# 第一章

## 【答案】

1、Spring框架的优点如下：

（1）非侵入式设计：Spring是一种非侵入式（non-invasive）框架，它可以使应用程序代码对框架的依赖最小化。

（2）方便解耦、简化开发：Spring就是一个大工厂，可以将所有对象的创建和依赖关系的维护工作都交给Spring容器管理，大大的降低了组件之间的耦合性。

（3）支持AOP：Spring提供了对AOP的支持，它允许将一些通用任务，如安全、事务、日志等进行集中式处理，从而提高了程序的复用性。

（4）支持声明式事务处理：只需要通过配置就可以完成对事务的管理，而无需手动编程。

（5）方便程序的测试：Spring提供了对Junit4的支持，可以通过注解方便的测试Spring程序。

（6）方便集成各种优秀框架：Spring不排斥各种优秀的开源框架，其内部提供了对各种优秀框架（如：Struts、Hibernate、MyBatis、Quartz等）的直接支持。

（7）降低了Java EE API的使用难度：Spring对Java EE开发中非常难用的一些API（如：JDBC、JavaMail等），都提供了封装，使这些API应用难度大大降低。。

2、IoC的全称是Inversion of Control，中文名称为控制反转。控制反转就是指在使用Spring框架之后，对象的实例不再由调用者来创建，而是由Spring容器来创建，Spring容器会负责控制程序之间的关系，而不是由调用者的程序代码直接控制。这样，控制权由应用代码转移到了Spring容器，控制权发生了反转。

DI的全称是Dependency Injection，中文称之为依赖注入。它与控制反转（IoC）的含义相同，只不过这两个称呼是从两个角度描述的同一个概念。从Spring容器的角度来看，Spring容器负责将被依赖对象赋值给调用者的成员变量，这相当于为调用者注入了它依赖的实例，这就是Spring的依赖注入。

# 第二章

## 【答案】

1、Bean的生命周期的整个执行过程描述如下。

（1）根据配置情况调用Bean构造方法或工厂方法实例化Bean。

（2）利用依赖注入来完成Bean中所有属性值的配置注入。

（3）如果Bean实现了BeanNameAware接口，则Spring调用Bean的setBeanName()方法传入当前Bean的id值。

（4）如果Bean实现了BeanFactoryAware接口，则Spring调用setBeanFactory()方法传入当前工厂实例的引用。

（5）如果Bean实现了ApplicationContextAware接口，则Spring调用setApplicationContext()方法传入当前ApplicationContext实例的引用。

（6）如果BeanPostProcessor和Bean关联，则Spring将调用该接口的预初始化方法postProcessBeforeInitialzation()对Bean进行加工操作，这个非常重要，Spring的AOP就是用它实现的。

（7）如果Bean实现了InitializingBean接口，则Spring将调用afterPropertiesSet()方法。

（8）如果在配置文件中通过init-method属性指定了初始化方法，则调用该初始化方法。

（9）如果有BeanPsostProcessor和Bean关联，则Spring将调用该接口的初始化方法postProcessAfterInitialization()。此时，Bean已经可以被应用系统使用了。

（10）如果在<bean> 中指定了该Bean的作用范围为 scope="singleton"，则将该Bean 放入Spring IoC的缓存池中，将触发Spring对该Bean的生命周期管理；如果在<bean>中指定了该Bean的作用范围为scope="prototype"，则将该Bean交给调用者，调用者管理该Bean的生命周期，Spring不再管理该Bean。

（11）如果Bean实现了DisposableBean接口，则Spring会调用destory()方法将Spring中的Bean销毁；如果在配置文件中通过destory-method属性指定了Bean的销毁方法，则Spring将调用该方法进行销毁。

