智能凭证生成开发文档

1. 概述

智能凭证生成 是财务系统中记账板块的一个重要功能。该功能的开发旨在提高记账效率和准确性，减少人工操作和错误。它的主要作用是在使用其他业务模块时，根据系统配置，自动从实际业务数据中生成「会计凭证」。此外，也可以手动录入会计凭证。通过会计凭证的记录，系统可以自动计算特定「会计期间」每个「会计科目」的期初余额、本期发生额和期末余额等「科目余额」信息。这些信息是编制财务报表的基础。

2. 功能需求

2.1 会计期间管理

- 系统应支持创建和修改会计期间，支持对当前会计期间进行结账，对非当前会计期间进行反结账，确保记账流程的完整性。
- 用户应能够设定会计期间的起始日期，并为其设定凭证编号。

2.2 会计科目管理

- 系统应支持创建、修改和删除会计科目，确保会计科目的准确分类和管理。
- 用户应能够为会计科目设置科目编号、科目名称、科目类别和层级关系等属性。

2.3 会计凭证管理

- 系统应支持录入、修改和打印会计凭证。
- 系统应支持自定义配置，由其他业务模块自动生成会计凭证。
- 用户应能够设定凭证的日期、借贷方金额、会计科目和摘要等信息，并指定上级进行审核操作。

2.4 科目余额表管理

- 系统应根据会计凭证的记录自动生成科目余额表，展示各个会计科目在特定会计期间的期初余额、本期发生和期末余额。
- 用户应能够查询和导出科目余额表，以便进行财务分析和报表编制。

3. 技术实现

3.1 开发语言及框架

- 前端：HTML、CSS、JavaScript、jQuery、Vue等
- 后端：Java、JFinal、Spring、MySQL等
- 数据库：使用关系型数据库 MySQL，用于存储会计期间、会计科目、会计凭证和会计余额等相关数据。

3.2 主要接口

- 会计期间

```
获取会计期间：getAccountingPeriodData()

获取显示的功能按钮：public void getDisplayedFunctionButtons()

结账与反结账：public void settLeAccountsAndBackSettleAccounts()
```

- 会计科目

```
验证科目：verifyAccountingSubject()

获取会计科目类型数据：getSubjectTypeData()

根据科目类型获取科目：getSubjectTreeDataByType()

根据父科目获取科目：getSubjectByParentSubjectId()

根据name查询科目：searchSubjectByName()

辅助核算记录查询：findSupplementaryAccountingEntityRecord()

获取可用的科目编码：getAvailableSubjectCode()

获取辅助核算：getSuppLementaryAccountingData()

获取科目大类：getSubj ectBigClassData()
```

- 会计凭证

```
通过用户ID获取会计凭证数据：getAccountingVoucherById()

储存会计凭证：saveAccountingVoucher()

导出会计凭证到ExceL：exportAccountingVoucherDetailDataToExcel()

重新生成凭证编号：regenerateAccountingVoucherNumber()

一键生成凭证：generateAccountingVoucher()

生成结转损益凭证：carryForwardProfitAndLoss()
```

- 科目余额

```
查询科目余额：getSubjectBalanceTableData()

导出到Excel：exportToExcel()
```

3.3 数据库字段设计

- 会计期间表：开始日期、是否当前期间、状态、凭证编号等字段。
- 会计科目表：上级科目、科目代码、科目名称、科目大类、科目类别、余额方向、辅助核算、层级、等字段。
- 科目余额表：会计期间、科目编码、科目名称、科目、上级科目、辅助核算、辅助核算记录、期初借方、期初贷方、本期借方、本期贷方、期末借方、期末贷方、子记录数量、余额方向等字段。

- 会计凭证表：记账日期、凭证字、编号、会计期间、附单据、业务日期、借方汇总、贷方汇总、业务来源模块、业务来源记录等字段。

4. 智能凭证生成简要流程

- 根据会计期间和会计科目，从会计凭证表中筛选对应的凭证记录。
- 遍历凭证记录，根据会计科目和借贷方金额进行计算。
- 根据期初余额、本期发生和借贷方金额计算期末余额。
- 更新科目余额表中对应会计科目和会计期间的数据。

5. 异常处理

- 对于凭证记录缺失或错误的情况，进行异常处理并记录日志，确保数据的准确性和完整性。
- 对于期初余额、本期发生和期末余额计算错误的情况，进行异常处理并记录日志，以便进行问题排查和修复。

6. 性能优化

- 对于大规模数据处理，使用并发处理和批量处理，提高生成效率。
- 对于频繁查询的场景，使用 Ehcache 缓存，减少数据库访问次数。

7. 测试和验证

- 设计合适的测试用例，包括正常情况和异常情况的测试。
- 进行单元测试、集成测试和系统测试，确保智能凭证生成模块的稳定性和正确性。
- 与其他模块进行集成测试，验证数据的一致性和准确性。