SpringMVC、Springboot

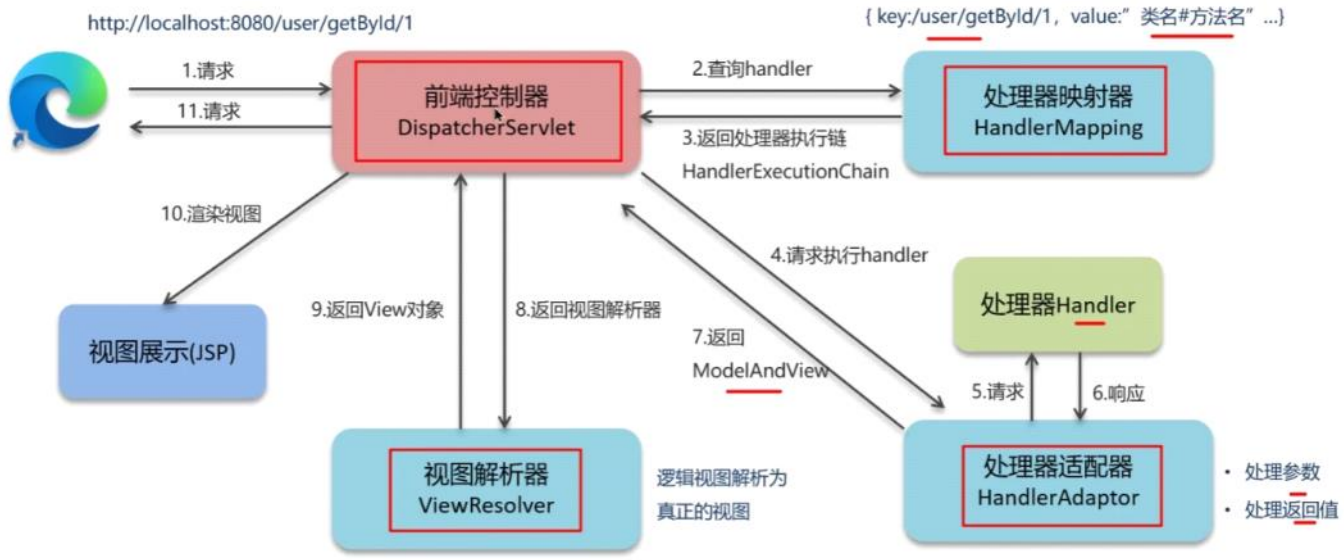
2024年4月6日 12:44

SpringMVC的执行流程知道嘛

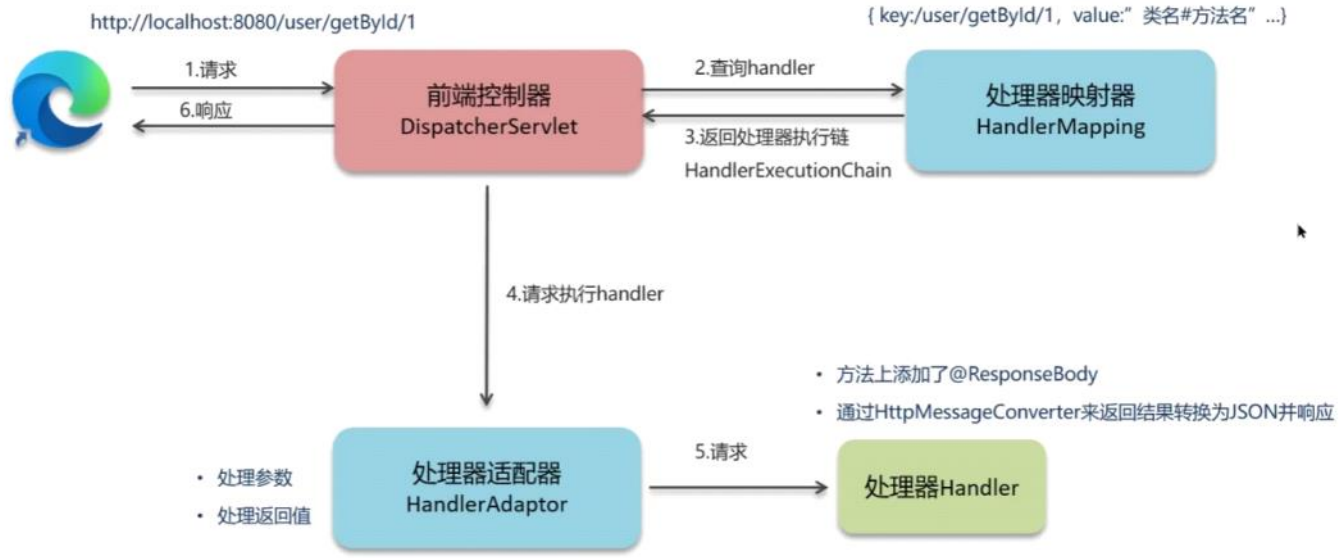
Springmvc的执行流程是这个框架最核心的内容

- 视图阶段（老旧JSP等）
- 前后端分离阶段（接口开发，异步）

视图阶段（JSP）



前后端分离阶段（接口开发，异步请求）



SpringMVC的执行流程知道嘛

① 用户发送出请求到前端控制器DispatcherServlet
② DispatcherServlet收到请求调用HandlerMapping（处理器映射器）
③ HandlerMapping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)，再一起返回给DispatcherServlet。
④ DispatcherServlet调用HandlerAdapter（处理器适配器）
⑤ HandlerAdapter经过适配调用具体的处理器（Handler/Controller）
⑥ Controller执行完成返回ModelAndView对象
⑦ HandlerAdapter将Controller执行结果ModelAndView返回给DispatcherServlet
⑧ DispatcherServlet将ModelAndView传给ViewReslover（视图解析器）
⑨ ViewReslover解析后返回具体View（视图）
⑩ DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）
⑪ DispatcherServlet响应用户

SpringMVC的执行流程知道嘛 （版本2：前后端开发，接口开发）

① 用户发送出请求到前端控制器DispatcherServlet
② DispatcherServlet收到请求调用HandlerMapping（处理器映射器）
③ HandlerMapping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)，再一起返回给DispatcherServlet。
④ DispatcherServlet调用HandlerAdapter（处理器适配器）
⑤ HandlerAdapter经过适配调用具体的处理器（Handler/Controller）
⑥ 方法上添加了@ResponseBody
⑦ 通过HttpMessageConverter来返回结果转换为JSON并响应

