第1章

【**思考题**】

1.请简述什么是MyBatis？

MyBatis是一个支持普通SQL查询、存储过程以及高级映射的持久层框架，它消除了几乎所有的JDBC代码和参数的手动设置以及对结果集的检索，使用简单的XML或注解进行配置和原始映射，将接口和Java的POJO映射成数据库中的记录，使得Java开发人员可以使用面向对象的编程思想来操作数据库。

2.请简要介绍MyBatis的工作原理。

（1）读取MyBatis配置文件mybatis-config.xml。

（2）由MyBatis配置文件加载映射文件Mapper.xml。

（3）通过MyBatis的环境等配置信息构建会话工厂SqlSessionFactory。

（4）由会话工厂创建SqlSession对象。

（5）MyBatis底层定义了一个Executor接口来操作数据库，它会根据SqlSession传递的参数动态的生成需要执行的SQL语句。

（6）在Executor接口的执行方法中，包含一个MappedStatement类型的参数，该参数是对映射信息的封装。

（7）在执行方法时，MappedStatement对象会对用户执行SQL语句的输入参数进行定义封装。

（8）在数据库中执行完SQL语句后，MappedStatement对象会对SQL执行输出的结果进行定义封装。

第2章

【**思考题**】

1.请简述<mappers>元素引入映射文件的4种方式。

（1）使用类路径引入。

（2）使用本地文件路径引入。

（3）使用接口类引入。

（4）使用包名引入。

2.请简述MyBatis映射文件中的常用元素及其作用。

<mapper>：映射文件的根元素，该元素只有一个namespace属性（命名空间），namespace属性作用如下。

（1）用于区分不同的mapper，全局唯一。

（2）绑定DAO接口，即面向接口编程。当namespace绑定某一接口之后，可以不用写该接口的实现类，MyBatis会通过接口的全限定名查找到对应的mapper配置来执行SQL语句，因此namespace的命名必须跟接口同名。

<cache>: 配置给定命名空间的缓存。

<cache-ref>：从其他命名空间引用缓存配置。

<resultMap>：描述数据库结果集和对象的对应关系。

<sql>：可以重用的SQL块，也可以被其他语句使用。

<insert>：用于映射插入语句。

<delete>：用于映射删除语句。

<update>：用于映射更新语句。

<select>：用于映射查询语句。

第3章

【**思考题**】

1.请简述MyBatis动态SQL中的常用元素及其作用。

<if>：判断语句，用于单条件判断。

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

<where>：简化SQL语句中where的条件判断。

<trim>：可以灵活地去除多余的关键字。

<set>：用于SQL语句的动态更新。

<foreach>：循环语句，常用于in语句等列举条件中。

2.请简述在使用<foreach>时，collection属性需要注意的几点。

（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。

第4章

【**思考题**】

1.请简述<collection>子元素的常用属性及其作用。

property：指定映射到的实体类对象属性，与表字段一一对应；

column：指定表中对应的字段；

ofType：它用于指定实体对象中集合类属性所包含的元素类型；

select：指定引入嵌套查询的子SQL语句，该属性用于关联映射中的嵌套查询；

fetchType：指定在关联查询时是否启用延迟加载。fetchType属性有lazy和eager两个属性值，默认值为lazy（即默认关联映射延迟加载）。

2.请简述MyBatis关联查询映射的两种处理方式。

MyBatis在映射文件中加载关联关系对象主要通过嵌套查询和嵌套结果两种方式。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。

第5章

【**思考题**】

1.请简述MyBatis的常用注解及其作用。

@Select：用于映射查询语句，其作用等同于xml配置文件中的<select>元素。

@Insert：用于映射插入语句，其作用等同于xml配置文件中的<insert>元素。

@Update：用于映射更新语句，其作用等同于xml配置文件中的<update>元素。

@Delete：用于映射删除语句，其作用等同于xml配置文件中的<delete>元素。@Param：用于指定SQL语句中的参数，通常用于SQL语句中参数比较多的情况。

@One：用于实现数据表的一对一关联查询，其作用等同于xml配置文件中的<assocation>元素。

@Many：用于实现数据表的一对多关联查询，@Many注解的作用等同于xml配置文件中的<collection>元素。

@Results：用于映射查询结果。

@Result：用于完成实体类中属性和数据表中字段的映射。

2.请简述@Result注解中的常用属性及其作用。

property：用于指定关联属性。

column：用于指定关联的数据库表中的字段。

one：用于一对一配置的关联查询。

many：用于一对多配置的关联查询。

## 第6章

【**思考题**】

1.请简述Spring框架的优点。

（1）非侵入式设计、降低耦合性，方便开发、支持AOP编程、支持声明式事务、方便程序的测试、方便集成各种优秀框架和降低Java EE API的使用难度。

（2）控制反转（Inversion of Control，缩写为IoC）是面向对象编程中的一个设计原则，用来降低程序代码之间的耦合度。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。

