AspirineManager

Description

Aspirine Manager 医药管理系统.

南京信息工程大学 暑期实训课程.

「You wont take me alive.」 *Tommy Vercetti ,Grand Theft Auto:Vice City*.

Source

GITEE.com （国内用户推荐） Github.com

开发全程可追溯. ***绝无抄袭行为***.

由于教学需要，项目并为打包（War），因此不能保证在不同平台的移植性。
如：项目在构建时需要 Tomcat 10 作为 Servlet 容器.同时需要 JDK 1.8.0_291 作为编译最低版本.
若因此造成的软件在质量上的缺失，概不负责.

若在测试过程中遇到问题，开发者可以提供 15 日以内的技术支持.
请联系 开发人员 ，并留下您的联系方式。

Developer

- LviatYi 易家俊
  - 主程.
  - 数据库管理员.
  - 架构设计.
- Rooter 徐淳
  - 后端开发工程师.
- nbyxs 陈梓濠
  - 项目组长.
  - 前端开发工程师.

- Titc-s 汤硕
  - 后盾开发工程师.
  - 产品文档维护.

Environment

IntelliJ IDEA Community Edition 2021.1.3 Visual Studio Code

Dev with

- Maven 3.8.1
- Tomcat 10.1-M2
- Jdk 1.8.0-291

Plugin with

- Alibaba Java Coding Guidelines 2.1.1
- Smart Tomcat 3.8.5
- PlantUML Diagram Generator 2021.1.2

Database with

- Oracle Database 19c.

Documentation

- UserStory

技术栈介绍

MAVEN

使用 Maven 作为项目构建工具.

Maven 是一个跨平台的项目管理工具.作为 Apache 组织的一个颇为成功的开源项目，其主要服务于基于 Java 平台的项目创建，依赖管理和项目信息管理.maven 是 Apache 的顶级项目，解释为「专家，内行」，它是一个项目管理的工具，maven 自身是纯 java 开发的（ Theresult is a tool that can now be used for building and managing any Java-basedproject ），可以使用 maven 对 java 项目进行构建、依赖管理.

Maven 功能：

- 依赖管理
- 一键构建
- 跨平台
- 应用在大型的项目种可以提高开发效率

MyBatis

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射.MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集.MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs（Plain Ordinary Java Object,普通的 Java 对象）映射成数据库中的记录.

我们把 Mybatis 的功能架构分为三层：

- API 接口层：提供给外部使用的接口 API，开发人员通过这些本地 API 来操纵数据库.接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理.
- 数据处理层：负责具体的 SQL 查找、SQL 解析、SQL 执行和执行结果映射处理等.它主要的目的是根据调用的请求完成一次数据库操作.
- 基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件.为上层的数据处理层提供最基础的支撑.

MyBatis 最强大的特性之一就是它的动态语句功能.如果您以前有使用 JDBC 或者类似框架的经历，您就会明白把 SQL 语句条件连接在一起是多么的痛苦，要确保不能忘记空格或者不要在 columns 列后面省略一个逗号等.动态语句能够完全解决掉这些痛苦.

尽管与动态 SQL 一起工作不是在开一个 party，但是 MyBatis 确实能通过在任何映射 SQL 语句中使用强大的动态 SQL 来改进这些状况.动态 SQL 元素对于任何使用过 JSTL 或者类似于 XML 之类的文本处理器的人来说，都是非常熟悉的.在上一版本中，需要了解和学习非常多的元素，但在 MyBatis 3 中有了许多的改进，现在只剩下差不多二分之一的元素.MyBatis 使用了基于强大的 OGNL 表达式来消除了大部分元素.

MyBatis Generator

MyBatis Generator（MBG）是 MyBatis MyBatis 和 iBATIS 的代码生成器.它将为所有版本的 MyBatis 以及版本 2.2.0 之后的 iBATIS 版本生成代码.它将内省数据库表（或许多表），并将生成可用于访问表的工件.这减少了设置对象和配置文件以与数据库表交互的初始麻烦.MBG 寻求对简单 CRUD（创建，检索，更新，删除）的大部分数据库操作产生重大影响.