2、Bean中主要包含三种装配方式，分别为基于XML的装配，基于Annotation的装配和自动装配，这三种装配方式的用法如下：

（1）基于XML的装配：Spring提供了2种基于XML的装配方式：设值注入（Setter Injection）和构造注入（Constructor Injection）。设置注入中的Bean类必须提供一个默认的无参构造方法，同时必须为需要注入的属性提供对应的setter方法。使用设值注入时，在Spring配置文件中，需要使用<bean>元素的子元素<property>来为每个属性注入值。使用构造注入时，在配置文件里，需要使用<bean>元素的子元素<constructor-arg>来定义构造方法的参数，可以使用其value属性（或子元素）来设置该参数的值。

（2）基于Annotation的装配：使用基于Annotation的装配时，首先需要使用@Repository、@Service与@Constroller分别对实现类进行标注，然后用@Autowired或@Resource注解对注入的Bean的属性进行标注，最后在Spring的配置文件中，通过<context:annotation-config />来开启注解处理器，或使用<context:component-scan base-package="Bean所在的包路径"/>的配置方式即可实现Annotation的装配。

（3）自动装配：在<bean>元素中使用autowire属性，并将其属性值设置为byName或者byType即可实现自动装配。

# 第三章

## 【答案】

1、AOP的专业术语包括Aspect、Joinpoint、Pointcut、Advice、Target Object、Proxy和Weaving，对于这些专业术语的解释，具体如下：

（1）Aspect（切面）：在实际应用中，切面通常是指封装的用于横向插入系统功能（如事务、日志等）的类，如图3-1中的Aspect。该类要被Spring容器识别为切面，需要在配置文件中通过<bean>元素指定。

（2）Joinpoint（连接点）：在程序执行过程中的某个阶段点，它实际上是对象的一个操作，例如方法的调用或异常的抛出。在Spring AOP中，连接点就是指方法的调用。

（3）Pointcut（切入点）：是指切面与程序流程的交叉点，即那些需要处理的连接点，如图3-2所示。通常在程序中，切入点指的是类或者方法名，如某个通知要应用到所有以add开头的方法中，那么所有满足这一规则的方法都是切入点。

（4）Advice（通知/增强处理）：AOP框架在特定的切入点执行的增强处理，即在定义好的切入点处所要执行的程序代码。可以将其理解为切面类中的方法，它是切面的具体实现。

（5）Target Object（目标对象）：是指所有被通知的对象，也被称为被增强对象。如果AOP框架采用的是动态的AOP实现，那么该对象就是一个被代理对象。

（6）Proxy（代理）：将通知应用到目标对象之后，被动态创建的对象。

（7）Weaving（织入）：将切面代码插入到目标对象上，从而生成代理对象的过程。

2、Spring中的通知按照在目标类方法的连接点位置，可以分为以下5种类型：

（1）org.aopalliance.intercept.MethodInterceptor（环绕通知）

在目标方法执行前后实施增强，可以应用于日志、事务管理等功能。

（2）org.springframework.aop.MethodBeforeAdvice（前置通知）

在目标方法执行前实施增强，可以应用于权限管理等功能。

（3）org.springframework.aop.AfterReturningAdvice（后置通知）

在目标方法执行后实施增强，可以应用于关闭流、上传文件、删除临时文件等功能。

（4）org.springframework.aop.ThrowsAdvice（异常通知）

在方法抛出异常后实施增强，可以应用于处理异常记录日志等功能。

（5）org.springframework.aop.IntroductionInterceptor（引介通知）

在目标类中添加一些新的方法和属性，可以应用于修改老版本程序（增强类）

# 第四章

## 【答案】

1、在Spring的配置文件中配置JDBC时，需要定义了三个Bean，分别是dataSource、jdbcTemplate和需要注入类的Bean。在定义jdbcTemplate时，需要将dataSource注入到jdbcTemplate中，而其他需要使用jdbcTemplate的Bean，也需要将jdbcTemplate注入到该Bean中，这样配置完成后，Spring JDBC就可以使用了。