2.请简述控制反转的概念和依赖注入的概念。

依赖注入（Dependency Inject，缩写DI）就是由IoC容器在运行期间动态地将某种依赖资源注入对象之中。例如，将对象B注入（赋值）给对象A的成员变量。依赖注入的基本思想是：明确地定义组件接口，独立开发各个组件，然后根据组件的依赖关系组装运行。

## 第7章

【**思考题**】

1.请简述XML配置文件的根元素是<beans>中的常用元素及作用。

* id：id属性是<bean>元素的唯一标识符,Spring容器对Bean的配置和管理通过id属性完成,装配Bean时也需要根据id值获取对象。
* name：name属性可以为Bean指定多个名称,每个名称之间用逗号或分号隔开。
* class：class属性可以指定Bean的具体实现类,其属性值为对象所属类的全路径。
* scope：scope属性用于设定Bean实例的作用范围,其属性值有：singleton（单例）、prototype（原型）、request、session和global session。

2. 请简述Bean的几种装配方式的基本用法。

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8章

【**思考题**】

1.请列举Spring AOP的术语并解释。

（1）切面（Aspect）：切面是指关注点形成的类（关注点是指类中重复的代码），通常是指封装的、用于横向插入系统的功能类（如事务管理、日志记录等）。

（2）连接点（Joinpoint）：连接点是程序执行过程中某个特定的节点，在Spring AOP中，一个连接点通常是一个方法的执行。

（3）切入点（Pointcut）：当某个连接点满足预先指定的条件时，AOP就能够定位到这个连接点，在连接点处插入切面，该连接点也就变成了切入点。

（4）通知/增强处理（Advice）：通知/增强处理就是插入的切面程序代码。可以将通知/增强处理理解为切面中的方法，它是切面的具体实现。

（5）目标对象（Target）：目标对象是指被插入切面的方法。

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

（7）代理（Proxy）：将通知应用到目标对象之后，程序动态创建的通知对象，就称为代理。

（8）引介（Introduction）：引介是一种特殊的通知，它为目标对象添加一些属性和方法。这样，即使一个业务类原本没有实现某一个接口，通过AOP的引介功能，也可以动态地为该业务类添加接口的实现逻辑，让业务类成为这个接口的实现类。

2.请列举AOP实现中Spring提供的注解并解释其作用。

@Aspect：配置切面，@Pointcut：配置切点，@Before：配置前置通知，@After：配置后置通知，@Around：配置环绕方式，@AfterReturning：配置返回通知，@AfterThrowing：配置异常通知.

## 第9章

【**思考题**】

1.请简述抽象类JdbcAccessor提供的一些访问数据库时使用的公共属性。

（1）DataSource：DataSource主要功能是获取数据库连接。在具体的数据操作中，DataSource还可以提供对数据库连接的缓冲池和分布式事务的支持。

（2）SQLExceptionTranslator：SQLExceptionTranslator是一个接口，全称为 org.springframework.jdbc.support.SQLExceptionTranslator。SQLExceptionTranslator接口负责对SQLException异常进行转译工作。通过必要的设置或者调用SQLExceptionTranslator接口中的方法，JdbcTemplate可以将SQLException的转译工作委托给SQLExceptionTranslator的实现类来完成。

2.请简述Spring JDBC是如何进行配置的。

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。

## 第10章

【**思考题**】

1.请简述Spring MVC框架的优点。

（1）Spring MVC是Spring框架的后续产品，可以方便地使用Spring框架所提供的其他功能。

（2）Spring MVC使用简单，很容易设计出干净简洁的Web层。

（3）Spring MVC支持各种请求资源的映射策略。

（4）Spring MVC具有非常灵活的数据验证、格式化和数据绑定机制，能使用任何对象进行数据绑定，不必实现特定框架的API。

（5）Spring MVC支持国际化，可以根据用户区域显示多国语言。

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

（7）Spring MVC灵活性强，易扩展。

2.请简述Spring MVC框架的工作执行流程。

（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。

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

（4）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。

（5）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。

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

（7）前端控制器请求视图解析器根据逻辑视图名解析真正的视图。

（8）ViewResolver解析后，会向DispatcherServlet中返回具体的View（视图）。

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

（10）前端控制器向用户响应结果。

## 第11章

【**思考题**】

1.请简述Controller注解的使用步骤。

（1）将@Controller注解标注在普通Java类上。

（2）Spring MVC的配置文件中添加<context:component-scan>元素指定被Spring扫描的类包。

2.请列举@RequestMapping注解的属性（至少3个）。

（1）value属性，value属性是@RequestMapping注解的默认属性。当value属性是@RequestMapping注解显式使用的唯一属性时，可以省略value的属性名。