Springboot自动配置原理

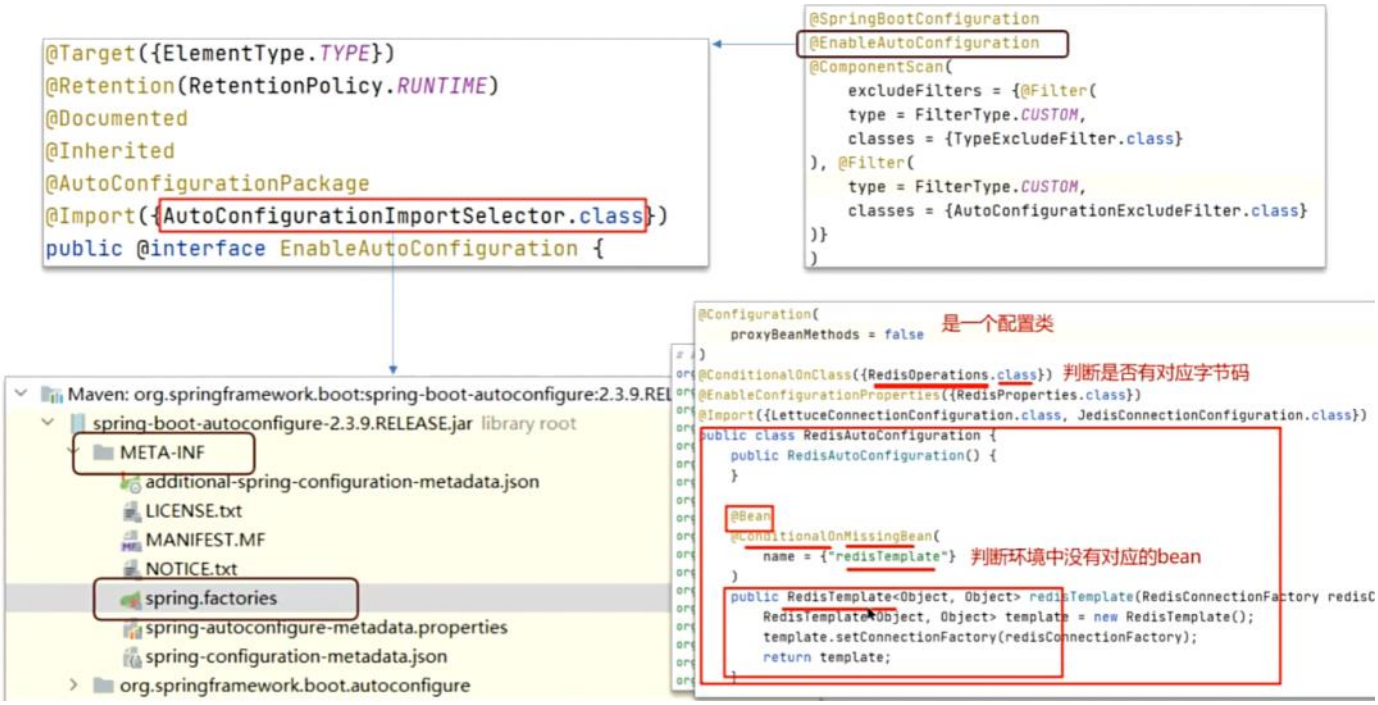
Springboot中最高频的一道面试题，也是框架最核心的思想

```
@SpringBootApplication
public class UserApplication {

    public static void main(String[] args) {
        SpringApplication.run(UserApplication.class,args);
    }
}
```

```
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
    excludeFilters = {@Filter(
    type = FilterType.CUSTOM,
    classes = {TypeExcludeFilter.class}
), @Filter(
    type = FilterType.CUSTOM,
    classes = {AutoConfigurationExcludeFilter.class}
)}
)
```

- @SpringBootConfiguration：该注解与 @Configuration 注解作用相同，用来声明当前也是一个配置类。
- @ComponentScan：组件扫描，默认扫描当前引导类所在包及其子包。
- @EnableAutoConfiguration：SpringBoot实现自动化配置的核心注解。



Springboot自动配置原理

1，在Spring Boot项目中的引导类上有一个注解@SpringBootApplication，这个注解是对三个注解进行了封装，分别是：

- @SpringBootConfiguration
- @EnableAutoConfiguration
- @ComponentScan

2，其中@EnableAutoConfiguration是实现自动化配置的核心注解。 该注解通过@Import注解导入对应的配置选择器。

内部就是读取了该项目和该项目引用的Jar包的的classpath路径下META-INF/spring.factories文件中的所配置的类的全类名。 在这些配置类中所定义的Bean会根据条件注解所指定的条件来决定是否需要将其导入到Spring容器中。

3，条件判断会有像@ConditionalOnClass这样的注解，判断是否有对应的class文件，如果有则加载该类，把这个配置类的所有的Bean放入spring容器中使用。

Spring 的常见注解有哪些?