2、在JdbcTemplate类中，提供了大量的更新和查询数据库的方法，我们就是使用的这些方法来操作数据库的，其常用的方法包括execute()、update()和query()。其中execute()方法能够完成执行SQL语句的功能，update()方法可以完成插入、更新和删除数据的操作，query()方法可以用来处理各种对数据库表的查询操作。

# 第五章

## 【答案】

1、Spring中的事务管理分为两种方式：一种是传统的编程式事务管理，另一种是声明式事务管理。其中，编程式事务管理是通过编写代码实现的事务管理，包括定义事务的开始、正常执行后的事务提交和异常时的事务回滚。声明式事务管理是通过AOP技术实现的事务管理，其主要思想是将事务管理作为一个“切面”代码单独编写，然后通过AOP技术将事务管理的“切面”代码植入到业务目标类中。

声明式事务管理最大的优点在于开发者无需通过编程的方式来管理事务，只需在配置文件中进行相关的事务规则声明，就可以将事务规则应用到业务逻辑中。这使得开发人员可以更加专注于核心业务逻辑代码的编写，在一定程度上减少了工作量，提高了开发效率。

2、Spring的声明式事务管理通过Annotation（注解）方式来实现时，开发者只需做两件事情：

1）在Spring容器中注册事务注解驱动，其代码如下：

<tx:annotation-driven transaction-manager="transactionManager"/>

2）在需要使用事务的Spring Bean类或者Bean类的方法上添加注解@Transactional。如果将注解添加在Bean类上，则表示事务的设置对整个Bean类的所有方法都起作用；如果将注解添加在Bean类中的某个方法上，则表示事务的设置只对该方法有效。

# 第六章

## 【答案】

1、Hibernate和MyBatis。这两个框架的主要区别如下：

（1）Hibernate：是一个全表映射的框架。通常开发者只需定义好持久化对象到数据库表的映射关系，就可以通过Hibernate提供的方法完成持久层操作。开发者并不需要熟练的掌握SQL语句的编写，Hibernate会根据制定的存储逻辑，自动的生成对应的SQL，并调用JDBC接口来执行，所以其开发效率会高于MyBatis。然而Hibernate自身也存在着一些缺点，例如它在多表关联时，对SQL查询的支持较差；更新数据时，需要发送所有字段；不支持存储过程；不能通过优化SQL来优化性能等。这些问题导致其只适合在场景不太复杂且对性能要求不高的项目中使用。

（2）MyBatis：是一个半自动映射的框架。这里所谓的“半自动”是相对于Hibernate全表映射而言的，MyBatis需要手动匹配提供POJO、SQL和映射关系，而Hibernate只需提供POJO和映射关系即可。与Hibernate相比，虽然使用MyBatis手动编写SQL要比使用Hibernate的工作量大，但MyBatis可以配置动态SQL并优化SQL，可以通过配置决定SQL的映射规则，它还支持存储过程等。对于一些复杂的和需要优化性能的项目来说，显然使用MyBatis更加合适。

2、MyBatis框架的工作执行流程如下：

（1）读取MyBatis配置文件mybatis-config.xml。

（2）加载映射文件Mapper.xml。

（3）构建会话工厂。

（4）创建SqlSession对象。

（5）使用Executor接口来操作数据库。

（6）使用MappedStatement类型的参数对映射信息进行封装。

（7）输入参数映射。

（8）输出结果映射。

# 第七章

## 【答案】

1、SqlSessionFactory对象的实例可以通过SqlSessionFactoryBuilder对象来构建，而SqlSessionFactoryBuilder则可以通过XML配置文件或一个预先定义好的Configuration实例构建出SqlSessionFactory的实例，通过XML配置文件构建出的SqlSessionFactory实例，其实现代码如下：

// 读取配置文件

InputStream inputStream = Resources.getResourceAsStream("配置文件位置");

// 根据配置文件构建SqlSessionFactory

SqlSessionFactory sqlSessionFactory =

new SqlSessionFactoryBuilder().build(inputStream);

