在此例中，我们将采用 MyBatis-Plus进行快速开发，利用其内置的分页功能来实现前后端联动的分页展示。该实现包括项目初始化、后端代码编写以及前端页面的交互。

1 初始化此项目

新建项目勾选你项目所需的依赖库

在pom.xml中注入依赖：构造器所需依赖

```
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>3.3.2</version>
</depdency>
<dependency>
    <groupId>org.apache.velocity</groupId>
    <artifactId>velocity</artifactId>
    <version>1.7</version>
</dependency>
```

MyBatis-Plus可以自动生成基础的 CRUD 代码，并提供了内置的分页功能，简化了分页查询的实现。

2 后端代码模块

2.1 代码自动生成

通过MyBatis-Plus提供的代码生成器来快速生成基础的 Controller、Service、Mapper 等模块，使用GenerateTest类来实现：

```
package com.dust;

import com.baomidou.mybatisplus.generator.AutoGenerator;
import com.baomidou.mybatisplus.generator.config.DataSourceConfig;
import com.baomidou.mybatisplus.generator.config.GlobalConfig;
import com.baomidou.mybatisplus.annotation.DbType;
import com.baomidou.mybatisplus.generator.config.PackageConfig;
import com.baomidou.mybatisplus.generator.config.StrategyConfig;
import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;

public class GenerateTest {
    public static void main(String[] args) {
       //创建generator对象
        AutoGenerator autoGenerator = new AutoGenerator();
       //数据源
        DataSourceConfig dataSourceConfig = new DataSourceConfig();
        dataSourceConfig.setDbType(DbType.MYSQL);
        dataSourceConfig.setDriverName("com.mysql.cj.jdbc.Driver");
        dataSourceConfig.setUsername("root");
        dataSourceConfig.setPassword("102233");
        dataSourceConfig.setUrl("jdbc:mysql://localhost:3306/mbtest");
        autoGenerator.setDataSource(dataSourceConfig);
       //全局配置
        GlobalConfig globalConfig = new GlobalConfig();
        globalConfig.setOutputDir(System.getProperty("user.dir")+"/src/main/java");
        globalConfig.setAuthor("admin");
        globalConfig.setOpen(false);
        globalConfig.setServiceName("%sService");
        autoGenerator.setGlobalConfig(globalConfig);
       //包信息
        PackageConfig packageConfig = new PackageConfig();
        packageConfig.setParent("com.dust");
        packageConfig.setEntity("entity");;
```

```
        packageConfig.setMapper("mapper");
        packageConfig.setService("service");
        packageConfig.setServiceImpl("service.impl");
        packageConfig.setController("controller");
        autoGenerator.setPackageInfo(packageConfig);
       //策略配置
        StrategyConfig strategyConfig = new StrategyConfig();
        strategyConfig.setInclude("fruit");
        strategyConfig.setNaming(NamingStrategy.underline_to_camel);
        strategyConfig.setColumnNaming(NamingStrategy.underline_to_camel);
        strategyConfig.setEntityLombokModel(true);
        autoGenerator.setStrategy(strategyConfig);
       //运行
        autoGenerator.execute();
    }
}
```

该代码生成器会根据数据库中的表结构自动生成实体类、Mapper、Service、Controller 等，极大提升开发效率。

2.2 分页插件配置

由于使用MyBatis-Plus构造器自动构造我们不用手动搭建各层结构。实体类直接依照数据自动能够创建。下面我们只需要编写以下代码：

> 1、引入MyBatis-Plus自带的分页器，确保分页查询的性能和简便性

在工具配置文件下新建一个类 PageConfiguration

```
package com.dust.configuration;

import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * 使用mybatis-plus自带分页插件
 */
@Configuration
@MapperScan("com.dust.mapper")
public class PageConfiguration {
    //注入mybatis-plus分页器PaginationInterceptor
    @Bean
    public PaginationInterceptor paginationInterceptor(){
        return  new PaginationInterceptor();
    }

}
```

2.3 Controller 层实现分页

在ProController中编写分页查询逻辑。使用 `IPage` 来封装分页结果，`Page` 对象用来接收分页参数：

> 2、Controller层

```
package com.dust.controller;


import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.dust.entity.Fruit;
```

```
import com.dust.entity.LayuiPage;
import com.dust.entity.Pro;
import com.dust.service.ProService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import org.springframework.stereotype.Controller;

import java.util.List;

/**
 * <p>
 *  前端控制器
 * </p>
 *
 * @author admin
 * @since 2022-02-24
 */
@RestController
@RequestMapping("/pro")
public class ProController {

    @Autowired
    private ProService proService;

    @RequestMapping("/pagelist")
    @ResponseBody
    //或者是GetMapping("/pagelist")
    public LayuiPage pagelist(int page, int limit){
        Page<Pro> pager=new Page<>(page,limit);
        IPage<Pro> proIPage =proService.page(pager,new QueryWrapper<>());
       //调用layuipage构造的方法
        //proIPage.getTotal()是指页面内数量
        //proIPage.getRecords()是指页数据
        return new LayuiPage(proIPage.getTotal(),proIPage.getRecords());
}
}
```

这里使用了@RequestParam来接收分页参数 page 和 limit，并通过ProService 调用MyBatis-Plus 的 page 方法实现分页查询，返回的 LayuiPage封装了总记录数和分页数据。

2.4 LayuiPage 实体类

用LayuiPage来封装分页结果，适配前端 LayUI 表格的格式：

> 3、layuipage实体类

```
package com.dust.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class LayuiPage<T> {
    private int code;
```

```
    private String msg;
    private Long count;
    private List<T> data;
   //此构造函数只包含了两个参数，当然这两个参数是够用的，如果可以可以构造四个参数
    public LayuiPage(Long count, List<T> data) {
        this.count = count;
        this.data = data;
    }
}
```

3 前端代码模块

前端采用layui，界面表格可以去layui官网选择粘贴到templates下改一下table模块下的代码与实体类对应，并且还要更改跳转接口与Controller层controller类url一致。示例：

```
<table class="layui-hide" id="demo" lay-filter="test"></table>

<script type="text/javascript">
    layui.use('table', function(){
        var table = layui.table;
       //执行渲染
        table.render({
            elem: '#demo',
            url: '/pro/pagelist', // 数据接口
            page: true, // 开启分页
            cols: [[
                {field: 'id', title: 'ID', width: 80, sort: true},
                {field: 'name', title: '名称', width: 120},
                {field: 'price', title: '价格', width: 100},
                {field: 'stock', title: '库存', width: 100}
            ]]
        });
    });
</script>
```