注解	说明
@Component、@Controller、@Service、@Repository	使用在类上用于实例化Bean
@Autowired	使用在字段上用于根据类型依赖注入
@Qualifier	结合@Autowired一起使用用于根据名称进行依赖注入
@Scope	标注Bean的作用范围
@Configuration	指定当前类是一个 Spring 配置类，当创建容器时会从该类上加载注解
@ComponentScan	用于指定 Spring 在初始化容器时要扫描的包
@Bean	使用在方法上，标注将该方法的返回值存储到Spring容器中
@Import	使用@Import导入的类会被Spring加载到IOC容器中
@Aspect、@Before、@After、@Around、@Pointcut	用于切面编程（AOP）

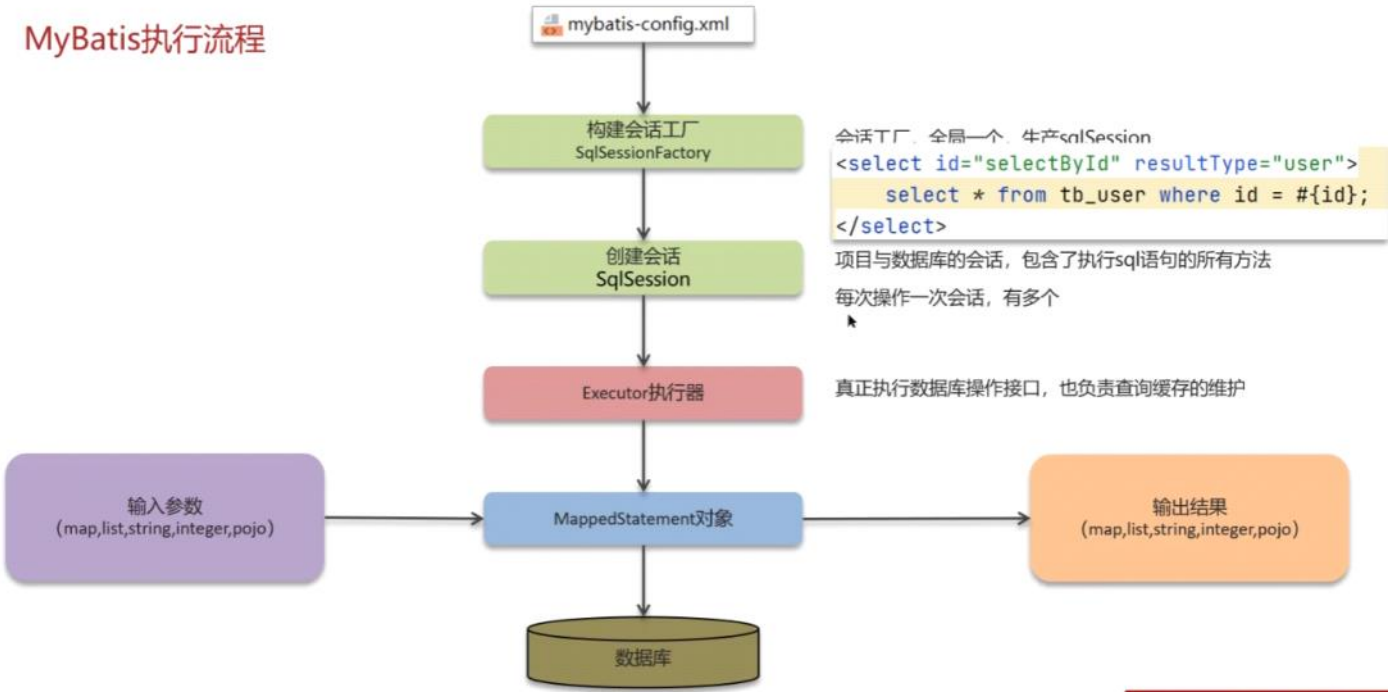
SpringMVC常见的注解有哪些?

注解	说明
@RequestMapping	用于映射请求路径，可以定义在类上和方法上。用于类上，则表示类中的所有的方法都是以该地址作为父路径
@RequestBody	注解实现接收http请求的json数据，将json转换为java对象
@RequestParam	指定请求参数的名称
@PathViriable	从请求路径下中获取请求参数(/user/{id})，传递给方法的形式参数
@ResponseBody	注解实现将controller方法返回对象转化为json对象响应给客户端
@RequestHeader	获取指定的请求头数据
@RestController	@Controller + @ResponseBody

Springboot常见注解有哪些?

