質量i'liang﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽﷽

**Vehicle Service System**

陸路運輸工具管理系統

編程規範﹣Programming Guideline

Prepared by: Atos Information Technology HK Limited

Date: 15/2015

Version: 0.3

變更記錄

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **備註** |
| 25/05/2015 | Crystal Zhao  David Siu  Steven | 1 | 初稿 |

目錄

1 目的 Objective 5

2 對象 Audience 6

3 術語 Terminology 7

4 框架設計 Framework Design 8

4.1 Java Library Used 8

4.2 前端開發框架 9

5 項目結構Project Structure 10

5.1 前臺文件結構 10

5.2 後臺結構 10

6 環境建設System Environment Setup<to be confirmed> 11

7 編程規範 Programming Convention 13

7.1 命名規範 Naming Convention 13

7.1.1 數據庫 13

7.1.2 源代碼命名 15

7.1.3 消息編號命名（Message） 16

7.2 代碼排版規範 17

7.2.1 Java排版規範 17

7.2.2 注释規範（comment） 18

7.2.3 HTML排版規範 21

7.3 編碼規範 21

7.3.1 Java編碼規範 21

8 數據庫設計規範 29

8.1 數據庫設計 29

8.1.1 規則 29

8.1.2 建議 29

8.2 SQL語法 29

8.2.1 規則 29

8.2.2 建議 31

8.3 程式設計 31

8.3.1 規則 31

9 前端開發 Front End Development<to be confirmed> 32

9.1 前端實現技術 32

9.2 前端靜態文件的緩存機制 32

9.3 用戶體驗（User Experience） 33

9.4 前臺編程規範 36

9.4.1 jQuery編程規範 36

9.4.2 HTML編程規範 38

9.4.3 頁面設計規範 39

10 後臺開發 Back End Development 44

10.1 項目説明 44

10.1.1 Parent項目説明 44

10.1.2 common模塊説明 45

10.1.3 系統common-web模塊 47

10.2 項目構建 49

10.2.1 API模块 49

10.2.2 Core模块 49

10.2.3 SQL模塊 51

10.2.4 WEB模块 51

10.2.5 SCH模块 51

10.2.6 其他模块 52

11 接口開發 Interface Development 54

12 排程工作開發 Schedule Job Development<David> 55

13 報表開發 ReporBt Development<David> 56

14 系統安全 Security 57

14.1 系統應用權限安全 57

14.2 編碼安全 57

14.2.1 預防SQL注入攻擊（SQL Injection） 57

14.2.2 預防iframe的點擊劫持(clickjacking) 58

14.2.3 CSRF防禦 58

14.3 數據安全 58

14.3.1 系統用戶信息 58

14.3.2 敏感訊息的保密 59

14.3.3 數據庫用戶權限 59

15 日誌 Logging<Crystal> 60

15.1 日誌工具 60

15.2 日誌分類 60

15.3 日誌分級 60

15.4 日誌記錄標準 60

16 數據校驗Data Validation<Crystal> 62

16.1 前端數據校驗 62

17 測試需求及測試案例 Testing Requirement/Test Case Setup<Crystal> 63

17.1 測試流程 63

17.2 單元測試(Unit Test) 63

17.3 系統集成測試（SIT） 63

17.4 驗收測試（UAT） 63

18 文檔 Documentation<Crystal> 64

19 源代碼管理和發佈 Source Control & Deployment<Steven> 66

19.1 源代碼管理Source Control 66

19.1.1 SVN分支/標籤建立管理規範 66

19.2 系统打包部署 66

20 系統性能 Performance Guideline (DO & NOT to DO) 67

21 系統參數定義(System Parameter Definition) 68

# 目的 Objective

本文檔針對目標項目的開發過程制定一系列規範，希望可以通過標準化系統的編程風格而獲得高質量的程序，避免出現不規範的代碼，防止錯誤的使用框架開發而引起不必要的問題。

* 安全性：增加代碼的正確性和穩定性，儘量減少各種易犯的錯誤和隱患。
* 易讀性：保證代碼的易讀性和可維護性。
* 美觀性：儘量統一項目成員的編程風格，使得代碼統一美觀。
* 易維護性：便於調試代碼，容易定位錯誤和糾錯。

# 對象 Audience

本文檔主要針對以下人員制定:

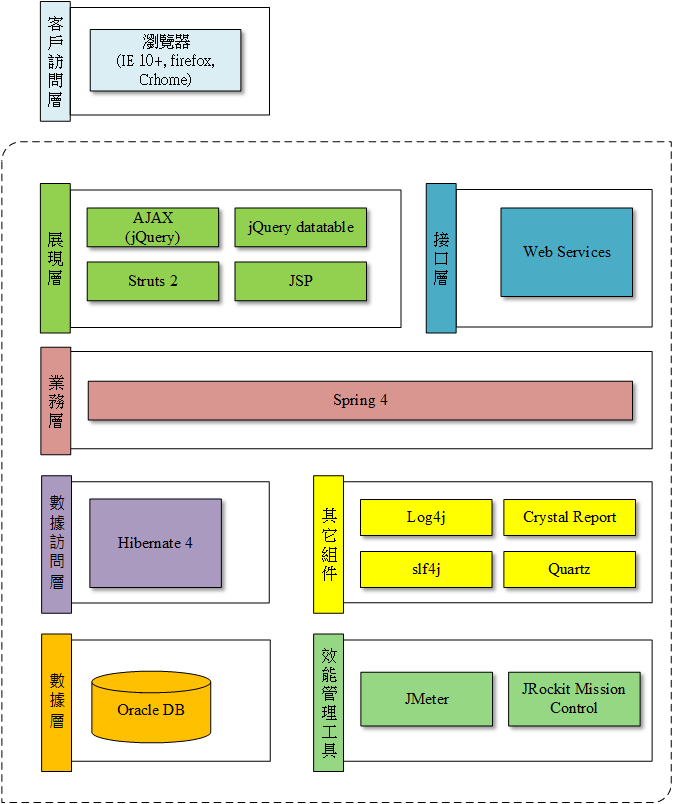
* 項目管理人員：包括技術管理和項目管理的人員。項目管理人員為本文檔中所有規範的制定者，共同決定如何根據本文檔所提出的規則和建議管理整個系統的開發過程和完善開發人員的合作方式。
* 質量保證人員：包括系統開發過程中，負責質量控制的人員。質量保證人員應由項目開始到結束階段，都監控和確保系統的開發根據本文檔制定的所有規則和建議進行。
* 開發人員：包括系統開發過程中，負責進行系統分析、設計和實現的人員。開發人員是本文檔最主要的對象，應該在進行開發工作前，閱讀并理解本文檔所列出的規範內容。
* 測試人員：包括系統開發過程中，負責進行系統SIT,UAT等測試的人員。測試人員主要參照本文檔列出的測試部份的內容，進行系統而同一的測試。

# 術語 Terminology

|  |  |
| --- | --- |
| 術語 | 解釋 |
| SA | 系統分析員(System Analyst) |
| AP | 開發人員(Application Programmer) |
| PM | 項目經理（Project Manager） |
| TL | 項目組長（Team Lead） |
| AS400 | 有IBM於1988年推出的中型機、又稱作iSeries、System i。在本文檔中泛指交通事務局於AS400上運行的陸路運輸工具管理系統。 |

# 框架設計 Framework Design

本項目主要使用Java作為開發語言，我們採用了基於MVC框架構建的SSH（Struts-Spring-Hibernate）架構，以及其他一些常用的開發框架。



## Java Library Used

* Spring-4.1.6.RELEASE
  + Spring-core
  + Spring-bean
  + Spring-context
  + Spring-aop
  + Spring-aspects
  + Spring-tx
  + Spring-web
* String MVC or Struts2-2.3.x
  + Struts2-core
  + Struts2-spring-plugin
  + Struts2-tile3-plugin
* Hibernate4.2.19.Final
  + Hibernate-core
  + Hibernate-entitymanager
  + Hibernate-validator
* Logging:
  + Log4j
  + Slf4j-api

## 前端開發框架

* jQuery 1.8
* jQuery Datatable xxx
* Bootstrap 3
* jQuery Validator

# 項目結構Project Structure

這裡的項目指的是開發時建立的eclipse項目。項目應該使用多層結構，將前臺文件和後臺文件分開。

<screen dump of the eclipse project.>

## 前臺文件結構

針對前台系統，我們會調用不同的js文件，css文件，以及圖片等資源。

1. js：建立js文件夾，存放所有js文件。應包括：

* jQuery.js
* jQuery.datatable.js
* 自定義js文件等

1. css: 建立css文件夾，存放所有css文件。應包括：

* boostrap.css
* 自定義css文件等。儘量使用同一個css文件，以便緩存。

1. images: 建立images文件夾，存放所有系統圖標，如logo,footer, icons等。
2. JSP：
3. HTML:

## 後臺結構

後臺系統的文件類別包括java class, java source, 以及xml, properties 等文件。

# 環境建設System Environment Setup<to be confirmed>

本系統主要提供三個系統環境：

* 生產環境（Production Environment）：系統上線後的運行環境。
* 測試環境（Testing Environment）：用於進行系統測試等。測試環境應儘量與生產環境一致。
* 開發環境（Development Environment）：開發環境將由開發人員自行安裝并搭建。可有多個開發環境。

所有下表提到的安裝程序將統一由SA下載并保存。開發人員須統一使用同一套安裝程序，自行搭建開發環境：

|  |  |
| --- | --- |
| 生產環境 | 版本 |
| 應用服務器 Application Server | Weblogic Cluster Enterprise Edition 11gR2 (10.3.5.0) – 2 nodes |
| jRocket | Oracle JRockit(R) (build R28.2.0-79-146777-1.6.0\_29-20111005-1807-linux-x86\_64, compiled mode) |
| 應用服務器 OS | Oracle Linux Server release 5.6 |
|  | |
| 數據庫 Database | Database RAC Enterprise Edition 11gR2 (11.2.0.4) – 4 nodes |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| 數據庫 OS | Oracle Linux Server release 6.4 |
|  | |
| SOA Suite | SOA Enterprise Edition 11.1.1.5 |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| SOA OS | Oracle Linux Server release 5.6 |

|  |  |
| --- | --- |
| 測試環境 | 版本 |
| 應用服務器 Application Server | Weblogic Cluster Enterprise Edition 11gR2 (10.3.5.0) – 2 nodes |
| jRocket | Oracle JRockit(R) (build R28.2.0-79-146777-1.6.0\_29-20111005-1807-linux-x86\_64, compiled mode) |
| 應用服務器 OS | Oracle Linux Server release 6.4 |
|  | |
| 數據庫 Database | Database RAC Enterprise Edition 11gR2 (11.2.0.4) – 2 nodes |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| 數據庫 OS | Oracle Linux Server release 6.4 |
|  | |
| SOA Suite | SOA Enterprise Edition 11.1.1.5 |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| SOA OS | Oracle Linux Server release 5.6 |

|  |  |
| --- | --- |
| 開發環境 | 版本 |
| 應用服務器 Application Server | Weblogic Enterprise Edition 11gR2 (10.3.5.0) |
| jRocket | Oracle JRockit(R) (build R28.2.0-79-146777-1.6.0\_29-20111005-1807-linux-x86\_64, compiled mode) |
| 應用服務器 OS | Oracle Linux Server release 6.4 |
|  | |
| 數據庫 Database | Database Enterprise Edition 11gR2 (11.2.0.4) |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| 數據庫 OS | Oracle Linux Server release 6.4 |
|  | |
| SOA Suite | SOA Enterprise Edition 11.1.1.5 |
| JDK | 1.6.0\_29 （Java(TM) SE Runtime Environment (build 1.6.0\_29-b11)） |
| SOA OS | Oracle Linux Server release 6.4 |

|  |  |
| --- | --- |
| 開發工具 | 版本 |
| Eclipse | Luna (64bit) |
| Maven | 3.3.3 |
| UI 設計工具 | Visio |
| 測試瀏覽器 Testing Browser | IE 10+  Firefox  Chrome  Safari |
| 報表工具CrystalReports | CrystalReports 2008 |

# 編程規範 Programming Convention

當所有代碼均保持一致的風格時，開發人員在理解其他團隊成員所編寫的代碼庫時會更加輕鬆和明瞭。

## 命名規範 Naming Convention

### 數據庫

數據庫的維護由DBA和開發人員完成。開發人員僅可針對數據結構進行維護，DBA負責創建用戶，表空間，數據文件等。

#### 數據庫命名﹣ DBA

以下內容的命名應僅有DBA可以進行，其餘開發人員若有需要創建，須先跟DBA申請創建。

|  |  |  |
| --- | --- | --- |
|  | 規則 | 例子 |
| 表空間  tablespace | * 所有字母小寫 * 用戶表空間用“**tbs\_表空間名**”表示 * 每個用戶表空間單獨創建對應的臨時表空間“**tmptbs\_表空間名**” * 每個用戶表空間單獨創建對應的回滾表空間“**undotbs\_表空間名**” | tbs\_vehicle  tmptbs\_vehicle  undotbs\_vehicle |
| 數據文件  datafile | * 以表空間名為開始，以“**.dbf**”為結尾 * 所有字母小寫，多個數據文件則在表空間名后添加“\_數字” | vehicle.dbf  vehicle\_2.dbf |
| 數據庫用戶  user/schema | * 生產環境的用戶名為“vss\_prod” * 測試環境的用戶名為“vss\_uat” * 開發環境的用戶名為“vss\_dev” |  |