（2）method属性，method属性可以对处理器映射的URL请求方式进行限定。当请求的URL和处理器映射成功，但请求方式和method属性指定的属性值不匹配，处理器也不能正常处理请求。

（3）params属性，params属性中定义的值可以将请求映射的定位范围缩小。当客户端进行请求时，如果请求参数的值等于params属性定义的值，可以正常执行所映射到的方法，否则映射到的方法不执行。

## 第12章

【**思考题**】

1.请简述简单数据类型中的@RequestParam注解及其属性作用。

@RequestParam注解来定义参数的别名，完成请求参数名称和处理器的形参名称不一致时的数据绑定。@RequestParam注解的属性主要有以下几个。

* name属性指定请求头绑定的名称。
* value属性指name属性的别名，这里指参数的名称，即入参的请求参数名称，如value="name"表示请求的参数中，名称为name的参数的值将传入。
* required属性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。
* defaultValue属性用于指定形参的默认值，表示如果请求中没有同名参数时的默认值

2.请简述复杂POJO绑定时的注意事项。

在POJO类型数据绑定时，客户端请求的参数名称（本例中指form表单内各元素name的属性值）必须与要绑定的POJO类中的属性名称保持一致。这样客户端发送请求时，请求数据才会自动绑定到处理器形参POJO对象中，否则处理器参数接收的值为null。

3.请简述@RequestBody注解的作用。

@RequestBody注解结合Jackson提供的JSON格式转换器，即可完成JSON格式数据绑定到方法形参中。

4.请简述包装POJO类型绑定时的注意事项。

（1）在复杂POJO数据绑定时，如果数据需要绑定到POJO属性对象的属性中，客户端请求的参数名（本例中指form表单内各元素name的属性值）的格式必须为“属性对象名称.属性”，其中“属性对象名称”要和POJO的属性对象名称一致，“属性”要和属性对象所属类的属性一致。

（2）在复杂POJO数据绑定时，如果数据绑定到List类型的属性，客户端请求的参数名称编写必须符合以下要求。

* 如果List的泛型为简单类型，则客户端参数名称必须和POJO类中List属性所属类中的属性名称保持一致。
* 如果List的泛型参数为对象类型，则客户端参数名称必须与POJO类的层次结构名称保持一致，并使用数组格式描述对象在List中的位置，即客户端参数名称必须和最终绑定在List中的某个对象的某个属性的名称保持一致。

（3）在复杂POJO数据绑定时，如果数据绑定到Map类型的属性，客户端请求的参数名称（本例中指form表单内各元素name的属性值）必须与POJO类的层次结构名称保持一致，并使用键值的映射格式描述对象在Map中的位置，即客户端参数名称必须和要绑定的Map中的具体对象的具体属性的名称保持一致。

5.请简述Spring MVC方法返回值3中不同类型的区别。

当Spring MVC方法的返回值为void类型，方法执行后会跳转到默认的页面。默认页面的路径由方法映射路径和视图解析器中的前缀、后缀拼接成，拼接格式为“前缀+方法映射路径+后缀”。

当Spring MVC方法的返回值为String类型时，控制器方法执行后，Spring MVC会根据方法的返回值跳转到对应的资源。如果Spring MVC的配置文件中没有视图解析器，处理器执行后，会将请求转发到与方法返回值一致的映射路径。

当Spring MVC方法的返回值为ModelAndView类型时，ModelAndView对象包含视图相关内容和模型数据两部分，其中视图相关的内容可以设置逻辑视图的名称，也可以设置具体的View实例；模型数据则会在视图渲染过程中被合并到最终的视图输出。

6.请简述@ResponseBody注解注解的作用。

@ResponseBody注解的作用是将处理器返回的对象通过适当的转换器转换为指定的格式之后，写入到HttpServletResponse对象的body区，@ResponseBody注解通常用来返回JSON数据。

