注入
   2. 控制反转
   3. 面向切面
   4. 整合ORM框架
   5. 独立的MVC框架
   6. 可作为web服务器
   7. 可独立出来开发
9. 以下注解是Spring用来实例化所需要Bean的是（ ）

A、@Value

B、@Prama

C、@Componet

D、@Controller

1. Spring AOP的通知类型包括：**（ ）**(多选)
   1. 超级通知
   2. 环绕通知
   3. 异常通知
   4. 后置通知
   5. 最终通知
2. 下列AspectJ注解用于定义切入点的是 （ ）

A、@PointCut

B、@poIntCut

C、@POINTCUT

D、@CutPoint

1. 下列在编程中不需要“池化”的对象是**（ ）**
   1. 数据源
   2. Pojo对象
   3. 服务器线程
   4. 通信线程
2. 对于Bean的作用域：A：singleton；B：prototype；C:request；D：session；按其生命期长度由小到大来排列顺序为（ ）

A、**prototype<request<session<singleton**

B、**request < session < prototype <singleton**

C、**request < prototype <session<singleton**

D、**prototype< session < request <singleton**

1. SpringMVC中使用占位符的映射( )

A、@Request

B、@Response

C、@Mapping

D、@RequestMapping

1. （“/text/${xxx}/xxx”），可以通过 (“xxx”)来得到占位符的信息。

A、@Value

B、@Prama

C、@pathValue

D、@pathVariable

1. Spring AOP的通知类型不包括**( )**
   1. 前置通知
   2. 后置通知
   3. 超级通知
   4. 最终通知
   5. 异常通知
2. 给定切点表达式 execution(\* com… \*. \*Service. \*(…))，下列说法正确的是（ ）
   1. 参数列表应使用..
   2. 返回参数不能用\*
   3. 省略包使用3个.
   4. 不需要的可以不写
3. 一个Bean被IoC容器创建过程中，调用到的方法种类有( ) (多选)
   1. set
   2. constructer
   3. get
   4. toString
4. 以下资源哪些是在Spring和MyBatis整合时需要配置在applicationContext.xml中的（ ）。(多选)
   1. dataSource
   2. SqlSessionFactory
   3. 事务transaction
   4. mapperScanner
5. MyBatis和Spring框架整合时所需要加入下列哪些**JAR包MyBatis框jar、Spring框架的jar、数据库驱动jar、整合的中间jar**
   1. MyBatis框jar
   2. Spring框jar
   3. 数据库驱动jar
   4. ORM插件jar
   5. Springmvc框jar

**二、简答题（3分1个）**

**Spring注解中@Autowired和@Resource的区别**

**请简述AOP的原理？AOP有何应用价值**  
**原理**：

**springMVC可以实现的功能.**

**Spring框架中，什么是控制反转？什么是依赖注入？有何区别？**

**事务的四个特性和每种特性的说明**

**SpringMVC工作原理**

**简述Spring Bean的实例化有哪几种常见的方法及过程**  
**方法**：四种。

**在MyBatis开发中，通常一个Xml映射文件，都会写一个Dao接口与之对应，请问这个Dao接口的工作原理是什么？Dao接口里的方法，参数不同时，方法能重载吗？**

**Spring控制器接受请求参数的常见方式有哪几种。**

**请简述MyBatis框架的优势。**

**请简述一下MyBatis的缓存分类和实现方法。**

**在MyBatis框架中，当实体类中的属性名和表中的字段名不一样，怎么办。**

**三、编程**

1. 在SpringMVC框架中，通过AJAX请求<http://localhost:8080/springmvcdemo/getUserList.action>向服务器发送查询用户列表请求，服务器端控制器返回包含所有用户对象User的集合对象userList，请根据此要求在横线处完善下述程序。(5分)

@RequestMapping(" / \_\_\_\_\_\_\_\_\_\_")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public List< User> getUserList(){

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

for (long i = 0; i <=10; i++) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

user.setId(i);

user.setName("ssm"+i);

user.setBirthday(new Date());

\_\_\_\_\_\_\_\_\_\_\_;

}

\_\_\_\_\_\_\_\_\_\_\_\_;

}

1. 使用mybatis框架完善以下基于user表的CRUD操作。user表所对应的实体类如下：（14分）

public class User {

private int id;

private String name;

private int age;

public User(String name,int age){

this.name=name;

this.age=age;

}

}

（1）请完善sql映射文件"com.qst.mapper.UserMapper"的实现：

<insert id="insertUser" parameterType=" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_">

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

</insert>

<delete id="deleteUser" parameterType=" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_">

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

</delete>

<update id="updateUser" parameterType=" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ">

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

</update>

<select id="selectUserById" parameterType="int" resultType=" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ">

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

</select>

<select id="selectAllUsers" resultType="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_">

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

</select>

(2) 在核心配置文件中配置sql映射文件:

<configuration>

<mapper resource="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"/>

</configuration>

(3)完善以下测试类中对user表的CRUD的操作:

public void insertUser(int id) {

SqlSession session = sessionFactory.openSession();

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

1. **编写业务类UserService，包括addUser()、deleteUser()、updateUser()、queryUser()四个方法，使用Spring配置文件为UserService类配置事务管理；要求addUser()、deleteUser、updateUser()必须具有事务；queryUser()方法可以有事务，也可以没有。addUser()和updateUser()的隔离级别设置为可重复读Isolation.REPEATABLE\_READ，其它方法使用默认隔离级别。（25分）**