- 如果存在与新生成的 XML 文件同名的现有文件，MBG 将自动合并 XML 文件.MBG 不会覆盖您对其生成的 XML 文件所做的任何自定义更改.您可以反复运行它，而不必担心会丢失对 XML 的自定义更改.MBG 将替换先前运行中生成的任何 XML 元素.
- MBG 不会合并 Java 文件，它可以覆盖现有文件或使用不同的唯一名称保存新生成的文件.如果对生成的 Java 文件进行更改并以迭代方式运行 MBG，则必须手动合并更改.当作为 Eclipse 插件运行时 ， MBG 可以自动合并 Java 文件.

Mybatis page helper

使用 Mybatis page helper 作为分页插件管理分页.

使用 PageHelper 需要在 pom.xml 中配置 PageHelper 的依赖.

PageHelper.offsetPage(PAGE_NUM, PAGE_SIZE)实际做的事情是在 ThreadLocal 中设置了分页参数，之后在查询执行的时候，获取当线程中的分页参数，执行查询的时候通过拦截器在 sql 语句中添加分页参数，之后实现分页查询，查询结束后在 finally 语句中清除 ThreadLocal 中的查询参数

最核心的逻辑在 PageInterceptor 中，PageInterceptor 是一个拦截器.

在 Mybatis 中有四个地方提供了拦截的机会

- Executor
- ParameterHandler
- ResultSetHandler
- StatementHandler

分页插件拦截的是 Executor ， 也就是在 sql 执行的时候

MVC 与三层架构

MVC

- 视图 View
  - 负责页面的显示；与用户的交互.包含各种表单. 实现视图用到的技术有 html/css/jsp/js 等前端技术.
  - 用户交互：用户鼠标点击页面；填写页面中各种表单.....等等
- 模型 Model 模型负责各个功能的实现（如登录、增加、删除功能）.模型用 JavaBean 实现.
- 控制器 Controller 控制器负责将视图与模型一一对应起来.相当于一个模型分发器.所谓分发就是：
  - 接收请求，并将该请求跳转（转发，重定向）到模型进行处理.
  - 模型处理完毕后，再通过控制器，返回给视图中的请求处.使用了 Servlet 实现控制器.

三层架构

首先来说，三层架构与 MVC 的目标一致：都是为了解耦和、提高代码复用.MVC 是一种设计模式，而三层架构是一种软件架构. 三层架构分为：表现层（UI）(web 层)、业务逻辑层（BLL）(service 层)、数据访问层（DAL）(dao 层)，再加上实体类库（Model）.

- 实体类库（Model），在 Java 中，往往将其称为 Entity 实体类.数据库中用于存放数据，而我们通常选择会用一个专门的类来抽象出数据表的结构，类的属性就一对一的对应这表的属性. 一般来说，Model 实体类库层需要被 DAL 层，BIL 层和 UI 层引用.
- 数据访问层（DAL），主要是存放对数据类的访问，即对数据库的添加、删除、修改、更新等基本操作 DAL 就是根据业务需求，构造 SQL 语句，构造参数，调用帮助类，获取结果，DAL 层被 BIL 层调用
- 业务逻辑层（BLL） BLL 层好比是桥梁，将 UI 表示层与 DAL 数据访问层之间联系起来.所要负责的，就是处理涉及业务逻辑相关的问题，比如在调用访问数据库之前，先处理数据、判断数据.

Servlet 与 Filter

Filter

Filter，过滤器，顾名思义，即是对数据等的过滤，预处理过程.为什么要引入过滤器呢？在平常访问网站的时候，有时候发一些敏感的信息，发出后显示时 就会将敏感信息用*等字符替代，这就是用过滤器对信息进行了处理.这只是一个简单的例子，当然，过滤器那么强大，它的功能也不可能局限于此，它不仅能预处 理数据，只要是发送过来的请求它都是可以预处理的，同时，它还可以对服务器返回的响应进行预处理，这样，大大减轻了服务器的压力.例如，实现 URL 级别的 权限访问控制、过滤敏感词汇、压缩响应信息等一些高级功能.

Filter 起过滤作用，对从客户端向服务器端发送的请求进行过滤，也可以对服务器端返回的响应进行处理.它使用户可以改变一个 request 和修改一个 response..Filter 不是一个 servlet，它不能产生一个 response，但是它能够在一个 request 到达 servlet 之前预处理 request，也可以在 response 离开 servlet 时处理 response.换句话说，filter 其实是客户端与 servlet 中间的一个传递者，并且它可以对要传递 的东西进行修改.