#### 數據結構命名

開發人員新建或者修改數據結構時，必須統一按以下方式命名：

|  |  |  |
| --- | --- | --- |
|  | 規則 | 例子 |
| 表格 Table | 格式：**系統名\_模塊名\_表義名\_tbl**   * 所有字母大寫 * 中間以下劃線分開 * 系統名必須為“VSS” * 模塊名參考其他文檔 * 表義名可使用英文單詞的縮寫，儘量使用3個字母。 * 最長不超過30個字符 | VSS\_VEH\_USER  VSS\_AUD\_LOG |
| 列 Column | * 使用有意義的英文單詞 * 所有字母大寫 * 單詞之間使用"\_"分隔 * 最長不超過15個字符 | USER\_ID  USER\_NAME  VEL\_NAME  VEL\_COLOR |
| 索引  Index | 格式：表名\_IDX[no.]   * 最長不超過30個字符 | VSS\_USER\_TBL\_IDX1 |
| 程序包  Package | 格式：系統名\_ XXX\_PKG   * 全部大寫 * “XXX”為自定義名，使用有意義的英文單詞或縮寫 * 最長不超過30個字符 | VSS\_USER\_PKG |
| 存儲過程  Store Procedure | 格式：系統名\_ XXX\_SP   * 全部大寫 * “XXX”為自定義名，使用有意義的英文單詞或縮寫 * 最長不超過30個字符 | VSS\_CAL\_FEE\_SP |
| 序列  Sequence | 格式：表名\_SEQ   * 若多表共用一個序列，則無需使用表格名稱 * 最長不超過30個字符 | VSS\_VEH\_USER\_SEQ |
| 視圖  View | 格式：系統名\_ XXX \_VIEW   * 全部大寫 * “XXX”為自定義名，使用有意義的英文單詞或縮寫 * 最長不超過30個字符 | VSS\_USERS\_VIEW |
| 函數  Function | 格式：系統名\_ XXX \_FUNC   * 全部大寫 * “XXX”為自定義名，使用有意義的英文單詞或縮寫 * 最長不超過30個字符 | VSS\_CAL\_FEE\_FUNC |
| 觸發器  Trigger | 格式：系統名\_ XXX \_TRG   * 全部大寫 * “XXX”為自定義名，使用有意義的英文單詞或縮寫 * 最長不超過30個字符 | VSS\_TEST\_TRG |

#### 鍵類的命名規則

|  |  |  |
| --- | --- | --- |
|  | 規則 | 例子 |
| 主鍵  Primary Key | 格式：表名\_PK   * 表格**必須**定義主鍵 * 最長不超過30個字符 | VSS\_VEH\_USER\_PK |
| 外鍵  Foreign Key | 格式：表名1\_表名2\_FK[no.]   * “表名1”為主表 * “表名2”為次表 * 因兩個表之間可能有多個外鍵，因此需在FK後面添加一個數字區別。 * 最長不超過30個字符 | VSS\_VEH\_USER\_VSS\_ALERT\_FK1 |
| 唯一鍵  Unique Key | 格式：表名\_UK[no.]   * 如果有多個聯合主鍵的話，UK後添加數字來區分。 * 最長不超過30個字符 | VSS\_VEH\_USER\_UK1 |

### 源代碼命名

系統開發的所有源代碼將由所有開發人員共同創建并維護。

<steven: please see how to name these pkg according to your framework>

|  |  |  |
| --- | --- | --- |
|  | 規則 | 例子 |
| 包名Package | * mo.gov.dsat.vss為主包名，以下劃分:   + action類   + ws類   + aop類   + dao層接口   + dao層實現類   + data數據接口   + entity層類   + tranform類   + validation類   + exception類   + service層接口類   + service層實現類   + Util層公共類 * 所有字母小寫 | mo.gov.dsat.vss.aop |
| 類名 Class | * 大寫開頭，使用駝峰命名法 * 使用名詞 |  |
| 方法 Method | * 小寫開頭，使用駝峰命名法 * 使用動詞，可以包括形容詞或名詞 | searchUserName |
| 變量 Variable | * 小寫開頭，使用駝峰命名法 * 使用名詞 | vehicleName |
| 常量 Constant | * 所有字母大寫 * 使用下劃線為分隔符 | VEHICLE\_COUNT |
| 枚舉 Enum | * 大寫開頭，使用駝峰命名法 * 對象集合應全部大寫 | enum Color{  RED, BLUE, BLACK  } |
| 代碼文件 | | |
| SQL (table, sequence, index) | * dll\_[大寫實例名].sql | dll\_VSS\_VEH\_USER.sql  dll\_VSS\_VEH\_USER\_ID\_SEQ.sql  dll\_VSS\_VEH\_USER\_IDX1.sql |
| SQL (store procedure, function, package) | * proc\_[大寫儲存過程名].sql * pkg\_[大寫儲存過程名].sql * func\_[大寫儲存過程名].sql | proc\_VSS\_VEH\_USER\_LOADER\_SP.sql  func\_VSS\_VEH\_USER\_VALID\_FUNC\_.sql  pkg\_VSS\_VEH\_USER\_PKG.sql |
| JS (javascript) | * 首字母小寫 * 使用駝峰命名法 * 單詞間不使用“\_”連接   包命名   * js * js.ui UI類js   js文件命名   * [模組名字][功能實現名字][子功能名字].js | 包命名   * js * js.ui * js.ui.datatable   js文件名   * vehDatatable.js * vehDatatableFilter.js * vehServiceClient.js |
| JSP | * 首字母大寫 * 使用駝峰命名法 * 單詞之間不使用“\_”連接 | testCase.jsp |
| XML | * Struts.xml * Beans.xml * Hibernate.cfg.xml * Web.xml |  |
| Properties | * jta.properties |  |
| 項目名稱 | | |
| Project | * 所有字母爲小寫 * 單詞之間使用‘-’分隔 * [系統名]-[模塊名]-[子模塊名] | vss-veh-core |
| Source Folder | src/main/java  src/main/resources  src/test/java  src/test/resources |  |

### 消息編號命名（Message）

|  |  |  |
| --- | --- | --- |
|  | 規則 | 例子 |
| 消息ID | * 格式為“VSS-<消息類別>\_5位數字” * 前綴“VSS-”為固定格式 * 消息類別:   + I: Info   + W: Warning   + E: Error | VSS-E00001  VSS-I10001 |

## 代碼排版規範

### Java排版規範

#### 規則

1. 程序塊採用縮進風格編寫，縮進的空格數為4個，不允許使用TAB縮進。（建議修改eclipse的默認縮進風格）
2. 分界符（如“{”和“}”）應各獨佔一行，同時與引用他們的語句左對齊。在函數體的開始，類和接口的定義、以及if, for, do, while, switch, case語句中的程序或者static, synchronized等語句中都要採用如上的縮進方式。（建議修改eclipse的默認縮進風格）

示例：

|  |
| --- |
| if (a>b)  {  doStart();  } |

1. 較長的語句、表達式或參數（>80字符）要分成多行書寫，長表達式要在低優先級操作符處劃分新行，操作符放在新航之首，劃分出的新行要進行適當的縮進，是排版整齊，語句可讀。
2. 不允許把多個短語句寫在一行中，即一行只寫一條語句。
3. if, for, do, while, case, switch, default等語句自占一行，且if, for, do, while, case, switch等語句的執行語句無論多少都要加括號{}；case的執行語句中如果定義變量必須加括號{}。
4. 相對獨立的程序塊之間、變量說明之後必須加空行。

示例：

|  |
| --- |
| if (a>b)  {  doStart();  }  //此處是空行  return; |

1. 在兩個以上的關鍵字、變量、常量進行對等操作時，它們之間的操作符之前、之後或者前後要加空格；進行非對等操作時，如果是關係密切的立即操作符（如.），后不應加空格。

示例：

|  |
| --- |
| if (a == b)  {  doStart();  }  a \*= 2; |

#### 建議

1. 類屬性(attribute)和類方法(method)不要交叉放置，不同存取范围的属性或方法也尽量不要交叉放置。

格式：

|  |
| --- |
| 类定义  {  类的公有属性定义  类的保护属性定义  类的私有属性定义  类的公有方法定义  类的保护方法定义  类的私有方法定义  } |

1. 修饰词按照指定顺序书写：[访问权限][static][final]

示例:

|  |
| --- |
| public static final String str = “abc”; |

### 注释規範（comment）

#### 规则

1. 包的注释：写入一个名为package.html的HTML格式的说明文件放入包所在路径。报的注释内容包括：简述本报的作用、详细描述本报的内容、产品模块名称和版本。

说明：方便JavaDoc收集，方便对包的了解

示例：

|  |
| --- |
| mo/gov/dsat/common/package.html  <html>  <body>  <p>一句话简述  <p>详细描述  <p>产品模块名称和版本  </body>  </html> |

1. 每個文件的開頭都應該有一句版權說明，以及此文件的主要用途解釋，也方便之後生成javadoc。

|  |
| --- |
| /\*\*  \*Title:  xxxx  \* Description: xxxxxxx  \* @Copyright: Copyright (c) 2015  \* @Company: Atos  \* @author: abc  \* @version: 1.0  \*/ |

1. 类（class）和接口（interface）的註釋放在class或interface關鍵字之前，import關鍵字之後。註釋主要是一句話功能簡述與功能詳細描述。類註釋使用“/\*\* \*/”註釋方式。

|  |
| --- |
| /\*\*  \* 〈一句話功能簡述〉  \* 〈功能詳細描述〉  \* @author     [作者]（必須）  \* @see        [相關類/方法]（可選） \* @since [產品/模塊版本] （必須）  \* @deprecated  （可選）  \*/ |

示例：

|  |
| --- |
| package com.huawei.iin.logwebsmap.comm;  import java.util.\*;  /\*\*   \* LogManager 類集中控制對日誌讀寫的操作。   \* 全部為靜態變量和靜態方法，對外提供統一接口。分配對應日誌類型的讀寫器，   \* 讀取或寫入符合條件的日誌紀錄。   \* @author     developer 1   \* @see          LogIteraotor   \* @see          BasicLog   \* @since       CommonLog1.0   \*/ |

public class LogManager

1. 類屬性(成員變量)、公有和保護方法註釋：寫在類屬性、公有和保護方法上面，註釋方式為“/\*\*  \*/”

示例：

|  |
| --- |
| /\*\*   \* 註釋內容   \*/  private String vehicleName;  /\*\*   \* 註釋內容   \* /  public void write() |

1. 公有和保護方法註釋內容：列出方法的一句話功能簡述、功能詳細描述、輸入參數、輸出參數、返回值、異常等。

格式：

|  |
| --- |
| /\*\*  \* 〈一句話功能簡述〉  \* 〈功能詳細描述〉  \* @param [參數1] [參數1說明]  \* @param [參數2] [參數2說明]  \* @return [返回類型說明]  \* @exception/throws [異常類型] [異常說明]  \* @see [類、類#方法、類#成員]  \* @since [起始版本]  \* @deprecated  \*/ |

示例：

|  |
| --- |
| /\*\*  \* 根據日誌類型和時間讀取日誌。  \* 分配對應日誌類型的LogReader，指定類型、查詢時間段、條件和反复器緩衝數，  \* 讀取日誌記錄。查詢條件為null或0的表示沒有限制，反复器緩衝數為0讀不到日誌。  \* 查詢時間為左包含原則，即[startTime, endTime]。  \* @param logTypeName 日誌類型名（在配置文件中定義的）  \* @param startTime 查詢日誌的開始時間  \* @param endTime 查詢日誌的結束時間  \* @param logLevel 查詢日誌的級別  \* @param userName 查詢該用戶的日誌  \* @param bufferNum 日誌反复器緩衝記錄數  \* @return 結果集，日誌反复器  \* @since 1.2  \*/  public static LogIterator read(String logType, Date startTime, Date endTime, int logLevel, String userName, int bufferNum) |

1. 對於方法內部用throw語句拋出的異常，必須在方法的註釋中標明，對於所調用的其他方法所拋出的異常，選擇主要的在註釋中說明。對於非RuntimeException，即throws子句聲明會拋出的異常，必須在方法的註釋中標明。

說明：異常註釋用@exception或@throws表示，在JavaDoc中兩者等價，但推薦用@exception標註Runtime異常，@throws標註非Runtime異常。異常的註釋必須說明該異常的含義及什麼條件下拋出該異常。

1. 註釋應與其描述的代碼相近，對代碼的註釋應放在其上方，並與其上面的代碼用空行隔開，註釋與所描述內容進行同樣的縮排。

示例：

|  |
| --- |
| /\*  \* 註釋  \*/  public void example2( )  {  // 註釋  CodeBlock One  // 註釋  CodeBlock Two  } |

1. 對於switch語句下的case語句，必須在每個case分支結束前加上break語句。
2. 修改代碼同時修改相應的註釋，以保證註釋與代碼的一致性。不再有用的註釋要刪除。
3. 避免在註釋中使用縮寫，特別是不常用的縮寫。在使用縮寫時或之前，應對縮寫進行必要的說明。
4. 對重載父類的方法必須進行@Override聲明。