2、在映射文件中，<mapper>元素是映射文件的根元素，其他元素都是它的子元素。这些子元素及其作用如下所示：

<select>：用于映射查询语句，可自定义参数，返回结果集等。

<insert>：用于映射插入语句，执行后返回一个整数，代表插入的条数。

<update>：用于映射更新语句，执行后返回一个整数，代表更新的条数。

<delete>：用于映射删除语句，执行后返回一个整数，代表删除的条数。

<sql>：用于定义一部分SQL，然后可被其他语句引用此SQL。

<cache>：用于给定命名空间的缓存配置。

<cache-ref>：用于其他命名空间缓存配置的引用。

<resultMap>：用于描述如何从数据库结果集中来加载对象。

# 第八章

## 【答案】

1、MyBatis动态SQL中的主要元素及其作用如下：

（1）<if>元素：用于判断语句，用于单条件分支判断。

（2）<choose>（<when>、<otherwise>）元素：相当于Java中的switch...case...default语句，用于多条件分支判断。

（3）<where>、<trim>、<set>元素：辅助元素，用于处理一些SQL拼装、特殊字符问题。

（4）<foreach>元素：循环语句，常用于in语句等列举条件中。

（5）<bind>元素： 从OGNL表达式中创建一个变量，并将其绑定到上下文，常用于模糊查询的sql中。

2、在使用<foreach>时，collection属性是必须指定的，而且在不同情况下，该属性的值是不一样的。主要有以下3种情况：

（1）如果传入的是单参数且参数类型是一个数组或者List的时候，collection属性值分别为array和list（或collection）；

（2）如果传入的参数是多个的时候，就需要把它们封装成一个Map了，当然单参数也可以封装成Map集合，这时候collection属性值就为Map的键。

（3）如果传入的参数是POJO包装类的时候，collection属性值就为该包装类中需要进行遍历的数组或集合的属性名。

所以在设置collection属性值的时候，必须按照实际情况配置，否则程序就会出现异常。。

# 第九章

## 【答案】

1、在java中，对象之间存在着三种关联关系，分别是一对一，一对多和多对多。在这三种关联关系中，一对一的关系就是在本类中定义对方类型的对象，如A类中定义B类类型的属性b，B类中定义A类类型的属性a；一对多的关系就是一个A类类型对应多个B类类型的情况，需要在A类中以集合的方式引入B类类型的对象，在B类中定义A类类型的属性a；多对多的关系就是在A类中定义B类类型的集合，在B类中定义A类类型的集合。

2、MyBatis在映射文件中加载关联关系对象主要通过两种方式：嵌套查询和嵌套结果。嵌套查询是指通过执行另外一条SQL映射语句来返回预期的复杂类型；嵌套结果是使用嵌套结果映射来处理重复的联合结果的子集。开发人员可以使用上述任意一种方式实现对关联关系的加载。

# 第十章

## 【答案】

1、MyBatis与Spring整合所需JAR包主要包括：Spring框架所需的JAR包、 MyBatis框架所需的JAR包、MyBatis与Spring整合的中间JAR、数据库驱动JAR包，以及数据源所需的JAR包。

2、MapperFactoryBean是MyBatis-Spring团队提供的一个用于根据Mapper接口生成Mapper对象的类，通过MapperFactoryBean可以配置接口文件以及注入SqlSessionfactory，从而完成一个Bean的实例化。MapperScannerConfigurer是MyBatis-Spring团队提供的一种用于以自动扫描形式来配置MyBatis中映射器的类，可以通过配置包路径来自动扫描包接口生成映射器，这使得开发人员可以在编写很少代码的情况下，完成对映射器的配置，从而提高开发效率。