实现原理

- 当客户端发生请求后，在 HttpServletRequest 到达 Servlet 之前，过滤器拦截客户的 HttpServletRequest .
- 根据需要检查 HttpServletRequest ， 也可以修改 HttpServletRequest 头和数据.
- 在过滤器中调用 doFilter 方法，对请求放行.请求到达 Servlet 后，对请求进行处理并产生 HttpServletResponse 发送给客户端.
- 在 HttpServletResponse 到达客户端之前，过滤器拦截 HttpServletResponse .
- 根据需要检查 HttpServletResponse ， 可以修改 HttpServletResponse 头和数据.
- 最后，HttpServletResponse 到达客户端.

Filter 接口中有三个重要的方法.

- init()方法：初始化参数，在创建 Filter 时自动调用.当我们需要设置初始化参数的时候，可以写到该方法中.
- doFilter()方法：拦截到要执行的请求时，doFilter 就会执行.这里面写我们对请求和响应的预处理.
- destroy()方法：在销毁 Filter 时自动调用.

生命周期

Filter 的创建和销毁由 web 服务器控制.

- 服务器启动的时候，web 服务器创建 Filter 的实例对象，并调用其 init 方法，完成对象的初始化功能.filter 对象只会创建一次，init 方法也只会执行一次.
- 拦截到请求时，执行 doFilter 方法.可以执行多次.
- 服务器关闭时，web 服务器销毁 Filter 的实例对象.

过滤器是用来拦截请求和响应的，不能产生响应，而 servlet 是用来处理请求并产生响应的.

JWT

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于 JSON 的开放标准（(RFC 7519).该 token 被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景.JWT 的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该 token 也可直接被用于认证，也可被加密.

JWT 是由三段信息构成的，将这三段信息文本用.链接一起就构成了 Jwt 字符串.

JWT 的构成

第一部分为头部（header),第二部分为载荷（payload, 类似于飞机上承载的物品)，第三部分是签证（signature).

- header
  jwt 头部承载两部分信息:
  - 声明类型，这里是 jwt
  - 声明加密的算法 通常直接使用 HMAC SHA256

playload

载荷就是存放有效信息的地方.这个名字像是特指飞机上承载的货品，这些有效信息包含三个部分

- 标准中注册的声明
- 公共的声明

- 私有的声明

标准中注册的声明

- **iss**: JWT 签发者
- **sub**: JWT 所面向的用户
- **aud**: 接收 JWT 的一方
- **exp**: JWT 的过期时间，这个过期时间必须要大于签发时间
- **nbf**: 定义在什么时间之前，该 JWT 都是不可用的.
- **iat**: JWT 的签发时间
- **jti**: JWT 的唯一身份标识，主要用来作为一次性 token,从而回避重放攻击.
- **公共的声明** 公共的声明可以添加任何的信息，一般添加用户的相关信息或其他业务需要的必要信息.但不建议添加敏感信息，因为该部分在客户端可解密.
- **私有的声明** 私有声明是提供者和消费者所共同定义的声明，一般不建议存放敏感信息，因为 base64 是对称解密的，意味着该部分信息可以归类为明文信息.

Signature

JWT 的第三部分是一个签证信息，这个签证信息由三部分组成：

- header (base64 后的)
- payload (base64 后的)
- secret

这个部分需要 base64 加密后的 header 和 base64 加密后的 payload 使用 `.` 连接组成的字符串，然后通过 header 中声明的加密方式进行 secret 组合加密，然后就构成了 jwt 的第三部分.

- 因为 json 的通用性，所以 JWT 是可以进行跨语言支持的，像 JAVA,JavaScript,NodeJS,PHP 等很多语言都可以使用.
- 因为有了 payload 部分，所以 JWT 可以在自身存储一些其他业务逻辑所必要的非敏感信息.
- 便于传输，JWT 的构成非常简单，字节占用很小，所以它是非常便于传输的.
- 它不需要在服务端保存会话信息, 所以它易于应用的扩展

Logback

Logback 是 Log4j 的改良版本，Logback 是 SpringBoot 内置的日志处理框架.

模块：

Logback-core：其它两个模块的基础模块

- Logback-classic：它是 log4j 的一个改良版本，同时它完整实现了 slf4j API 使你可以很方便地更换成其它日志系统如 log4j 或 JDK14 Logging
- Logback-access：访问模块与 Servlet 容器集成提供通过 Http 来访问日志的功能