#### 建議

1. 避免在一行代碼或表達式的中間插入註釋。
2. 在代碼的功能、意圖層次上進行註釋，提供有用、額外的信息。

示例：如下註釋意義不大。

|  |
| --- |
| // 如果receiveFlag 為真  if (receiveFlag) |

而如下的註釋則給出了額外有用的信息。

|  |
| --- |
| // 如果從連結收到消息  if (receiveFlag) |

1. 對關鍵變量的定義和分支語句（條件分支、循環語句等）必須編寫註釋。
2. 註釋儘量使用英文描述。中文描述可以使用作為補充說明。
3. 方法內的單行註釋使用 //。(調試程序的時候可以方便的使用 /\* 。。。\*/ 註釋掉一長段程序)
4. 一些複雜的代碼邏輯需要說明。

示例：這裡主要是對閏年算法的說明。

|  |
| --- |
| //1. 如果​​能被4整除，是閏年；  //2. 如果能被100整除，不是閏年；  //3. 如果能被400整除，是閏年。 |

### HTML排版規範

#### 規則

1. 使用4個空格作為一個縮進層級，不允許使用TAB字符。

|  |
| --- |
| <ul>  <li>first</li>  <li>second</li>  </ul> |

#### 建議

1. 每行不得超過120個字符。過長的代碼不容易閱讀與維護。

## 編碼規範

### Java編碼規範

#### 規則

1. 數據庫操作、IO操作等需要使用結束close()的對象必須在try -catch-finally 的finally中close()，如果有多個IO對象需要close()，需要分別對每個對象的close()方法進行try-catch,防止一個IO對象關閉失敗其他IO對像都未關閉。

示例：

|  |
| --- |
| try  {  // ... ...  }  catch(I​​OException ioe)  {  //... ...  }  finally  {  try  {  out.close();  }  catch (IOException ioe)  {  //... ...  }  try  {  in.close();  }  catch (IOException ioe)  {  //... ...  }  } |

1. 系統非正常運行產生的異常捕穫後，如果不對該異常進行處理，則應該記錄日誌。

示例：

|  |
| --- |
| try  {  // ... ...  }  catch(I​​OException ioe)  {  logger.error(ioe);  } |

1. 自己拋出的異常必須要填寫詳細的描述信息。

示例：

|  |
| --- |
| throw new IOException("Writing data error! Data: " + data.toString()); |

1. 運行時異常使用RuntimeException的子類來表示，不用在可能拋出異常的方法聲明上加throws子句。非運行期異常是從Exception繼承而來的，必須在方法聲明上加throws子句。

說明：

非運行期異常是由外界運行環境決定異常拋出條件的異常，例如文件操作，可能受權限、磁盤空間大小的影響而失敗，這種異常是程序本身無法避免的，需要調用者明確考慮該異常出現時該如何處理方法，因此非運行期異常必須有throws子句標出，不標出或者調用者不捕獲該類型異常都會導致編譯失敗，從而防止程序員本身疏忽。運行期異常是程序在運行過程中本身考慮不周導致的異常，例如傳入錯誤的參數等。拋出運行期異常的目的是防止異常擴散，導致定位困難。因此在做異常體系設計時要根據錯​​誤的性質合理選擇自定義異常的繼承關係。還有一種異常是Error 繼承而來的，這種異常由虛擬機自己維護，表示發生了致命錯誤，程序無法繼續運行例如內存不足。我們自己的程序不應該捕獲這種異常，並且也不應該創建該種類型的異常。

1. 在程序中使用異常處理還是使用錯誤返回碼處理，根據是否有利於程序結構來確定，並且異常和錯誤碼不應該混合使用，推薦使用異常。

說明：

一個系統或者模塊應該統一規劃異常類型和返回碼的含義。但是不能用異常來做一般流程處理的方式，不要過多地使用異常，異常的處理效率比條件分支低，而且異常的跳轉流程難以預測。

1. 不要在finally中拋出異常

在finally block中對於捕獲的異常不應該再拋出, 否則會湮沒之前的異常, 給代碼的debug造成很大困難

1. 注意運算符的優先級，並用括號明確表達式的操作順序，避免使用默認優先級。

示例：

|  |  |
| --- | --- |
| 不易理解 | 易理解 |
| high<< 8 | low | (high<< 8) | low |
| If (a | b && a & c) | If ((a | b)&&( a & c)) |

1. 避免使用數字作為返回值，使用有意義的標識來替代。

示例：

|  |
| --- |
| 如下的程序可讀性差  if (state == 0)  {  state == 1;  … // program code  }  應改為：  private final static int TRUNK\_IDLE = 0;  private final static int TRUNK\_BUSY = 1;  private final static int TRUNK\_UNKOWN = -1;  if (state == TRUNK\_IDLE)  {  state = TRUNK\_BUSY;  … //program code  } |

1. 數組聲明的時候使用int[] index ，而不要使用int index[] 。

說明：

使用int index[] 格式使程序的可讀性較差，int [] index 表示聲明了一個int數組(int [])叫做index

1. 不要使用System.out 與System.err 進行控制台打印，應該使用工具類(如：日誌工具) 進行統一記錄或者打印。
2. 用調測開關來切換軟件的DEBUG版和正式版，而不要同時存在正式版本和DEBUG版本的不同源文件，以減少維護的難度。
3. 集合必須指定模板類型

示例：

|  |
| --- |
| Map<String,MyObject> map = new HashMap<String,MyObject>(); |

1. 一個文件不要定義兩個類(並非指內部類)。
2. 所有的數據類必須覆寫toString()、hashCode()、equals() 方法，toString()方法返回該類有意義的內容。

說明：

方便數據類的比較，父類如果實現了比較合理的toString() ，子類可以繼承不必再重寫。hashCode與equals可以使用eclipse自動生成。

示例：

|  |
| --- |
| public TestVehicle  {  private String vlName;  public String toString()  {  return “vehicle name: ” + vlName;  }  } |

1. 判斷語句不要使用“\* == true”來判斷為真。

|  |
| --- |
| 以下錯誤：  if (ok == true)  {  …  }  以下正確：  if (ok)  {  …  } |

1. 不要寫沒有必要的向上強制轉型。

示例：

|  |
| --- |
| 以下錯誤：  FileInputStream fis = new FileInputStream(f);  InputStream is = (InputStream)fis; |

1. 數組複製使用System.arraycopy(\*) 。
2. 不要使用new String()

說明：因為構造函數String()裡面的參數已經是一個String實例了。

示例：

|  |
| --- |
| 錯誤： String s = new String(“test”);  正確： String s = “test”; |

1. 不要使用循環將集合轉為數組，可以使用集合的toArray()方法。

示例：

|  |
| --- |
| ArrayList list = new ArrayList();  list.add....  String [] array = new String[list.size()];  list.toArray(array); |

1. 字符串的相加等於處理應該使用StringBuffer。

說明：大量的String相加等於處理性能消耗較多。

示例：

|  |
| --- |
| 不推薦：  String str = “”;  str += ”a”;  str += ”b”;  推薦：  StringBuffer sb = new StringBuffer();  sb.append(“aa”);  sb.append(“bb”) ;  sb.append(“cc”); |

1. 使用charAt()，不要使用startsWith()。因為startsWith()速度相對慢。
2. 使用SimpleDateFormat來控制日期的輸出格式

示例：

|  |
| --- |
| Date dateTest=new Date();  //用來顯示某一個時間的中文格式  SimpleDateFormat df = new SimpleDateFormat("yyyy年MM月dd日");  String timevalue = df.format(dateTest);  //用來顯示某一個時間的其它國家格式  DateFormat fmt=DateFormat.getDateInstance(DateFormat.MEDIUM,  new Locale("en","US"));  String newprintformat1 = fmt.format(dateTest); |

錯誤示例：

|  |
| --- |
| // 不要使用如下代码.  SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd");  String timevalue = df.format(data);  String year\_num = timevalue.substring(0, 4);  String month\_num = timevalue.substring(5, 7);  String day\_num = timevalue.substring(8, 10);  String newprintformat = year\_num + "年" + month\_num + "月"+ day\_num + "日"; |

1. 3.2.2 使用正则表达式来进行字符串的条件判断

示例：

|  |
| --- |
| //使用JDK提供的正则工具判断。  //numStr 是需要进行判断的字符串。  Pattern p=Pattern.compile("\\d\*");  Matcher m=p.matcher(numStr);  //当字符串中包含非数字字符时返回false;  boolean isNum= m.matches(); |

錯誤示例：

|  |
| --- |
| //不要使用如下代碼.  public static boolean isNumber(String tmp) {  if (tmp == null)  return false;  ArrayList list = new ArrayList();  list.add("0");  list.add("1");  list.add("2");  for (int i = 0; i < tmp.length(); i++) {  String temp = tmp.substring(i, i + 1);  if ((list.indexOf(temp) > -1) || (temp.equals("."))) {  } else {  return false;  }  }  return true;  } |

1. 使用Double.parseDouble()而不是Double.valueOf().doubleValue()。（類似有Integer.parseInt()，Long.parseLong()等）
2. Boolean.valueOf(boolean)優於new Boolean(boolean)

說明：因为new Boolean(boolean)每次会产生一个新的对象，而Boolean.valueOf(boolean)重用已有的Boolean.TRUE和Boolean.FALSE对象。

1. 不要多次初始化一個對象

示例：

|  |
| --- |
| List list = new ArrayList();  list = getUserList(); |

應該優化為：

|  |
| --- |
| List list = getUserList(); |

1. 批量更新數據時，不應該使用循環SQL的方式進行，而應該直接調用批量方法。

#### 建議

1. 記錄異常不要保存exception.getMessage()，而要記錄exception.toString()，一般可通過日誌工具記錄完整的異常堆棧信息。
2. 一個方法不應拋出太多類型的異常。

說明：

如果程序中需要分類處理，則將異常根據分類組織成繼承關係。如果確實有很多異常類型首先考慮用異常描述來區別，throws/exception子句標明的異常最好不要超過三個。

1. 異常捕獲盡量不要直接catch (Exception ex)，應該把異常細分處理。
2. 集合中的數據如果不使用了應該及時釋放，尤其是可重複使用的集合。
3. 源程序中關係較為緊密的代碼應盡可能相鄰。

示例：

|  |
| --- |
| 矩形的長與寬關係較密切，放在一起。  rect.length = 10;  rect.width = 5; |

1. 盡量使用Java 5.0新循環寫法。

說明：代碼更加簡潔。

示例：

|  |
| --- |
| ArrayList<String> list = new ArrayList<String>();  List.add…  for(String str:list)  {  …. } |

1. 使用Java 5.0枚舉來替代以前用數字與字符串的同等目的的操作。

示例：

|  |
| --- |
| public enum EnumColor  {  RED, BLUE, BLACK  }  In other function：  EnumColorcolorEnum = EnumColor. RED;  if (colorEnum == EnumColor. RED)  {  …  } |

1. interface 中定義的常量不要寫public、static、final的修飾詞，方法不要寫public修飾詞。
2. 新起一個線程，都要使用Thread.setName(“…”)設置線程名。
3. public類型的底層函數需對輸入參數進行判斷，參數不合法應該主動拋出RuntimeException。

說明：

底層函數必須保證輸入參數的正確性，再進行其他處理(防止後面的代碼拋出錯誤，減少沒有必要的後續代碼運行)。使用RuntimeException 減少了try catch的編寫，並有利於快速定於異常代碼。

示例：

|  |
| --- |
| public void doDivide(int a,int b)  {  if (b == 0)  {  throw new IllegalArgumentException(“denominator can't be zero”);  }  ...  } |

1. 盡量使用JDK自帶的API函數，不要自己寫類似功能的函數。
2. IO操作流使用有Buffer功能的class。

說明：更好的性能，沒有Buffer的輸出流頻繁IO操作，效率反倒低。

示例：

|  |
| --- |
| FileOutputStream file= new FileOutputStream("test.txt");  BufferedOutputStream out = new BufferedOutputStream(file);  for (int i = 0; i < bytes.length; i++ )  {  out.write(…);  }  out.flush(); |

1. 在循環體內減少不必要的操作

如下面代碼：

|  |
| --- |
| List list;  ...  for (int i=0; i<list.size(); i++) {  ...  } |

每次循環都需要調用方法list.size()。可優化為：

|  |
| --- |
| for (int i=0， j=list.size(); i<j; i++) {  ...  } |

1. ArrayList和HashMap的并发访问性能优于Vector和Hashtable。因為ArrayList和HashMap不支持同步(synchronized)，而Vector和Hashtable支持同步,因此在不需要同步的情況下應使用前者。