注解	说明
@SpringBootConfiguration	组合了- @Configuration注解，实现配置文件的功能
@EnableAutoConfiguration	打开自动配置的功能，也可以关闭某个自动配置的选
@ComponentScan	Spring组件扫描

MyBatis执行流程

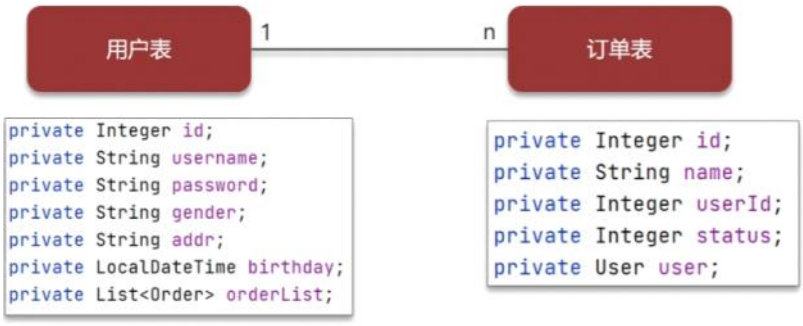


MyBatis执行流程

① 读取MyBatis配置文件：mybatis-config.xml加载运行环境和映射文件
② 构造会话工厂SqlSessionFactory
③ 会话工厂创建SqlSession对象（包含了执行SQL语句的所有方法）
④ 操作数据库的接口，Executor执行器，同时负责查询缓存的维护
⑤ Executor接口的执行方法中有一个MappedStatement类型的参数，封装了映射信息
⑥ 输入参数映射
⑦ 输出结果映射

Mybatis是否支持延迟加载?

Mybatis支持延迟记载，但默认没有开启
什么叫做延迟加载?




```
private Integer id;
private String username;
private String password;
private String gender;
private String addr;
private LocalDateTime birthday;
private List<Order> orderList;
```

```
private Integer id;
private String name;
private Integer userId;
private Integer status;
private User user;
```

查询用户的时候，把用户所属的订单数据也查询出来，这个是立即加载
查询用户的时候，暂时不查询订单数据，当需要订单的时候，再查询订单，这个就是延迟加载

Mybatis是否支持延迟加载?

Mybatis支持延迟记载，但默认没有开启
什么叫做延迟加载?

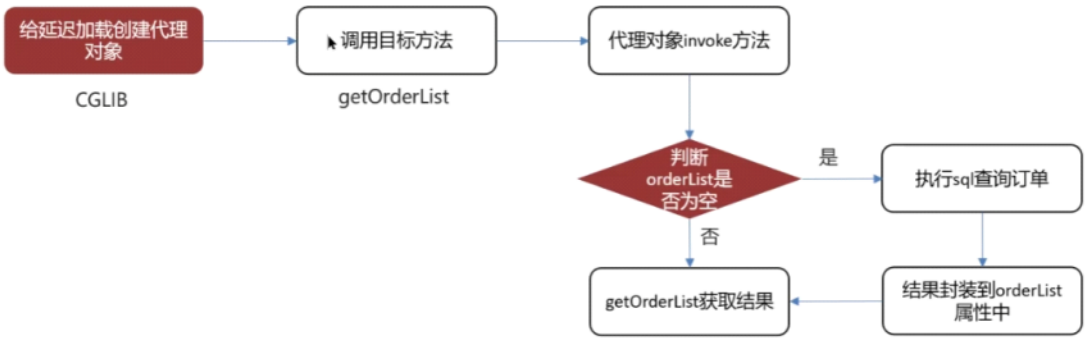

```
private Integer id;
private String username;
private String password;
private String gender;
private String addr;
private LocalDateTime birthday;
private List<Order> orderList;
```

```
private Integer id;
private String name;
private Integer userId;
private Integer status;
private User user;
```

查询用户的时候，把用户所属的订单数据也查询出来，这个是立即加载
查询用户的时候，暂时不查询订单数据，当需要订单的时候，再查询订单，这个就是延迟加载

延迟加载的原理

1. 使用CGLIB创建目标对象的代理对象
2. 当调用目标方法user.getOrderList()时，进入拦截器invoke方法，发现user.getOrderList()是null值，执行sql查询order列表
3. 把order查询上来，然后调用user.setOrderList(List<Order> orderList)，接着完成user.getOrderList()方法的调用



Mybatis是否支持延迟加载?

