今天内容

1. Spring Boot自动配置
2. 编写MyBatis Starter
3. 整合MyBatis-Plus
4. 创建前端工程
5. 创建后端工程
6. 开发字典接口
7. 开发省市区列表接口

1. Spring Boot自动配置

2. 编写MyBatis Starter

1. 创建 `etoak-mybatis-spring-boot-starter` ，导入maven依赖

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.5</version>
</parent>

<dependencies>
  <!-- spring-boot-starter-jdbc -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
  </dependency>

  <!-- mybatis -->
  <dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.10</version>
  </dependency>

  <!-- mybatis-spring -->
  <dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>2.1.0</version>
  </dependency>
```

```
    <!-- configuration-processor -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-configuration-processor</artifactId>
      <optional>true</optional>
    </dependency>

    <!-- autoconfigure-processor -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-autoconfigure-processor</artifactId>
      <optional>true</optional>
    </dependency>
  </dependencies>
```

2. 创建配置类：`com.etoak.configuration.MyBatisAutoConfiguration`
3. 创建Properties配置类：`com.etoak.configuration.MyBatisProperties`
4. 在src/main/resources下创建文件：

   `META-INF/spring/org.springframework.boot.autoconfigure.AutoConfiguration.imports`

   并在文件中添加自动配置类类名：`com.etoak.configuration.MyBatisAutoConfiguration`
5. 修改`MyBatisProperties`（配置mapperLocations、typeAliasesPackage）
6. 修改`MyBatisAutoConfiguration`实现自动配置
7. 将项目安装到本地Maven仓库
8. 在`boot-02-mybatis`中替换原来的MyBatis官方的Starter依赖

3. 整合MyBatis-Plus

1. 官网：https://www.baomidou.com/
2. Spring Boot配置

   mapperLocations默认配置：

   `private String[] mapperLocations = new String[]{"classpath*:/mapper/**/*.xml"};`
3. 使用MyBatis Plus编写SQL

   1、单表的自动生成

   2、条件构造器

   3、手动编写SQL语句（Mapper映射文件中）
4. 用到的注解

   @TableName：表名称和实体类名称不一致时使用

   @TableId：标记主键生成策略
5. 接口、类：BaseMapper、IService、ServiceImpl

   IService接口：通用 Service CRUD 封装IService接口

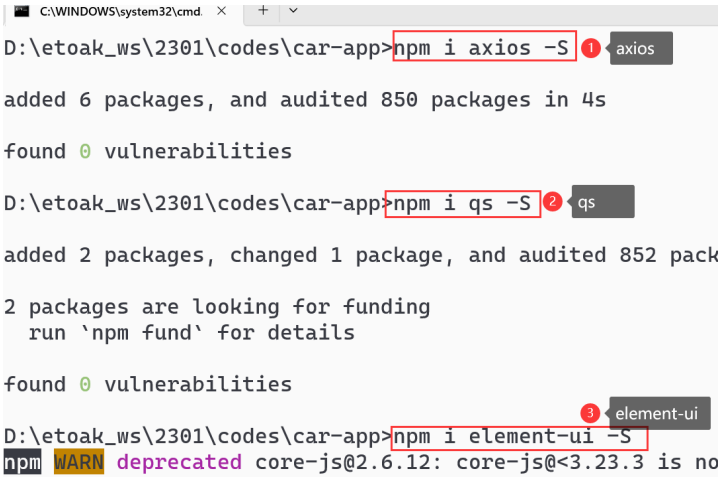
BaseMapper接口：通用 CRUD 封装BaseMapper接口

创建前端工程：

1. vue create car-app

   选择Babel、VueRouter

2. 安装axios、qs、element-ui



创建后端工程

字典接口

1. 接口地址：http://localhost:8000/dict/list?type=level
2. 请求方法：`GET`
3. 请求参数

参数名	参数类型	是否必填	参数说明
type	String	是	type：字典类型 type=level：获取级别列表 type=gearbox：获取变速箱列表 type=disp：获取变速箱列表

4. 响应结果

```
{
  "code": 200,
  "msg": "success",
  "data": [
```

```
    {
      "name": "自动",
      "value": "1"
    },
    {
      "name": "手动",
      "value": "2"
    }
  ]
}
```


省市区级联列表接口

1. 接口地址：http://localhost:8000/area/list
2. 请求方法：`get`
3. 请求参数：`无`
4. 响应结果


```
{
  "code": 200,
  "msg": "success",
  "data": [
    {
      "name": "山东省",
      "id": "370000",
      "children": [
        {
          "name": "济南市",
          "id": "370100",
          "children": [
            {
              "name": "历下区",
              "id": "370102",
              "children": null
            },
            {
              "name": "天桥区",
              "id": "370105",
              "children": null
            }
          ]
        },
        {
          "name": "青岛市",
          "id": "370200",
          "children": [
            {
              "name": "市南区",
```

```
            "id": "370202",
            "children": null
          },
          {
            "name": "市北区",
            "id": "370203",
            "children": null
          }
        ]
      }
    ]
  }
]
}
```