# 數據庫設計規範

## 數據庫設計

系統中所有的SQL應寫入特定的SQL類中，統一管理。

### 規則

1. 為避免full table scan，所有表格必須設有主索引鍵（Primary Key）及合適的索引（Index）。
2. 作為篩選條件的欄位必須設有合適的索引（Index），並視乎欄位的特性選擇index種類(UNIQUE, Foreign Key等)。
3. 所有會用作篩選條件的欄位不能為空值（Null），在表設計時將其欄位設有預設值。
4. 避免在約束條件(Foreign Key)使用cascade deletes、cascade updates一類的功能。
5. 對需要進行運算輸出的欄位，如TRIM()，TRUNC()，SUBSTR()及UPPER()等應儘量在INSERT及UPDATE時先行計算儲存，而不應在SELECT時進行運算，也可考慮增加欄位或功能索引（Function Index）。
6. 避免以Max()+1的方法對表序列欄位生成連續的唯一號碼，應使用序列（Sequence）來實現。
7. 不容許在數據庫的相關Schema內加入和實際應用無關的物件，例如因開發需生成的物件。
8. 應避免對資料做大量更新，減低對Undo Segment的影響及產生“ORA-01555: Snapshot Too Old”的錯誤訊息。
9. 沒有中文或葡文的欄位不應使用NCHAR()及NVARCHAR()。
10. 用以建立串聯（Join）的欄位，其資料類型（Data Type）必須一致，且不應使用須計算的值。

### 建議

1. 對一些經常使用固定條件做查詢的SQL，建議使用視圖（View），以方便管理。
2. 如需要查詢跨越Schema的物件，建議使用同義詞（Synonym）。
3. 如需要查詢另一Schema資料的視圖（View），建議將視圖（View）放在原始資料的Schema內，并授予適當的權限。
4. 如有需跨越多個Schema資料的視圖（View），建議將視圖（View）放在登入用戶的Schema內，并授予適當的權限。

## SQL語法

主要由SQL的執行計劃（Execution Plan）來判定SQL的運行效率。

### 規則

1. 在編寫SQL時避免使用任何提示（Hints）的方法。
   1. 在編寫串聯的SQL時，不可有Catesian Join, 及避免產生Full Table Scan及Full Index Scan。
2. 經常使用日期為有限篩選條件。
3. 不應使用經運算的值作為查詢的篩選條件，如UPPER()，TRIM()，TRUNC()，TO\_CHAR()，DECODE()，SUBSTR()，NVL()等，除非與一些篩選效率高的條件同時使用。
4. 避免使用以％為首字元的LIKE，如：…LIKE ‘%ABC%’，如有必要可考慮用OracleText一類的功能。
5. 避免於已經計算的視圖（View）或子搜尋SQL(Sub-Query SQL) 再進行串聯或計算的動作，如有必要可考慮用版本視圖（Materialized View）一類的功能。
6. 在編寫數據插入（Insert）語法，須標明插入數據的欄位名稱，不建議SQL語法：Insert testtbl values(1,2,3); 建議SQL語法：Insert testtbl(A,B,C) values(1,2,3)。
7. 插入/更新的值，須對應欄位的資料類型，避免出現隱式轉換（Implicit Conversions）。
8. 查詢的結果必須作排序（Order），如沒有特定的欄位作為排序，則以主索引鍵作為排序。
9. 減少使用rownum()一類功能處理翻頁。
10. 如SQL 句子需根據用戶輸入而生成，不應產生沒有篩選效果的條件（如like‘%%’..）。
11. SQL查詢語句，應該顯示獲取列，不可使用\*獲取所有列。

示例：

|  |
| --- |
| SELECT user\_id FROM user\_tbl; |

錯誤SQL:

|  |
| --- |
| SELECT \* FROM user\_tbl; |

1. 多表連接時，必須使用表的別名(Alias)來引用列。別名儘量使用較短的字串。

示例：

|  |
| --- |
| SELECT a.user\_id, v.vehicle\_name from user\_tbl u, vehicle\_tbl v WHERE u.user\_id = v.user\_id; |

1. SQL語句的所有表名、字段名全部小寫；保留字（如SELECT, FROM等）、內置函數名（DECODE等）全部大寫。統一的書寫方式，可以方便library caching判斷是否為同一個SQL語句。從而提高SQL執行的效率
2. 连接符OR、IN、AND、以及＝、<=、>=等前后加上一个空格

|  |
| --- |
| SELECT a.user\_id, v.vehicle\_name from user\_tbl u, vehicle\_tbl v WHERE u.user\_id = v.user\_id AND v.vechile\_id = 123; |

1. 尽量避免对索引列进行计算。如对索引列计算较多，请提请系统管理员建立函数索引。
2. 尽量避免不同類型數據的比較。如user\_id為數字，比較的字段也應該為數字，不要使用字符。

錯誤示例：

|  |
| --- |
| SELECT user\_name FROM user\_tbl WHERE user\_id = ‘1’; |

1. WHERE語句里有多個過濾條件時，應該將過濾記錄條數最多的條件放在最前面。

說明：ORACLE采用自下而上的顺序解析WHERE子句,根据这个原理,表之间的连接必须写在其他WHERE条件之前, 那些可以过滤掉最大数量记录的条件必须写在WHERE子句的末尾

|  |
| --- |
| 例如: (低效,执行时间156.3秒)  SELECT …  FROM EMP E  WHERE SAL > 50000  AND JOB = ‘MANAGER'  AND 25 < (SELECT COUNT(\*) FROM EMP  WHERE MGR=E.EMPNO);    (高效,执行时间10.6秒)  SELECT …  FROM EMP E  WHERE 25 < (SELECT COUNT(\*) FROM EMP  WHERE MGR=E.EMPNO)  AND SAL > 50000  AND JOB = ‘MANAGER'; |

1. 使用綁定變量（bind variable）, 不要使用字符串的表達方式。

說明：ORACLE分析完SQL後，會將SQL放入共享池(shared pool)中。如果下次遇到一樣的語句，則直接調用而不再重新分析。如果使用字符串的表達方式，會被認為是兩條不同的SQL，進行重新分析，導致效率價格低。

示例：

|  |
| --- |
| SELECT user\_name FROM user\_tbl WHERE user\_name = :1; |

錯誤示例：

|  |
| --- |
| SELECT user\_name FROM user\_tbl WHERE user\_name = ‘ABC’; |

### 建議

1. 建議使用EXISTS或NOT EXISTS的語句代替IN或NOT IN的語句。
2. 對表格或視圖（View）進行查詢時，儘量使用已索引的欄位作篩選條件。
3. 大量的排序操作影响系统性能，所以尽量减少order by和group by排序操作。

如必须使用排序操作，请遵循如下规则：

* 排序尽量建立在有索引的列上。
* 如结果集不需唯一，使用union all代替union。

1. 儘量避免使用不等於運算符(!=, <>)。（視乎到底有多少個值來決定）

說明：索引只能用于查找表中已有的数据。每当在WHERE子句中使用不等于运算符时，都将无法使用所引用的列的索引。

示例：

|  |
| --- |
| SELECT vehicle\_no FROM vehicle\_tbl WHERE vehicle\_color IN (‘red’,’blue’,’black’); |

錯誤示例：

|  |
| --- |
| SELECT vehicle\_no FROM vehicle\_tbl WHERE vehicle\_color != ‘white’; |

## 程式設計

### 規則

1. 不可對已確認之歷史記錄作物理刪除。
2. 不可將系統邏輯的正確性依賴於數據庫的響應速度。
3. 不使用數據庫功能來達成程式的目標，如利用約束鍵（Foreign Key）來清除某記錄的子記錄。
4. 不要使用物件擁有者賬戶（Object Owner）來連接數據庫，可用專用登入賬戶（如XXX\_USER）的同義詞（Synonym）或視圖（View）進行指名存取。
5. 如需以多路徑而且固定的賬戶登入數據庫（例：同時以Client/Server及Web Application連接），考慮用不同的賬戶登入來分辨功能，方便除錯及優化。
6. 避免對沒有更新的資料作重複的定時覆蓋。
7. 使用Oracle提供的驅動程式。

# 前端開發 Front End Development<to be confirmed>

系統前端的開發，基於以下幾個要求進行設計：

* 縮短反應時間(Response Time)：因原本系統為AS400，用戶已習慣其快速的反應時間。本系統應儘量滿足反應時間在1秒以內的要求，以免給用戶造成新系統比舊系統還慢的錯覺。
* 提高頁面友好（User Friendly）: 因用戶可能同時間需要查看較多的數據，因此要求頁面的顯示儘量簡潔和整齊，方便用戶快速查看需要的資料。
* 減少頁面刷新次數：採用異步調用數據的方式，儘量減少頁面整頁刷新的機率。特別是頁面上數據輸入時，避免提交時報錯的機率。
* 統一系統風格：整個系統的風格應統一。所有字體，圖標和顏色等應該簡潔易懂。
* 明確易懂的提示信息：系統錯誤，或者用戶操作錯誤時，用戶界面上應有明確的提示信息，幫助用戶糾正輸入或操作上的錯誤，或者反饋給系統管理人員。

## 前端實現技術

主要採用以下技術來實現前端開發：

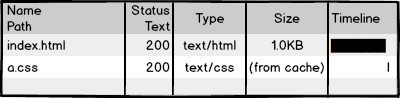
1. Bootstrap CSS Template: 採用儘量簡潔的Bootstrap CSS模板，為頁面提供一個統一的LOOK & FEEL。
2. jQuery: 採用jQuery，實現數據的異步調用，以減少頁面的刷新次數。可選擇適當的jQuery插件，幫助完成某些功能，如jQuery datables等（所選用的插件，但必須先獲得所有SA一致同意）。
3. jQuery datatable: 實現表格的分頁，排序。使用jQuery datatable的客戶端分頁和排序功能。
4. JSP：jsp代碼中，儘量使用struts2的標籤。

## 前端靜態文件的緩存機制

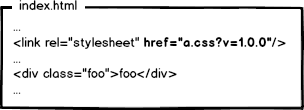
因為本系統為web-base系統，前端網頁需要多個靜態文件，如js文件，css文件，圖標圖片等資源的支持展示。為了縮短頁面反應時間的需求，我們應該儘量緩存這些靜態文件。