# 第十一章

## 【答案】

1、Spring MVC主要有如下优点：

（1）是Spring框架的一部分，可以方便的利用Spring所提供的其他功能。

（2）灵活性强，易于与其他框架集成。

（3）提供了一个前端控制器DispatcherServlet，使开发人员无需额外开发控制器对象。

（4）可自动绑定用户输入，并能正确的转换数据类型。

（5）内置了常见的校验器，可以校验用户输入。如果校验不能通过，那么就会重定向到输入表单。

（6）支持国际化。可以根据用户区域显示多国语言。

（7）支持多种视图技术。它支持JSP、Velocity和FreeMarker等视图技术。

（8）使用基于XML的配置文件，在编辑后，不需要重新编译应用程序。

2、Spring MVC框架的工作流程如下：

（1）用户通过浏览器向服务器发送请求，请求会被Spring MVC的前端控制器DispatcherServlet所拦截。

（2）DispatcherServlet拦截到请求后，会调用HandlerMapping处理器映射器。

（3）处理器映射器根据请求URL找到具体的处理器，生成处理器对象及处理器拦截器（如果有则生成）一并返回给DispatcherServlet。

（4）DispatcherServlet会通过返回信息选择合适的HandlerAdapter（处理器适配器）。

（5）HandlerAdapter会调用并执行Handler（处理器），这里的处理器指的就是程序中编写的Controller类，也被称之为后端控制器。

（6）Controller执行完成后，会返回一个ModelAndView对象，该对象中会包含视图名或包含模型和视图名。

（7）HandlerAdapter将ModelAndView对象返回给DispatcherServlet。

（8）DispatcherServlet会根据ModelAndView对象选择一个合适的ViewReslover（视图解析器）。

（9）ViewReslover解析后，会向DispatcherServlet中返回具体的View（视图）。

（10）DispatcherServlet对View进行渲染（即将模型数据填充至视图中）。

（11）视图渲染结果会返回给客户端浏览器显示。

# 第十二章

## 【答案】

1、@Controller注解在使用时可分为如下两步：

（1）在相应的类上标注@Controller注解。

（2）在Spring MVC的配置文件中添加相应的扫描配置信息。首先需要在配置文件中引入context名称空间的配置，然后使用<context:component-scan>元素指定需要扫描的类包。

2、请求处理方法的参数类型主要有：HttpServletRequest、HttpServletResponse、HttpSession、Model、HttpMethod等；

请求处理方法的返回类型主要有：ModelAndView、Map、String、void、HttpEntity<?>、View等。

# 第十三章

## 【答案】

1、@RequestParam注解主要用于对请求中的参数进行定义，多用于在请求参数名与方法形参名不一致时的参数间接绑定。在使用时可以指定他的4个属性，具体如下：

（1）value：name属性的别名，这里指参数的名字，即入参的请求参数名字，如果只使用vaule属性，则可以省略value属性名。

（2）name：用于指定请求头绑定的名称。

（3）required ：用于指定参数是否必须，默认是true，表示请求中一定要有相应的参数。

（4）defaultValue：默认值，表示如果请求中没有同名参数时的默认值。

2、在使用包装POJO类型数据绑定时，前端请求的参数名编写必须符合以下两种情况：

（1）如果查询条件参数是包装类的直接基本属性，则参数名直接用对应的属性名，如上面代码中的ordersId；

（2）如果查询条件参数是包装类中POJO的子属性，则参数名必须为“对象.属性”，其中“对象”要和包装POJO中的对象属性名称一致，“属性”要和包装POJO中的对象子属性一致。

# 第十四章

## 【答案】

1、JSON数据交互时，主要使用了两个注解@RequestBody和@ResponseBody。其中@RequestBody用于将请求体中的数据绑定到方法的形参中，在使用时标准在在方法的形参上。@ResponseBody用于直接返回return对象，在使用时标准在方法上。

