核心功能：

- 用户管理（学生、老师、管理员多角色）
- 课程管理（视频上传、章节管理）
- 订单支付
- 直播
- 作业提交与批改

第二周：用户权限管理

- 用户注册登录功能
- JWT Token 认证机制
- Spring Security 权限控制
- 多角色权限设计
- 密码加密与安全
- 用户信息管理接口

环境准备与依赖配置

```
<dependencies>
  <!-- Spring Security -->
  <dependency>
    <groupID> *** </groupID>
    <artifactId> *** </artifactId>
  </dependency>

  <!-- JWT -->
  <denpendency>
    <groupID>io.jsonwebtoken</groupID>
    <artifactId>jjwt-api</artifactId>
    <version> *** </version>
```

```
    </denpendency>
    <denpendency>
      <groupID>io.jsonwebtoken</groupID>
      <artifactId>jjwt-impl</artifactId>
      <version> *** </version>
    </denpendency>

    <!-- 密码加密 -->
     <denpendency>
      <groupID>arg.springframework.security</groupID>
      <artifactId>spring-security-crypto</artifactId>
      <version> *** </version>
    </denpendency>
```

JWT 配置属性

```
  jwt:
    secret: adsfdsifsiudaiow12313e2
    expiration: 604800
    header: Authorization
    prefix: "Bearer "
```

JWT 工具类开发

JWT 配置类

```
@Data
@Component
@ConfigurationProerties(prefix = "jwt")
public class JwtProperties{
  private String secret;
  private Long expiration;
  private String header;
  private String prefix;
}
```

JWT 工具类

```
@Component
@Slf4j
public class JwtTokenUtil {
  @Autowired
  private JwtProperties jwtProperties;

  private Key getSignKey() {
    byte[] keyBytes =
jwtProperties.getSecret().getBytes(StandardCharsets.UTF_8);
    return Keys.hmacShaKeyFor(keyBytes);
  }

  /*
  生成 Token
  */
  public String generateToken(String username, String role, Long userId)
{
    Date now = new Date();
    Date expiryDate = new Date(now.getTime() +
jwtProperties.getExpiration() * 1000);
```

```
    return Jwts.builder()
      .setSubject(username)
      .claim("role", role)
      .claim("userId", userId)
      .setIssuedAt(now)
      .setExpiration(expiryDate)
      .signWith(getSignKey(), SignatureAlgorithm.HS512)
      .compact();
  }


  /*
  从 Token 获取Claims
  */
  public String getClaimsFromToken(String token) {
    return Jwts.parserBuilder()
      .setSigningKey(getSignKey())
      .build()
      .parseClaimsJws(token)
      .getBody();
  }

  /*
  从 Token 获取用户名
  */
  public String getUsernameFromToken(String token) {
    Claims claims = getClaimsFromToken(token);
    return claims.getSubject();
  }

  /*
  从 Token 获取用户角色
  */
```

```
    /*
    从 Token 获取用户ID
    */

    /*
    验证 Token 是否有效
    */
    public boolean validateToken(String token) {

    }

    /*
    判断 Token 是否过期
    */
    public boolean isTokenExpired(String token) {

    }
}
```

用户详情服务实现

自定义 UserDetails 实现

```
@Data
public class UserPrincipal implements UserDetails {
    private Long id;
    private String username;
    private String password;
    private String email;
    ****
```

```
    public boolean isAccountNonExpired() {

    }

    public boolean isAccountNonLocked() {

    }
}
```

UserDetailsService 实现

```
@Service
@Transactional
@Slf4j
public class UserDetailServiceImpl implements UserDetailsService {
  @Autowired
  private UserRepository userRepository;

}
```

用户注册登录功能

DTO 定义

```
// 注册请求 DTO
@Data
@Valid
public class RegisterRequest {
  @NotBlank(message = "用户名不能为空")
  @Size(min = 3, max = 20, message = "用户名长度必须在 3-20 个字符之间")
  private String username;

  @NotBlank(message = "密码不能为空")
```

```
    @Size(min = 6, max = 20, message = "密码长度必须在 6-20 个字符之间")
    private String password;

    @Email(message = "邮箱格式不正确")
    private String email;

    @Pattern(regexp = "^1[3-9]\\d{9}$", message = "手机号格式不正确")
    private String phone;

    private String nickname;

    @NotNull(message = "角色不为空")
    private UserRole role;
}
```

用户 Repository 层

```
/**
从 UserRepository 中获取:
findByUsername
findByEmail
findByPhone
existsByUsername
existsByEmail
existsByPhone
*/
```

用 Service 层

```
/**
用户注册：
  检查用户名是否存在
  检查邮箱是否存在
  检查手机号是否存在
  创建用户
用户登录：
  认证
  获取用户信息
  生成 Token
  构造响应
**/
```

用户信息管理接口

用户管理 Service

```
// 更新用户信息：检查邮箱是否被其他用户占用、用户名是否被占用
// 修改密码：是否和原密码一样
// 检查用户是否是资源拥有者
```

开源资源

https://github.com/gavenwangcn/vole

https://github.com/vt-middleware/passay

https://github.com/elunez/eladmin

作业

作业1：JWT认证实现

1. 生成JWT的静态创建JWT工具类

1. 集成JWT依赖，创建JWT工具类
2. 实现用户注册和登录接口
3. 配置Spring Security，添加JWT过滤器
4. 测试认证流程是否正常工作

作业2：权限控制实现

1. 实现多角色权限控制（学生、教师、管理员）
2. 使用@PreAuthorize注解控制方法级权限
3. 创建不同角色的测试接口
4. 测试权限控制是否生效

作业3：用户管理功能

1. 实现用户信息更新接口
2. 实现密码修改功能
3. 添加用户状态管理（启用/禁用）
4. 编写相应的单元测试

作业4：安全加固

1. 添加密码复杂度验证
2. 实现登录失败次数限制
3. 添加请求频率限制
4. 完善异常处理和日志记录