緩存機制：

* 強制瀏覽器使用本地緩存（cache-control/expries）。即瀏覽器不會在刷新頁面時，和服務器通信。



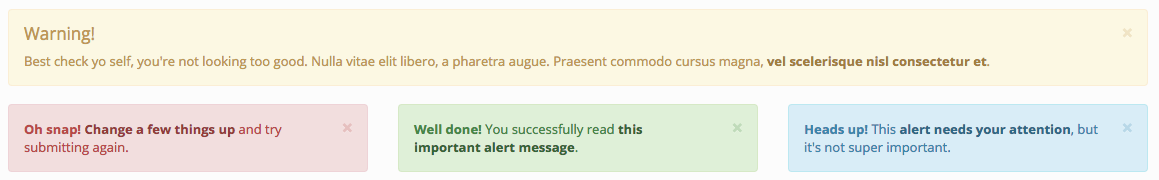
* 靜態文件的調用時加入版本號，從而在瀏覽器端更新資源。



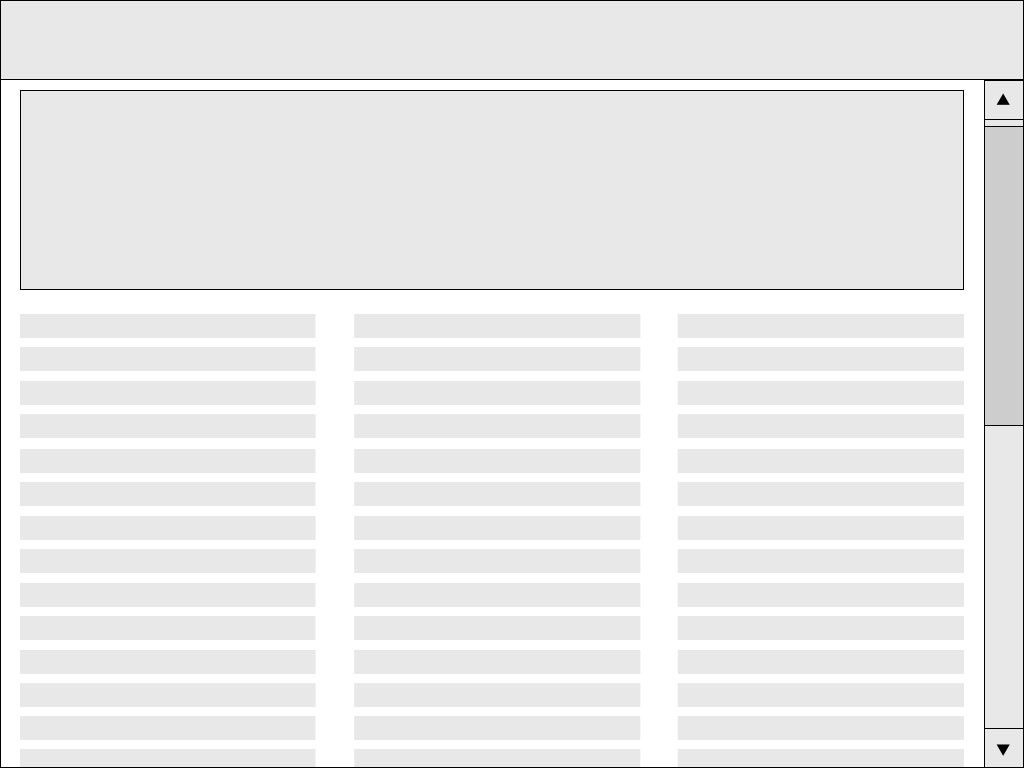
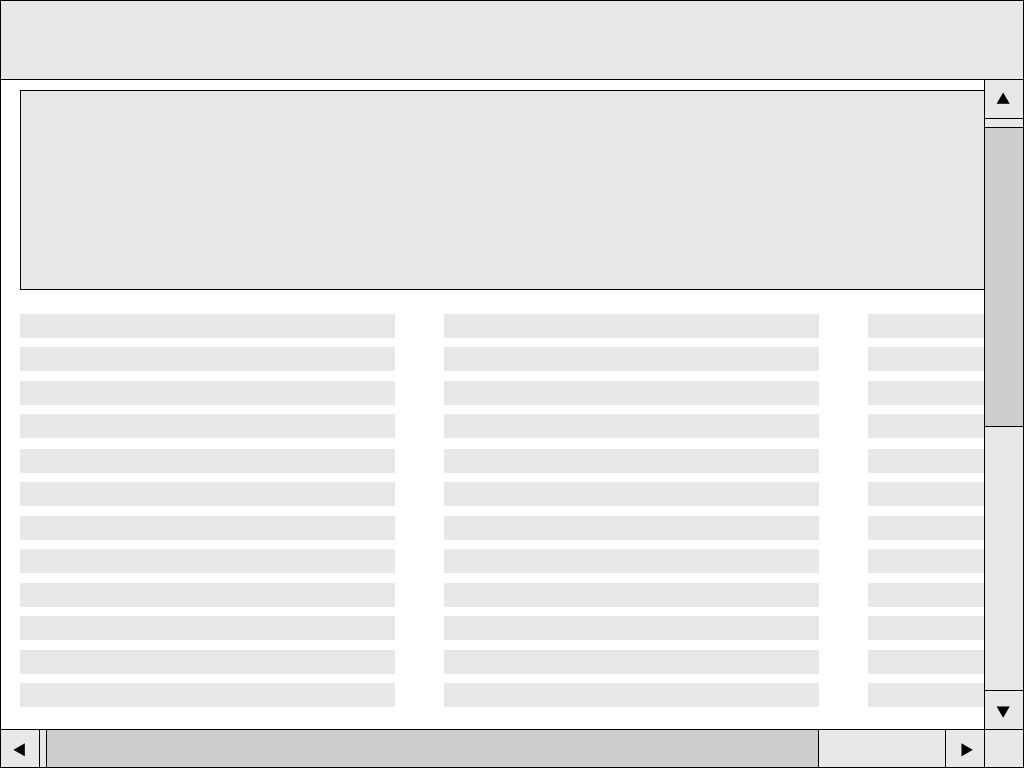
* 生產環境中的js,css文檔做GZIP壓縮，降低文件大小。
* 將樣式表（stylesheet）放在頁面的頭部，將JS文件放在尾部。

## 用戶體驗（User Experience）

1. 大量數據處理的頁面，需添加進度條（如生成報表，上載文件時）。一般來說，處理時間超過**5秒**以上，即需要添加進度條。
2. 使用不同背景顏色和消息代碼（參考“消息編號命名（Message）”）來區分系統信息的不同層度，方便用戶判斷。不能過分依賴顏色，考慮到有的用戶可能有色弱。
   1. 黃色：警示
   2. 紅色：錯誤
   3. 綠色：成功信息等
   4. 藍色：提示信息等

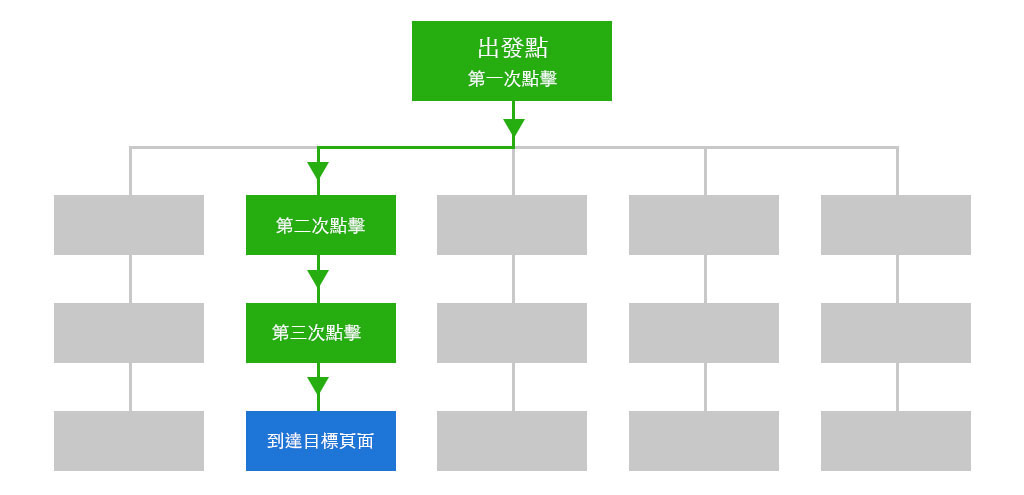


1. HTTP 400， 404， 500錯誤等，應該配置對應的錯誤導向頁面，不應該讓用戶看到此類錯誤信息。
2. 減少使用水平式滾動。



|  |  |  |  |
| --- | --- | --- | --- |
| Macintosh HD:Users:deshk:Desktop:no.png | 人比較習慣向下閱讀，如果出現直立式及水平式同時出現雙向滾動，這樣會增加瀏覧和操作難度。 | Macintosh HD:Users:deshk:Desktop:yes.png | 直立式滾動瀏覧比較舒服。 |

1. 選項和菜單設計盡量令用戶可以在**三次**點擊到達目標頁面。



1. 頁面應顯示有用內容，不顯示跟該頁主題無關和多餘的內容。
2. 在表單上，明確顯示必填和自由選擇填寫的項目。使用紅色星號(\*)代表必填選項。
3. 在表單上，針對意思難理解或不常用的欄位，使用tooltip tag的方式，添加說明字段，指導用戶輸入。



1. 提示任何錯漏的操作，並指導用戶作出修改。信息必須顯明易見，不需要scroll到特定位置。

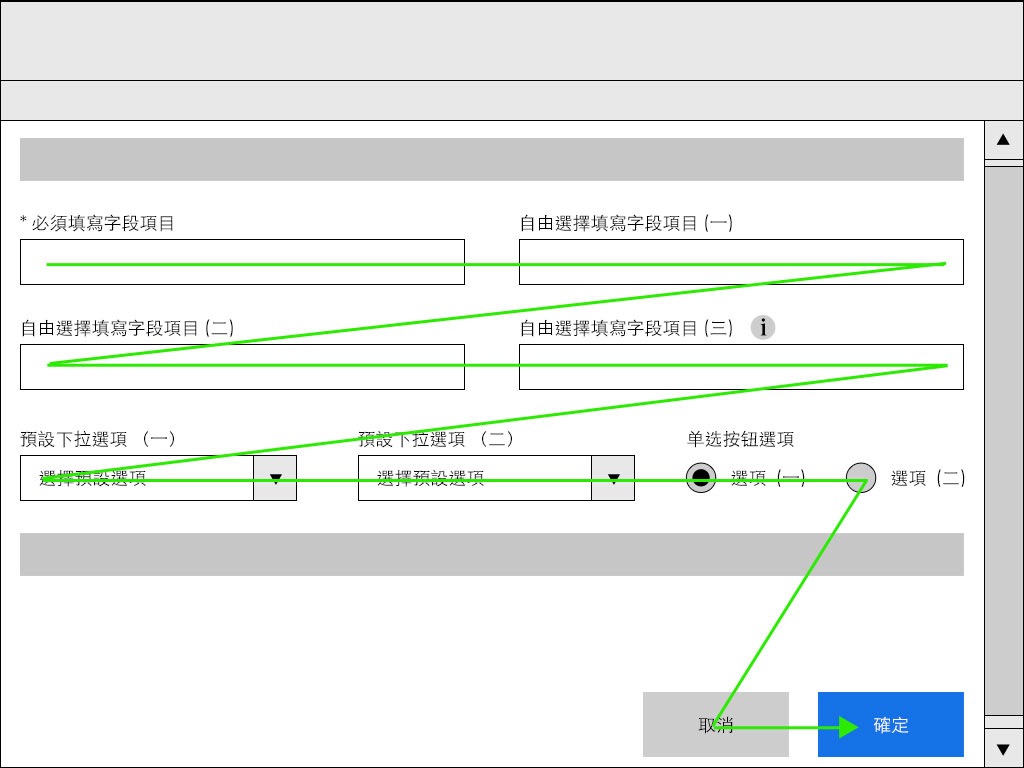


1. 在複習和大量資訊的表單頁面，可考慮把內容分為幾個步驟分類顯示，並加上合適和明確的標籤。



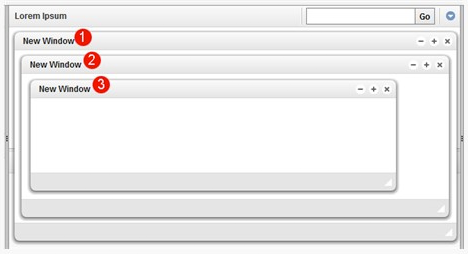


1. 在表單填寫頁面中，盡量減少不必要的瀏覧按鈕，若用戶按下任何終止表單填寫操作的按鈕或項目時，系統必須提示用戶，並讓用戶選擇繼續或終止填寫操作。
2. 在重要的操作，例如删除，終止，修改，傳輸等等，系統必須作出確認提示。防止出錯。
3. 盡量把需要一起使用的操作按鈕放在一起，減少鼠標的操作距離，減少尋找操作按鈕的時間，加快操作速度。



|  |  |  |  |
| --- | --- | --- | --- |
| Macintosh HD:Users:deshk:Desktop:no.png | 如果把傳送按鈕放到上方，用戶需要由上至下填好表單後，再滾動返回上方，尋找傳送按鈕，加長了操作距離。 | Macintosh HD:Users:deshk:Desktop:yes.png | 用戶由上至下填寫表單，然後到最下方的傳送按鈕。流程比較順暢。 |

1. 在適當和不破壞整體風格的地方使用顏色、形狀、字體大小來區分不同的操作按鈕和項目。
2. 窗口嵌套的層次不能超過3層。



1. 所有頁面顯示的物件應該上和左對齊

## 前臺編程規範

### jQuery編程規範

#### 選擇器（Selector）

1. 給需要選擇的元素添加id，使用id選擇器來選擇元素。比使用其他選擇器會快很多。

|  |
| --- |
| $('#content').hide(); |

1. 或者由id選擇器繼承來選擇多個元素

|  |
| --- |
| $('#content p').hide(); |

1. 在class前面使用tag。

jQuery中第二快的选择器就是tag选择器（如$('head')），因为它和直接来自于原生的Javascript方法getElementByTagName()。所以最好总是用tag来修饰class(并且不要忘了就近的ID).

|  |
| --- |
| var receiveNewsletter = $('#nslForm input.on'); |

jQuery中class选择器是最慢的，因为在IE浏览器下它会遍历所有的DOM节点。尽量避免使用class选择器。也不要用tag来修饰ID。下面的例子会遍历所有的div元素来查找id为'content'的那个节点：

|  |
| --- |
| var content = $('div#content');  // 非常慢，不要使用 |

1. 使用子查詢

將父對象緩存起來以備將來的使用。

|  |
| --- |
| var header = $('#header');  var menu = header.find('.menu');  // 或者  var menu = $('.menu', header); |

1. 優化選擇器以適用Sizzle的“從右至左”模型。

自版本1.3之後，jQuery採用了Sizzle庫，與之前的版本在選擇器引擎上的表現形式有很大的不同。它用“從左至右”的模型代替了“從右至左”的模型。確保最右的選擇器具體些，而左邊的選擇器選擇範圍較寬泛些：

|  |
| --- |
| var linkContacts = $('.contact-links div.side-wrapper'); |

而不要使用

|  |
| --- |
| var linkContacts = $('a.contact-links .side-wrapper'); |

1. 采用find()，而不使用上下文查找

.find()函数的确快些。但是如果一个页面有许多DOM节点时，需要来回查找时，可能需要更多时间：

|  |
| --- |
| var divs = $('.testdiv', '#pageBody');  // 2353 on Firebug 3.6  var divs = $('#pageBody').find('.testdiv');  // 2324 on Firebug 3.6 - The best time  var divs = $('#pageBody .testdiv');  // 2469 on Firebug 3.6 |

#### DOM元素

1. 將你經常用的元素緩存起來：

|  |
| --- |
| var header = $('#header');  var divs = header.find('div');  var forms = header.find('form'); |

1. 當要進行DOM插入時，將所有元素封裝成一個元素

直接的DOM操作很慢。盡可能少的去更改HTML結構。

|  |
| --- |
| var menu = '<ul id="menu">';  for (var i = 1; i < 100; i++) {  menu += '<li>' + i + '</li>';  }  menu += '</ul>';  $('#header').prepend(menu);  // 千万不要这样做：  $('#header').prepend('<ul id="menu"></ul>');  for (var i = 1; i < 100; i++) {  $('#menu').append('<li>' + i + '</li>');  } |