2、静态资源访问的配置有3种方式，分别如下：

1）使用<mvc:resources… />元素用于配置静态资源的访问路径。

2）使用<mvc:default-servlet-handler>标签，可以访问所有静态资源。

3）激活Tomcat默认的Servlet来处理静态文件访问。

# 第十五章

## 【答案】

1、通常拦截器类可以通过两种方式来定义。一种是通过实现HandlerInterceptor接口，或继承HandlerInterceptor接口的实现类（如HandlerInterceptorAdapter）来定义；另一种是通过实现WebRequestInterceptor接口，或继承WebRequestInterceptor接口的实现类来定义。

2、单个拦截器在执行时，程序首先会执行拦截器类中的preHandle()方法，如果该方法的返回值为true，则程序会继续向下执行处理器中的方法，否则将不再向下执行；在业务处理器（即控制器Controller类）处理完请求后，会执行postHandle()方法，然后会通过DispatcherServlet向客户端返回响应；在DispatcherServlet处理完请求后，才会执行afterCompletion()方法。

当有多个拦截器同时工作时，它们的preHandle()方法会按照配置文件中拦截器的配置顺序执行，而它们的postHandle()方法和afterCompletion()方法则会按照配置顺序的反序执行。

# 第十六章

## 【答案】

1、文件上传表单需要满足的3个条件如下：

（1）form表单的method属性设置为post；

（2）form表单的enctype属性设置为multipart/form-data；

（3）提供<input type="file" name="filename" />的文件上传输入框。

2、解决中文文件名称下载时的乱码问题，需要从前端页面发送请求前先对中文名进行统一编码，然后在后台控制器类中对文件名称进行相应的转码。在具体实现时可分为如下两步：

（1）在下载页面中对中文文件名编码。可以使用Servlet API中提供的URLEncoder类中的encoder(String s, String enc)方法将中文转为UTF-8编码。

（2）在后台控制器类中对文件名根据不同的浏览器进行相应的转码。

# 第十七章

## 【答案】

1、由于Spring MVC是Spring框架中的一个模块，所以Spring MVC与Spring之间不存在整合的问题，只要引入相应JAR包就可以直接使用。因此SSM框架的整合就只涉及到了Spring与MyBatis的整合，以及Spring MVC与MyBatis的整合。Spring与MyBatis框架的整合时，通过Spring实例化Bean，然后调用实例对象中的查询方法来执行MyBatis映射文件中的SQL语句的，如果能够正确查询出数据库中的数据，那么就可以认为Spring与MyBatis框架整合成功。加入Spring MVC后，如果可以通过前台页面来执行查询方法，并且查询出的数据能够在页面中正确显示，那么就可以认为三大框架整合成功。

2、SSM框架整合时，Spring配置文件中配置信息主要包括：

（1）读取数据库常量配置文件db.properties的配置信息；

（2）数据源的配置信息；

（3）事务管理器的配置信息；

（4）开启事务注解的配置；

（5）MyBatis工厂SqlSessionFactory的配置信息；

（6）mapper文件扫描器的配置；

（7）扫描Service层的配置。

# 第十八章

## 【答案】

1、BOOT客户管理系统根据功能的不同，将项目结构主要划分为以下几个层次：

（1）持久对象层（也称持久层或持久化层）：该层由若干持久化类（实体类）组成。

（2）数据访问层（DAO层）：该层由若干DAO接口和MyBatis映射文件组成。接口的名称统一以Dao结尾，且MyBatis的映射文件名称要与接口的名称相同。

（3）业务逻辑层（Service层）：该层由若干Service接口和实现类组成。在本系统中，业务逻辑层的接口统一使用Service结尾，其实现类名称统一在接口名后加Impl。该层主要用于实现系统的业务逻辑。

（4）Web表现层：该层主要包括Spring MVC中的Controller类和JSP页面。Controller类主要负责拦截用户请求，并调用业务逻辑层中相应组件的业务逻辑方法来处理用户请求，然后将相应的结果返回给JSP页面。

2、引入SQL文件主要步骤如下：

（1）创建数据库；

（2）选择所创建的数据库；

（3）使用source命令导入数据库文件