- 延迟加载的意思是：就是在需要用到数据时才进行加载，不需要用到数据时就不加载数据。
- Mybatis支持一对一关联对象和一对多关联集合对象的延迟加载
- 在Mybatis配置文件中，可以配置是否启用延迟加载lazyLoadingEnabled=true|false，默认是关闭的

延迟加载的底层原理知道吗?

1. 使用CGLIB创建目标对象的代理对象
2. 当调用目标方法时，进入拦截器invoke方法，发现目标方法是null值，执行sql查询
3. 获取数据以后，调用set方法设置属性值，再继续查询目标方法，就有值了

Mybatis的一级、二级缓存用过吗?

- 本地缓存，基于PerpetualCache，本质是一个HashMap
- 一级缓存：作用域是session级别
- 二级缓存：作用域是namespace和mapper的作用域，不依赖于session

一级缓存

一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当Session进行flush或close之后，该Session中的所有Cache就将清空，默认打开一级缓存

```
//2. 获取SqlSession对象，用它来执行sql
SqlSession sqlSession = sqlSessionFactory.openSession();
//3. 执行sql
//3.1 获取UserMapper接口的代理对象
UserMapper userMapper1 = sqlSession.getMapper(UserMapper.class);
UserMapper userMapper2 = sqlSession.getMapper(UserMapper.class);

User user = userMapper1.selectById(6);
System.out.println(user);

System.out.println("----------------------");
User user1 = userMapper2.selectById(6);
System.out.println(user1);
```

只会执行一次sql查询

二级缓存

二级缓存是基于namespace和mapper的作用域起作用的，不是依赖于SQL session，默认也是采用 PerpetualCache，HashMap 存储

```
//2. 获取SqlSession对象，用它来执行sql
SqlSession sqlSession1 = sqlSessionFactory.openSession();

//3. 执行sql
//3.1 获取UserMapper接口的代理对象
UserMapper userMapper1 = sqlSession1.getMapper(UserMapper.class);

User user1 = userMapper1.selectById(6);
System.out.println(user1);
sqlSession1.close();

SqlSession sqlSession2 = sqlSessionFactory.openSession();

System.out.println("----------------------");
UserMapper userMapper2 = sqlSession2.getMapper(UserMapper.class);
User user2 = userMapper2.selectById(6);
System.out.println(user2);

//4.关闭资源
sqlSession2.close();
```

查询两次SQL

二级缓存默认是关闭的

开启方式，两步：

1，全局配置文件

```
<settings>
   <setting name="cacheEnabled" value="true
</settings>
```

2，映射文件

使用<cache/>标签让当前mapper生效二级缓存

二级缓存

注意事项：

1，对于缓存数据更新机制，当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了新增、修改、删除操作后，默认该作用域下所有 select 中的缓存将被 clear

2，二级缓存需要缓存的数据实现Serializable接口

3，只有会话提交或者关闭以后，一级缓存中的数据才会转移到二级缓存中

Mybatis的一级、二级缓存用过吗?

- 一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当Session进行flush或close之后，该Session中的所有Cache就将清空，默认打开一级缓存
- 二级缓存是基于namespace和mapper的作用域起作用的，不是依赖于SQL session，默认也是采用PerpetualCache，HashMap 存储。需要单独开启，一个是核心配置，一个是mapper映射文件

Mybatis的二级缓存什么时候会清理缓存中的数据

当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了新增、修改、删除操作后，默认该作用域下所有 select 中的缓存将被 clear。