1. 儘管jQuery不會拋出異常，但開發者也應該檢查對象

儘管jQuery不會拋出大量的異常給用戶，但是開發者也不要依賴於此。jQuery通常會執行了一大堆沒用的函數之後才確定一個對像是否存在。所以在對一個作一系列引用之前，應先檢查一下這個對象存不存在。

1. 使用直接函數，而不要使用與與之等同的函數。

為了獲得更好的性能，你應該使用直接函數如$.ajax()，而不要使用$.get(),$.getJSON(),$.post()，因為後面的幾個將會調用$. ajax()。

1. 採用jQuery的內部函數data()來存儲狀態

不要忘了採用.data()函數來存儲信息：

|  |
| --- |
| $('#head').data('name', 'value');  // 之后在你的应用中调用:  $('#head').data('name'); |

1. 使用jQuery utility函數

不要忘了簡單實用的jQuery的utility函數。如$.isFunction(), $isArray()和$.each()。

#### 其他優化建議

1. 推遲到$(window).load

有時候採用$(window).load()比$(document).ready()更快，因為後者不等所有的DOM元素都下載完之前執行。你應該在使用它之前測試它。

1. 使用Event Delegation

當你在一個容器中有許多節點，你想對所有的節點都綁定一個事件，delegation很適合這樣的應用場景。使用Delegation，我們僅需要在父級綁定事件，然後查看哪個子節點(目標節點)觸發了事件。當你有一個很多數據的table的時候，你想對td節點設置事件，這就變得很方便。先獲得table，然後為所有的td節點設置delegation事件：

|  |
| --- |
| $("table").delegate("td", "hover", function(){  $(this).toggleClass("hover");  }); |

1. 使用ready事件的簡寫

如果你想壓縮js插件，節約每一個字節，你應該避免使用$(document).onready()

|  |
| --- |
| // 也不要使用  $(document).ready(function (){  // 代码  });  // 你可以如此简写：  $(function (){  // 代码  }); |

### HTML編程規範

1. 將HTML代碼和程序代碼分隔開，要求做到修改HTML頁面時，無須改寫程序，而修改程序代碼時亦不會影響HTML頁面效果。
2. CSS和JavaScript代碼儘量寫到外部文件中，方便充分利用瀏覽器的緩存機制。
3. 所有圖片必須設置高度(height)和寬度(width)屬性。
4. 儘量使用圖片地圖。可以緩存在瀏覽器中。
5. 儘量不使用如下代碼，以免出現錯誤，卻又無錯誤提示：

|  |
| --- |
| href=”javascript:void(0)” |

1. FORM表單的方法使用HTTP POST。(參考13.2.3CSRF防禦)
2. 因本系统采用Bootstrap的模板，因此設計界面佈局時，儘量使用div+css的方式。因为Bootstrap主要採用了div+css的佈局方式。
3. 因本系統支持的瀏覽器為IE10+, chrome, 和firefox，都可以支持HTML5。因此可以使用HTML5兼容的代碼，頭聲明應使用HTML5的W3C聲明方式，以提供更強的可擴展性：

|  |
| --- |
| <!DOCTYPE html> |

1. 所有文件必须声明UTF-8编码

|  |
| --- |
| <%@pagelanguage=*"java"*contentType=*"text/html; charset="*pageEncoding=*"utf-8"*%> |

### 頁面設計規範

#### 字體

1. 各個頁面必須使用統一的字體樣式。
2. 所有界面基准字體大小為11pt。
3. 字體風格使用css文檔定義，不能直接在HTML代碼中編寫。中文字體和葡文字體可分開定義。
4. 不考慮特殊字體（如草書等）。
5. 不要濫用粗體，斜體等，真正需要吸引人的注意或表示強調時方可使用。一個頁面中，儘量不要出現多個地方有粗體斜體字體的強調。
6. 非特殊原因，不要使用陰影等效果。

#### 尺寸單位

1. 所有界面必須使用統一的尺寸單位。如字體一律使用“pt”，對象和線條一律使用“px”作為單位。

#### 顏色

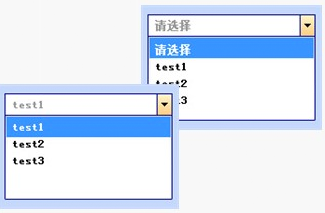
1. 界面使用的顏色儘量使用我們所選的BOOTSTRAP模板中所使用的顏色。
2. 若需添加新的顏色定義，則必須使用web-safe顏色範圍內的顏色定義。（web-safe color請參考<http://websafecolors.info/>）
3. 使用顏色時應該保持一致性，如錯誤提示信息用紅色表示，正常信息用綠色表示等。

#### 表單控件

1. 複選框（checkbox）和單選按鈕（radio button）的值，按選擇機率的高低而先後排序。
2. 複選框（checkbox）和單選按鈕（radio button）要有默認選項，并支持TAB選擇。
3. 按鈕的名字需要統一，如使用“確認”，就不要使用“確定”等。
4. 若使用圖標作為按鈕，圖標按鈕必須添加tooltip tag說明其功能。



1. 若下拉菜單不存在默認值時，則默認為“請選擇”；若有默認值，則綁定顯示默認值。



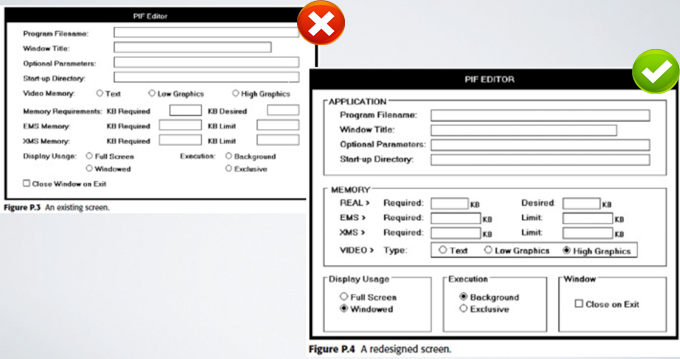
1. 可輸入/選擇框以正常顏色顯示；不可輸入/選擇框以灰色顯示



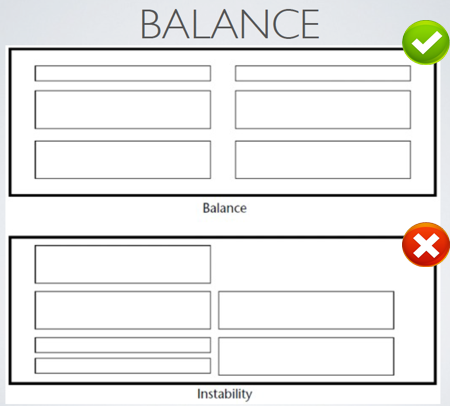
1. 對於不可輸入/選擇框，通過鼠標或鍵盤都不可讓光標定位至此控件。

#### 整體設計的可用性(Usability)

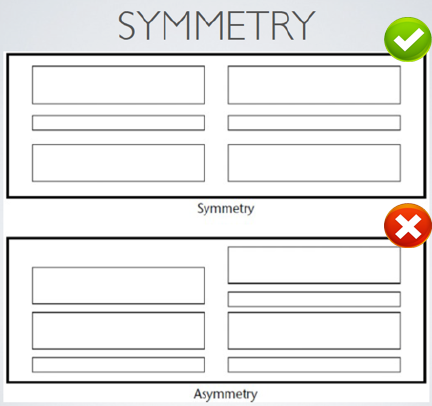
1. 若一個界面需要顯示的欄位數量過多（10個以上），界面可能會比較複雜。儘量將相關的欄位用frame來分組顯示，方便用戶快速定位欄位和使界面更加簡潔明瞭。



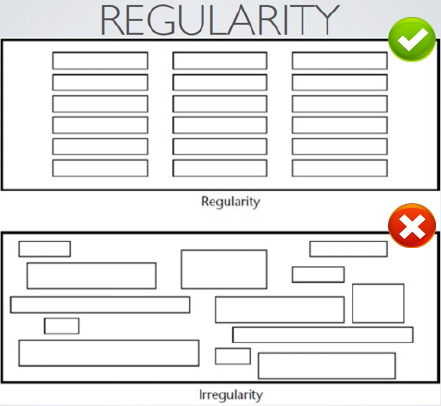
1. 根據人眼視覺的適應性來設計界面。人眼對於以下的設計會感覺到更加舒適：
   1. 平衡(Balance)：控件左右上下的平衡。



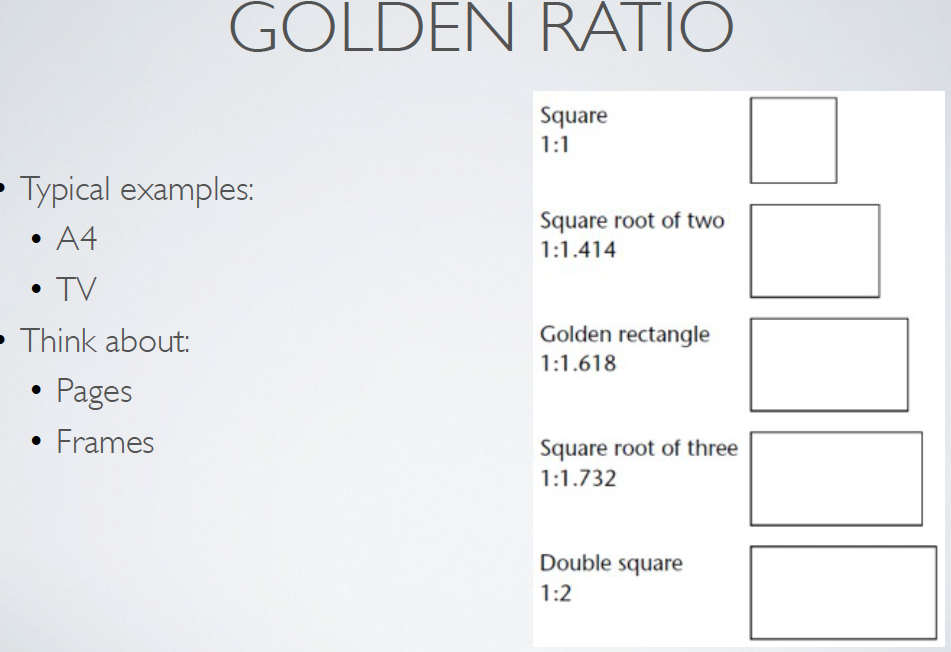
* 1. 對稱（Symmetry）：



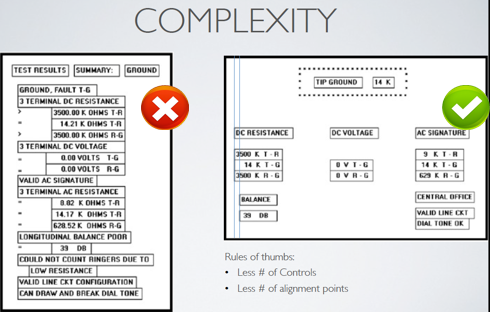
* 1. 規律性（Regularity）



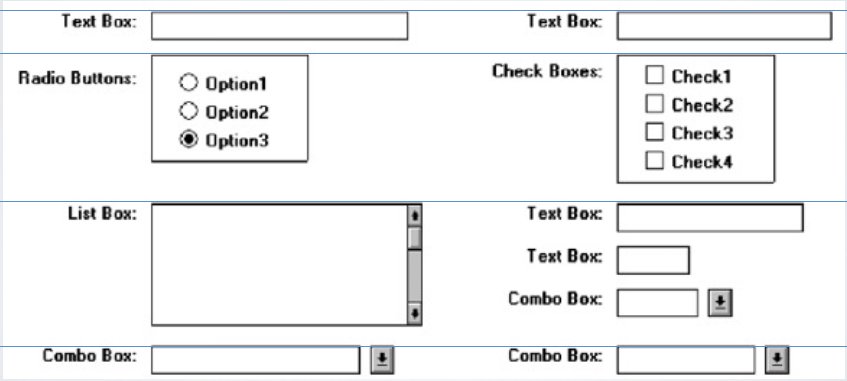
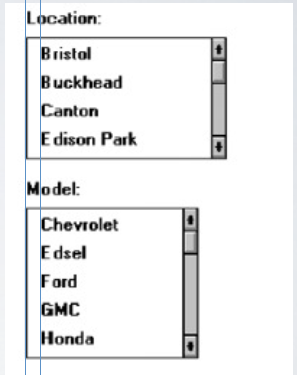
* 1. 黃金比例：最好所有的框（frame）使用以下比例，k



* 1. 複製性（Complexity）



1. 對齊:儘量靠左和靠頂對齊所有控制按鈕以及描述文字.



