**后端功能开发详细步骤**

**一、导入SQL**

SQL文件是 ctf\_platform\_20220722.sql,导入后数据库名是 ctf\_platform。

**二、启动顺序**

因为有依赖，所以启动顺序如下：

1. Redis
2. Nacos服务端
3. ctf-gateway 网关模块的 CtfGatewayApplication
4. ctf-auth 认证授权模块的 CtfAuthApplication
5. ctf-admin-service 系统管理模块的 CtfAdminApplication
6. ctf-account-service 账户管理模块的 CtfAccountApplication
7. 其他模块的 Application

**三、项目目录结构**

需要前端使用代码生成器，智能生成前后端各层次基础代码，提高开发效率。

新增功能以 用户管理 的 编辑用户 为例

**四、项目开发**

**1、编写Mybatis数据持久层的SQL**

示例代码：

<update id="updateSysUser" parameterType="SysUser">

UPDATE mscode\_sys\_user

<set>

<if test="password != null and password != ''">password=#{password},</if>

<if test="email != null and email != ''">email=#{email},</if>

<if test="mobile != null and mobile != ''">mobile=#{mobile},</if>

<if test="roleId != null">role\_id=#{roleId},</if>

<if test="orgId != null">org\_id=#{orgId},</if>

<if test="status != null">status=#{status},</if>

prefix=#{prefix,jdbcType=VARCHAR}

</set>

WHERE id=#{id,jdbcType=BIGINT}

</update>

为了安全性和健壮性，Mybatis 数据持久层不要缺少对字段非空和类型的判断处理。

**2、编写数据持久接口层的代码**

示例代码：

/\*\*

\* 编辑用户

\*

\* @param sysUser 用户对象

\* @return

\*/

int updateSysUser(SysUser sysUser);

**3、编写业务逻辑实现层的代码**

示例代码：

/\*\*

\* 编辑用户

\*/

@Override

public void updateSysUser(SysUser sysUser) {

Integer existing = sysUserMapper.getSysUserByIdEmailMobile(sysUser.getId(), sysUser.getEmail(), sysUser.getMobile());

if (existing != null && existing > 0) {

throw new IllegalArgumentException("邮箱或手机号已存在");

}

if (StringUtils.isNotBlank(sysUser.getPassword())) {

sysUser.setPassword(encoder.encode(sysUser.getPassword()));

}

sysUserMapper.updateSysUser(sysUser);

logger.info("用户已编辑： {}", sysUser.getUsername());

}

**4、编写业务逻辑接口层的代码**

示例代码：

/\*\*

\* 编辑用户

\*

\* @param sysUser 用户对象

\*/

void updateSysUser(SysUser sysUser);

**5、编写控制层的代码**

前端传来的参数必须经过 @Validated 校验。

控制层的 ResultBuilder 统一处理可能出现的异常，并写入日志。

示例代码：

/\*\*

\* 编辑用户

\*

\* @param sysUser 用户对象

\* @return

\*/

@RequestMapping(path = "/updateSysUser", method = RequestMethod.PUT)

public ActionResult updateSysUser(@Validated(UpdateValidator.class) @RequestBody SysUser sysUser) {

ActionResult result = new ActionResult();

try {

sysUserService.updateSysUser(sysUser);

ResultBuilder.buildActionSuccess(result);

} catch (Exception e) {

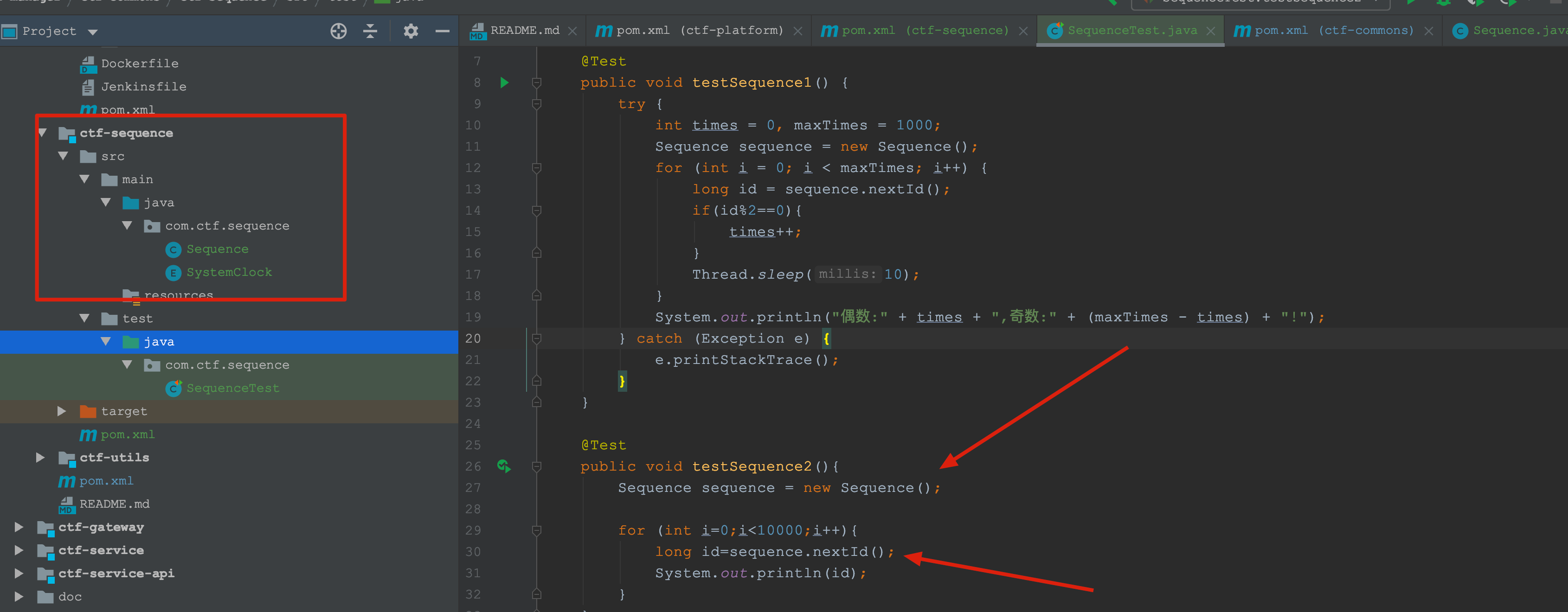
ResultBuilder.buildActionWarn(result, e);

}

return result;

}

**五、增加分布式全局唯一ID生成器**



使用方法，直接在项目中实例化，分布式，部署不统计器上时候，实例化构造器需要采用有参。