@ResponseBody注解可以标注在方法和类上，当标注在类上时，表示该类中的所有方法均应用@ResponseBody注解。

如果需要当前类中的所有方法均应用@ResponseBody注解，也可以使用@RestController注解，@RestController注解相当于@Controller+@ResponseBody两个注解的结合。

## 第13章

【**思考题**】

1.请简述Spring MVC统一处理异常的3种方式。

（1）使用简单异常处理器SimpleMappingExceptionResolver，SimpleMappingExceptionResolver实现了简单的异常处理，通过SimpleMappingExceptionResolver可以将不同类型的异常映射到不同的页面，当发生异常的时候，SimpleMappingExceptionResolver根据发生的异常类型跳转到指定的页面处理异常信息。SimpleMappingExceptionResolver也可以为所有的异常指定一个默认的异常处理页面，当应用程序抛出的异常没有对应的映射页面，则使用默认页面处理异常信息。

（2）使用自定义异常处理器， 除了可以使用SimpleMappingExceptionResolver进行异常处理，还可以自定义异常处理器统一处理异常。可以通过实现HandlerExceptionResolver接口，重写异常处理方法resolveException()来定义自定义异常处理器。当Handler执行并且抛出异常时，自定义异常处理器会拦截异常并执行重写的resolveException()方法，resolveException()方法返回值是ModelAndView类型的对象，可以在ModelAndView对象中存储异常信息，并跳转到异常处理页面。

（3）使用异常处理注解，从Spring 3.2开始，Spring 提供了一个新注解@ControllerAdvice。@ControllerAdvice注解结合方法型注解@ExceptionHandler，可以捕获Controller中抛出的指定类型的异常，从而实现不同类型的异常统一处理。

2.请简述单个拦截器和多个拦截器的执行流程。

（1）单个拦截器的执行流程

程序收到请求后，首先会执行拦截器中的preHandle()方法，如果preHandle()方法返回的值为false，则将中断后续所有代码的执行；

如果preHandle()方法的返回值为true，则程序会继续向下执行Handler的代码。当Handler执行过程中没有出现异常时，接着会执行拦截器中的postHandle()方法。postHandle()方法执行后会通过DispatcherServlet向客户端返回响应，并且在DispatcherServlet处理完请求后，执行拦截器中的afterCompletion()方法；如果Handler执行过程中出现异常，将跳过拦截器中的postHandle()方法，直接由DispatcherServlet渲染异常页面返回响应，最后执行拦截器中的afterCompletion()方法

（2）多个拦截器的执行流程

当有程序中配置了多个拦截器时，拦截器中的preHandle()方法会按照配置文件中拦截器的配置顺序执行，而拦截器中的postHandle()方法和afterCompletion()方法则会按照拦截器的配置顺序的相反顺序执行。

3.请简述上传表单需要满足的3个条件。

（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。

4.请简述如何解决中文文件名称下载时的乱码问题。

在使用Content-Disposition设置参数信息时，如果Content-Disposition中设置的文件名称出现中文字符，需要针对不同的浏览器设置不同的编码方式。目前Content-Disposition支持的编码方式有UrlEncode编码、Base64编码、RFC2231编码和ISO编码。

## 第14章

【**思考题**】

1.请简述SSM框架整合思路。

（1）搭建项目基础结构。首先需要在数据库中搭建项目对应的数据库环境；然后创建一个Maven Web项目，并引入案例所需的依赖；最后创建项目的实体类，创建三层架构对应的模块、类和接口。

（2）整合Spring和MyBatis。在Spring配置文件中配置数据源信息，并且将SqlSessionFactory对象和Mapper对象都交由Spring管理。

（3）整合Spring和Spring MVC。Spring MVC是Spring框架中的一个模块，所以Spring整合Spring MVC只需在项目启动时分别加载各自的配置即可。

2.请简述SSM框架整合时，Spring整合Mybatis的配置文件中的配置信息（无需写代码，只需简单描述所要配置的内容即可）。

（1）使用<context:property-placeholder>元素引入数据库连接信息的文件。

（2）使用<bean>元素定义数据源。

（3）使用<bean>元素创建SqlSessionFactory对象。

（4）使用<bean>元素扫描Dao包，创建动态代理对象

## 第15章

【**思考题**】

1.请简述系统中各个层次的组成和作用。

（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。DAO接口的名称统一以Mapper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。业务逻辑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，然后将处理结果返回给JSP页面。

2.请简述引入SQL文件的过程。

（1）登录数据库。

（2）创建数据库。

（3）选择所创建的数据库。

（4）导入数据库文件。