# 後臺開發 Back End Development

## 項目説明

### Parent項目説明

[dsat-parent] 項目不儲存任何代碼，其内容衹有一個pom.xml文件。作爲其他所有項目的父項目，[dsat-parent]使用依賴管理來管理整個開發項目中的基本包以及版本，以後如果有其他常用包以及版本的更換，衹需要修改裏面的pom.xml并且重新構建其他所有關聯項目即可。

目前，[dsat-parent]所管理的包及版本有：

|  |  |  |
| --- | --- | --- |
| # | 項目包名稱 | 版本 |
| 1 | commons-configuration | 1.10 |
| 2 | hibernate-core  hibernate-entitymanager | 3.6.10.Final |
| 3 | hibernate-validator | 5.1.3.Final |
| 4 | slf4j-api | 1.7.12 |
| 5 | aspectjrt | 1.7.2 |
| 6 | spring-core  spring-beans  spring-context  spring-jdbc  spring-orm  spring-web  spring-tx  spring-aspects | 4.1.6.RELEASE |
| 7 | struts2-core  struts2-spring-plugin  struts2-tiles3-plugin | 2.3.24 |
| 8 | jaxb-api | 2.1 |
| 9 | modelmapper | 0.5.6 |
| 10 | ojdbc6 | 10.2.0.3 |
| 11 | jackson-core-asl | 1.9.13 |
| 12 | jersey-json | 1.19 |
| 13 | jsr311-api | 1.1.1 |

### common模塊説明

|  |  |  |  |
| --- | --- | --- | --- |
| # | 包 | 類 | 描述 |
|  | root (service) | AbstractService | 事務抽象類 |
|  |  | CreationTrackable | 對象創建跟蹤抽象類 |
|  |  | UpdateTrackable | 對象更新跟蹤抽象類 |
|  | audittrail |  | 审计追踪接口 |
|  | dao | DAO/HibernateDAO | 通用HibernateDaoSupport抽象类,应用Hibernate的Session对数据库进行操作，定义通用DAO方法 |
|  |  | DaoCmd | 应用于普通固定条件的sql或hql查询，通过parameter String传入条件 |
|  |  | DynamicDaoCmd |  |
|  |  | OptionalParamDaoCmd | 应用于动态查询条件的sql或hql查询 |
|  |  | Oracle11gDialet | 應用與創建基於Dialet的查詢類 |
|  | Data | - | 基礎數據接口，統一實現不同對象的序列化及反序列化 |
|  |  | Query | 數據查詢接口 |
|  |  | Update | 數據更新接口 |
|  |  | DTO | 數據傳輸對象接口 |
|  |  | ICollectionDTO | 數據傳輸對象集接口 |
|  |  | CollecitonDTO | 數據傳輸對象集父類 |
|  |  | ClonableDTO | 可克隆數據傳輸對象抽象類 |
|  |  | ComplexQueryDTO | 複雜查詢數據傳輸對象父類，包含查詢結果總數量和過濾結果記錄數量 |
|  |  | UserStatus | 用戶狀態enum |
|  | entity | Entity | 持久層映射實抽象類，繼承UpdateTrackable接口，實現  createUser, createTime, lastUpdUser, lastUdpTime的getter與setter的實現，對應需要這幾個字段的常規表格使用 |
|  | exception | BaseException  DatabaseException  ErrorMessageHelper  PrivilegeException  RecordNotFoundException  UnknownException  UnsupportedActionException  ValidationException | 系統中常用的異常 |
|  | marshal | BooleanDeserializer  BooleanSerializer | 序列化以及逆序列化工具 |
|  | process | Process | 事務管理接口 |
|  | Util | - | 常用工具類 |
|  | validation | - | 常用校驗類 |

### 持久層接口（DAO）使用説明

HibernateDAO –通用HibernateDaoSupport抽象类,应用Hibernate的Session对数据库进行操作，定义通用DAO方法

* getSingle–针对确认是unique result set的查询使用

以下是方法的多态实现

|  |
| --- |
| **public**<T> T getSingle(DaoCmd cmd, Class<?> T, ResultTransformer transformer) |
| 与Transformer参数将result set编程DTO或者Entity对象 |
| **public**<T> T getSingle(DaoCmd cmd, Class<?> T) |
| 直接转换result set为某个Java的通用对象，比如说Date，String，只能针对单列查询结果的result set使用 |

* getList - 针对返回多行的result set的查询使用

多态方法

|  |
| --- |
| **public** <T> List<T> getList(DaoCmd cmd, Class<?> T) |
| 通用getList方法，直接将单列的result set转换成Java的通用对象的List |
| **public**<T> List<T> getList(DaoCmd cmd, Class<?> T, Integer maxResult,  Integer startFrom, ResultTransformer transformer) |
| 与Transformer参数将result set编程DTO或者Entity对象的List，并允许对查询定义起始行数以及页面大小 |
| **public**<T> List<T> getList(DaoCmd cmd, Class<?> T, Integer maxResult,  ResultTransformer transformer) |
| 与Transformer参数将result set编程DTO或者Entity对象的List，并允许对查询定义页面大小 |
| **public** <T> List<T> getList(DaoCmd cmd, Class<?> T, Integer maxResult) |
| 直接转换result set为某个Java的通用类的List，比如说Date，String，只能针对单列查询结果的result set使用，并允许对查询定义起始行数以及页面大小 |
| **public**<T> List<T> getList(DaoCmd cmd, Class<?> T, Integer maxResult,  Integer startFrom) |
| 直接转换result set为某个Java的通用类的List，比如说Date，String，只能针对单列查询结果的result set使用，并允许对查询定义页面大小 |

* execute –使用Hibernate Session直接执行hql或者sql，一般针对预先定义的关于insert、update和delete语句
* flush – 使用默认Flush Mode强制执行flush清空Session缓存，Hibernate判断对象属性有没有改变，如果被更改成为脏数据，则在一个查询语句前将更新此改动以保证数据库的同步。
* setQueryFilePath–设定查询语句文件路径
* setQueryFileURL –设定查询语句文件链接

### common項目中默認容器ApplicationContext説明

* common中的ApplicationContext主要用於数据源信息的配置，以及使用Spring Profile来管理应用程序在不同环境下（local\_dev,dev, prod）的访问对象

|  |
| --- |
| <beansprofile=*"dev,prod"*>  <beanabstract=*"true"*id=*"baseEntityManagerFactory"*  class=*"org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"*>  <propertyname=*"dataSource"*ref=*"dataSource"*/>  <propertyname=*"jpaVendorAdapter"*>  <beanclass=*"org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"*/>  </property>  <!-- Override JPA properties -->  <propertyname=*"jpaProperties"*>  <props>  <propkey=*"hibernate.dialect"*>org.hibernate.dialect.Oracle10gDialect</prop>  <propkey=*"hibernate.hbm2ddl.auto"*>validate</prop>  <propkey=*"hibernate.format\_sql"*>true</prop>  <propkey=*"hibernate.query.substitutions"*>true 'Y', false 'N'</prop>  <propkey=*"hibernate.transaction.manager\_lookup\_class"*>org.hibernate.transaction.WebSphereExtendedJTATransactionLookup</prop>  </props>  </property>  </bean>  </beans> |

### 系統common-web模塊

|  |  |  |  |
| --- | --- | --- | --- |
| # | 包 | 類 | 描述 |
|  | datatable | DatatableFilterMapping | 應用于datatable的通用類 |
|  |  | DefaultFilterValueParser |
|  |  | FilterValueParser |
|  | mock | MockUserServiceImpl | 用戶驗證服務的模擬類 |
|  | struts.action | AbstractAction | Struts Action的抽象類以及一些通用的Action |
|  |  | ErrorHandler |
|  |  | LoginAction |
|  |  | LogoutAction |
|  |  | MenuAction |
|  | ws | AbstractDatatableWS | 獲取Datatable的WebService的抽象類 |
|  |  | ExceptionWrapperJsonProvider | 基於JacksonJsonProvider的Exception類的Deserializer |
|  |  | WebServiceUtil | Web Service功能通用工具 |
|  | ws.data | AbstractDatatableDTO | 使用于Web Service的常用數據模型，這裏列舉了  Web Service返回的適用于datatable的數據傳輸對象的抽象類  用於調用WebService的datatable查詢的抽象類  通用的WebService調用結果類 |
|  |  | AbstractDatatableQuery |
|  |  | WebServiceResult |
|  | ws.exception | JsonDeserializeException | Web Service call中的常用異常 |
|  |  | … |

## 項目構建

一個基本的webapp項目需要包含

* dsat-parent
* dsat-test
* dsat-common
* dsat-common-web

當然，一個基本的項目一般也需要實現日志以及用戶驗證功能，因此，當開發人員的本地的maven庫里還沒有建立這些功能，就需要從代碼控制中取出相關的項目。

以user的項目爲例則需要取出

* dsat-user-api
* dsat-user-core

然後把dsat-user-api加入至maven的dependencies中，，這樣當前項目就可以使用user裏面的服務

### API模块

一个基本模块的对外接口，允许这个模块中的功能被其他模块调用。当使用maven去管理不同的模块时，当模块B需要调用模块A的功能，只需要放入A模块的API包即可。

* 事务接口–定义基本事务或允许其他模块调用的事务
* 数据接口–定义与事务接口对应的数据接口，基本接口包括DTO，Update和Query，也可以在这里管理Constant数据如Status，Type来为外部调用提供条件
* 数据传输对象（DTO）- 定义与每个事务对应的传输对象，作为模块输入和输出的基本对象，是模块与模块之间，表现层和核心层之间作为数据逻辑集合来用于接收和发送的简单的公用POJO
* 数据查询（Query）与数据更新（Update），在配置文件中管理事务与查询或更新的操作，事务可以根据参数中不用的查询或更新，来返回不同数据传输对象

### Core模块

配置文件ApplicationContext

* 配置aop事务代理

|  |
| --- |
| <beanid=*"cscToCspBatchServiceAspect"*  class=*"com.cathaypacific.cargo.crms.csc.csp.aop.CSCToCspBatchServiceAspect"*/> |

* 配置事务管理

|  |
| --- |
| <!-- Service -->  <beanid=*"sampleProjectService"*class=*"mo.gov.dsat.vss.veh.sampleProjectServiceImpl"*>  <propertyname=*"processes"*>  <map>  <entrykey=*"getVehData\_GetVehSummaryQuery"*value-ref=*"getVehSummaryProcess"*/>  <entrykey=*"updateVehData\_UpdateVehSummaryQuery"*value-ref=*"updateVehSummaryProcess"*/>  </map>  </property>  </bean |

* 声明DAO控制器

|  |
| --- |
| <!-- DAO -->  <beanid=*"vehSampleDao"*class=*"mo.gov.dsat.vss.veh.sampleDAO"*>  <propertyname=*"queryFileURL"*value=*"#{vehSampleURLWrapper.url}"*/>  </bean> |

* 声明查询配置文件地址

|  |
| --- |
| <beanid=*"vehSampleURLWrapper"*class=*"mo.gov.dsat.util.URLWrapper"*>  <constructor-argtype=*"java.net.URL"* value=*"classpath:vss-veh-query.xml"*/>  </bean> |

* 建立持久层的实例管理工厂

|  |
| --- |
| <beanid=*"cscExportToCspEmf"*parent=*"baseEntityManagerFactory"*>  <propertyname=*"persistenceUnitName"*value=*"crms-csc-csp"*/>  <propertyname=*"persistenceXmlLocation"*value=*"classpath:META-INF/csc-csp-persistence.xml"*/>  </bean> |

### 事務實現（ServicesImplementation）

* 建立可以使用的事务，以及每个事务的处理方法（一般定义事务的传播行为），配合TransactionManager来管理事务的超时，提交以及回滚操作
* 基本事务有：updateData，getData
* 事务入口：事务的入口有里2个，1. 更新Update或查询Query；2. 用户User。如之前所说，事务会根据不同的查询或更新来返回不同的数据传输对象（DTO），而用户则是用来进行用户权限的限制
* 事务传播行为
  + TransactionDefinition.PROPAGATION\_REQUIRED：如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务。
  + TransactionDefinition.PROPAGATION\_REQUIRES\_NEW：创建一个新的事务，如果当前存在事务，则把当前事务挂起。
  + TransactionDefinition.PROPAGATION\_SUPPORTS：如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务的方式继续运行。
  + TransactionDefinition.PROPAGATION\_NOT\_SUPPORTED：以非事务方式运行，如果当前存在事务，则把当前事务挂起。
  + TransactionDefinition.PROPAGATION\_NEVER：以非事务方式运行，如果当前存在事务，则抛出异常。
  + TransactionDefinition.PROPAGATION\_MANDATORY：如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常。
  + TransactionDefinition.PROPAGATION\_NESTED：如果当前存在事务，则创建一个事务作为当前事务的嵌套事务来运行；如果当前没有事务，则该取值等价于TransactionDefinition.PROPAGATION\_REQUIRED。

例子：

|  |
| --- |
| @ServiceMethod  @Transactional(readOnly = **true**, propagation = Propagation.*SUPPORTS*)  /\*\*  \* Entry point of retrieving the latest CRMS CSC batch reference  \* **@param** query  \* **@param** user  \* **@return**  \*/  **public** DTO getCscData(Query query, User user) {  **return** defaultProcess(query, "getCscData");  } |

### SQL模塊

此模塊應該包含所有項目所有的SQL語句，其中包括

* 普通dll的SQL，其中包括表、索引、序列、視圖的創建語句
* 用於batch處理的SQL，存放在batch文件夾中，其中包括包，儲存過程，函數和出發器的語句，或者其他PL/SQL文件
* 在部署之前需要更新進系統的初始數據，包括新的系統參數，或者對應與新增功能的對應系統設置（如計劃任務的觸發時間，系統指示器等）

### WEB模块

此模塊包含

* struts功能實現（struts implementation）

tiles.xml

struts.xml

* Web相關功能設置

web.xml

web-context.xml

* RESTFUL WebServices實現
* JSP頁面

### SCH模块

SCH模塊用於是實現程序計劃任務的功能模塊，其中包括：

* 用於大量數據處理的獨立存在的事務，這一部分事務應該符合：
  + 基於核心事務（core）所做的批量數據創建及更新
  + 需要設定計劃時間運行的核心事務（core）
  + 脫離核心事務（core）及頁面功能（web）而獨立存在的批量數據處理，如批量數據傳輸，批量報表生成，等
* 計劃任務時間應儲存在數據庫的系統參數表中，比如關於VSS的計劃任務的運行參數應儲存在VSS\_JOB\_CONTROL表中
* 視乎系統需要進行計劃任務的分類，可以再分成子模塊sch-week, sch-mon，來避免模塊重啓對不同時間間隔的計劃任務造成的影響
* 具體部署方法
  + 計劃任務的運行時間是crontab （不論用Quartz還是Commonj），爲了讓系統更加彈性得去操作運行的時間和間隔，這個數值被設計成儲存在數據庫的計劃任務控制表格中。爲了更好的進行維護，不同系統的控制表格應該分開，如VSS系統的計劃任務控制參數應儲存在VSS\_JOB\_CONTROL中
  + 使用ApplicationContext初始化crontab爲JavaBean對象，當中會使用的主要兩個類是
  + JobControlPlacehholder
  + JobControlPlaceholderConfigurer

|  |
| --- |
| <beanid=*"JobControlPlaceholderConfigurer"*class=*"mo.gov.dsat.vss.util.JobControlPlaceholderConfigurer"*>  <propertyname=*"dataSource"*ref=*"dataSource"*/>  <propertyname=*"jobControlPlaceholder"*>  <list>  <beanclass=*" mo.gov.dsat.vss.util.JobControlPlaceholder"*>  <propertyname=*"varName"*value=*"vssScheduleJob1Cron"*/>  <propertyname=*"sysCode"*value=*"VSS"*/>  <propertyname=*"paramUse"*value=*"REPORT1\_GEN"*/>  <propertyname=*"paramCode"*value=*"VSS\_REPORT1\_CRON"*/>  <propertyname=*"returnFieldName"*value=*"PARAM\_CHAR"*/>  </bean>  </list>  </property>  </bean> |

* + 在程序初始化成功之後，從數據庫讀取的crontab對象則通過注解的方式賦給scheduler

|  |
| --- |
| @Component("vssScheduleJob1Cron")  @Lazy(**false**)  **publicclass**VssReport1GenTask {  ………  ……… |

### 其他模块

* 統一處理以上模塊不能囊括的ava功能比如説對外部系統的接口，shell scripts等，視乎用戶本身需求再作定製。

# 接口開發 Interface Development

# 排程工作開發 Schedule Job Development<David>

<Waiting for David input>

# 報表開發 ReporBt Development<David>

<Waiting for David input>

# 系統安全 Security

系統安全不僅僅靠系統架構設計上的各種防火牆、殺毒軟件等來確保，在編程和系統應用設計時，也應該考慮到。

主要體現在系統安全主要包括以下幾個方面：

* 系統應用權限安全
* 編碼安全

## 系統應用權限安全

應用權限指系統內部不同角色的用戶因有不同的權限配置。本系統主要採用“role-base”的權限分配機制。即所有系統用戶應被賦予不同的系統角色，而每一個系統角色可操作或使用一系列對應的系統功能。

Role-base的應用權限機制，幫助系統管理員方便，清晰的界定每個用戶針對系統可進行的不同操作，降低用戶錯誤操作的機率。

在此基礎上，所有頁面添加權限控制代碼，即每個頁面都需要判斷當前用戶是否有對應的操作或查看的權限。

## 編碼安全

使用JAVA編寫代碼時，應考慮到一系列可能會發生的系統漏洞或系統攻擊，從而針對性的編寫代碼，提供系統安全。

### 預防SQL注入攻擊（SQL Injection）

SQL注入攻擊是針對數據庫進行攻擊的常用手段之一。當程序員編寫代碼時，應當對用戶輸入數據的合法性進行判斷，降低程序的安全隱患。

程序必須採取以下幾個步驟，確定數據輸入的安全。

#### 數據驗證（Validation）:

使用正則表達式（regular expression）驗證數據輸入的正確性。SQL注入攻擊多在特殊字符上下手腳，如“’”，“＊”，“/”，“--”等。前台編程時，儘量阻止含有特殊字符的請求。

如：

|  |  |
| --- | --- |
| 用戶名 | 只能輸入數字和字母，最大長度為20個字符。 |
| 密碼 | 可以輸入數字，字母和“﹣”，“@”；不能輸入其他字符。最大長度為20個字符。 |
| 電話 | 只能輸入數字 |

#### SQL參數的處理

<steven please input more based on our current framework>

本系統主要使用Hibernate作為數據訪問層，在書寫數據庫查詢語句時，應注意以下幾點：

* 儘量不使用拼接的語句。（以下為錯誤示例）

|  |
| --- |
| String sqlstr = "select \* from users where user\_id = '"+param+"'"; |

* 對參數名稱進行綁定：

|  |
| --- |
| Query query = session.createQuery(hql);  Query.setString(“name”, name); |

* 對參數位置進行綁定：

|  |
| --- |
| Query query=session.createQuery(hql);  query.setString(0,name1);  query.setString(1,name2);  ... |

* setParameter()方法：

|  |
| --- |
| Query query=session.createQuery(hql);  query.setParameter(“name”,name,Hibernate.STRING); |

* setProperties()方法：

|  |
| --- |
| Entity entity=new Entity();  entity.setXx(“xx”);  entity.setYy(100);  Query query= session.createQuery(hql);  query.setProperties(entity); |

### 預防iframe的點擊劫持(clickjacking)

因本系統將會與客戶方的portal做整合，以iframe的方式放置到portal中，不能屏蔽掉iframe的使用。但是使用iframe又會增加clickjacking的可能性。因此，在頁面的head中添加“X Frame options”選項。

|  |
| --- |
| <meta http-equiv="X-Frame-Options" content="SAMEORIGIN"> |

<check DSAT settings for this…….>

### CSRF防禦

* 在SESSION中綁定token。
* 在FORM表單中自動填入token字段。
* 在AJAX請求中自動添加token。
* 在服務器端對比POST提交參數的token與Session中綁定的token是否一致，以驗證CSRF攻擊。
* 對於“寫”數據的操作，FORM表單的方法使用HTTP POST。

## 數據安全

### 系統用戶信息

* 系統通過讀取客戶方提供的目錄服務器（Directory Server）,進行用戶的身份驗證。在系統中不應保存任何用戶密碼。
* 系統用戶的登錄賬號和顯示名分開。即登錄賬號不可直接在頁面上顯示出來，而顯示名不能用於登錄，但是可以公開給其他用戶看。

### 敏感訊息的保密

因系統中需要記錄車主的個人資料，如身份證明文件等。系統保存時，應該對證件的信息（如編號）進行加密處理。除此之外，還有車主的信用卡資料等。

1. 身份證明文件編號使用MD5的方法進行加密；數據庫中不能明文保存。
2. 加密和解密時，必須先驗證用戶權限，且添加審計日誌（Audit Trail）記錄操作人，操作時間等信息。

### 數據庫用戶權限

建立數據庫用戶時，不應該直接賦予DBA權限，而應該根據其所需要做的工作，分別賦予不同的角色權限。

# 日誌 Logging

系統日誌是維護人員監控系統的主要信息來源，沒有日誌的程序讓維護人員提心吊膽，太多雜亂無章的日誌也令人難以理解和查閱。

## 日誌工具

本系統採用log4j和slf4j為系統日誌開發的主要library。

## 日誌分類

主要分三大類：

* 安全類信息：記錄系統邊界交換行為和信息
* 業務類信息：記錄系統內部業務處理行為和信息
* 性能類信息：記錄系統硬件對業務處理的支撐能力

## 日誌分級

1. fatal - 嚴重的，造成服務中斷的錯誤；
2. error - 其他錯誤運行期錯誤；
3. warn - 警告信息，如程序調用了一個即將作廢的接口，接口的不當使用，運行狀態不是期望的但仍可繼續處理等；
4. info - 有意義的事件信息，如程序啟動，關閉事件，收到請求事件等；
5. debug - 調試信息，可記錄詳細的業務處理到哪一步了，以及當前的變量狀態；
6. trace - 更詳細的跟踪信息；

## 日誌記錄標準

1. 日誌應該一句一行。一行最好不超過80個字符。
2. 小心日誌會洩漏隱私。應該避免將私人信息記入日誌，受保護的內容肯定也不允許記錄。這在編寫系統框架級代碼時尤為重要，因為很難預知哪些是私人信息和受保護信息。
3. 绝对不要使用System.out.println()。
4. 避免无用的计算

先看下面的日志语句：

|  |
| --- |
| Logger.debug("Current user is " + user.toString()); |

即使在logger的DEBUG級別關閉的情況下，"Current user is " + user.toString()的計算還是會執行的。可以改成下面的代碼：

|  |
| --- |
| if (Logger. isDebugEnabled()) {  Logger.debug("Current user is " + user.toString());  } |

1. 對類中日誌工具對象logger應聲明為static。

說明：

防止重複new 出logger對象(logger指各種日誌工具類，可以是log4j，common log，jdk logger，內部API等，儘管一些logger對LogFactory工廠有一些優化，但是我們也必須防止代碼沒有必要的運行)

# 數據校驗Data Validation

及時有效的數據校驗可以給用戶良好的體驗，避免用戶輸入半天，提交的時候，才告訴用戶輸入的數據有若干的問題。用戶輸入數據時，一些基本的數據類型可以直接在頁面，通過一些腳本如正則表達式，或者js進行校驗。複雜一些的校驗可以通過ajax調用後臺服務來完成。

## 前端數據校驗

針對數據的校驗，頁面上主要使用jQuery的插件validator來完成。開發人員還應自行編寫公共的數據校驗腳本文件。

1. 系統應有一個公共的數據校驗腳本文件：validation.js
2. 輸入的字段需要驗證數據類型。如：
   1. 用戶名：只能包含字母和數字
   2. 密碼：只能包含字母和數字
   3. 日期：驗證格式是否正確
3. 儘量在鼠標焦點離開文本框時驗證。

# 測試需求及測試案例 Testing Requirement/Test Case Setup<Crystal>

測試規範針對系統開發的測試工作提供說明指導，包括單元測試，系統測試和用戶驗收測試等。

## 測試流程

## 單元測試(Unit Test)

由開發人員自行進行驗證。

## 系統集成測試（SIT）

* 系統開發完成後，進入系統測試階段。由專門負責測試的人員，對整個系統進行測試。
* 測試人員應先編寫系統測試計劃書，再根據測試計劃的步驟，逐條進行詳細的測試。
* 系統測試必須在本項目的測試環境中進行，以便與生產環境儘量保持一致。
* 測試完成後，測試人員應提供詳細的測試結果報告。

## 驗收測試（UAT）

驗收測試是每個系統的測試活動中必須進行的一道測試，是唯一需要用戶參與的測試工作。

驗收測試開始時，應確保系統代碼已封版，即在此版本上的代碼將不可再進行任何修改和編輯。

# 文檔 Documentation<Crystal>

本系統所提交的文檔交付物請參考文檔“xxxxx”。

# 源代碼管理和發佈 Source Control & Deployment<Steven>

## 源代碼管理Source Control

系統開發階段，所有源代碼將使用SVN進行管理。SVN 服務器為：

<SVN Server Info, like ip, version, users and etc.>

1. 每個開發人員對應使用自己的SVN賬號進行check-in, check-out等工作；
2. 賬號的創建統一由SA負責；
3. 每次提交(commit)代碼，必須填寫備註信息(comments)。

### SVN分支/標籤建立管理規範

## 系统打包部署

# 系統性能 Performance Guideline (DO & NOT to DO)

1. 數據庫連接使用weblogic的datasource，不要直接在程序代碼中配置。
2. 所有文件必須使用UTF-8編碼。
3. 所有註釋不可以打中文。
4. DSAT best practice
5. 儘量考慮系統的flexibility。（儘量不可hard-code任何參數，使用DB或者property file記錄）
6. 注意系統不應受客戶端的操作~統版本，語言及區域設定（尤其是日期格式等）影響運作。
7. 系統應支持中葡文，所有語言使用property file，不應該hard code。
8. 報表視乎需要加入記錄數量限制（例：不可多於一年等）。
9. 在不影響運作的情況下，考慮容許將大型交易類（Transaction）表格的歷史記錄移至另一表格來減少經常需處理的記錄數量。
10. 注意系統不應受客戶端的操作系統版本，語言及區域設定（尤其是日期格式等）影響運作。
11. 避免於修改系統參數或設定時會引起服務中斷。

# 系統參數定義(System Parameter Definition)

|  |  |
| --- | --- |
| 日期格式 | YYYY/MM/DD |
| 時間格式 | YYYY/MM/DD HH:MI:SS |
|  |  |