**Avatar Style Guide (MLD1)**

更新日期：2022/03/14

本文件提供 Avatar 在使用 Java™ 程式語言撰寫原始碼時，對程式碼標準的完整定義。當一份 Java 原始檔符合這份文件規範才能被視為 Avatar 風格。

**1. Introduction**

如同其他程式風格指南一樣，這份文件所探討的範圍不僅只有格式的問題，還包含了其他類型的公約以及程式碼風格。然而，本文件將著重於通常我們都會依循的必要規則 (hard-and-fast rules)，像那些不那麼明確地需要被執行的部份 (不論是人或工具)，這邊也會避免提供建議。

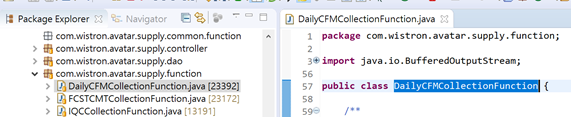
在文件中的範例程式為非規範性的 (non-normative)。這意思是說，程式碼格式應該是要有選擇性的，而不該把範例視為唯一的風格而將之當作規則強制執行，那只是要呈現出 Avatar設計的程式碼風格。

**2. Source file basics**

**2.1. File name**

源碼檔名包含其最上層級類別 (top-level class) 並區分大小寫，再加上副檔名 .java。

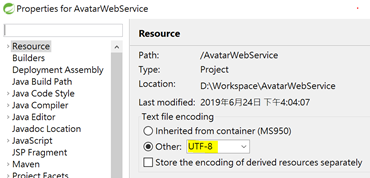
Example:

[](https://camo.githubusercontent.com/2d47bf285a6e1a4a5bd66ce25abf1b33316f4cc58a165e1c340b4f3666436b9b/68747470733a2f2f692e696d6775722e636f6d2f754767534275712e706e67)

**2.2. File encoding: UTF-8**

源碼檔的編碼格式使用 UTF-8。

Example:

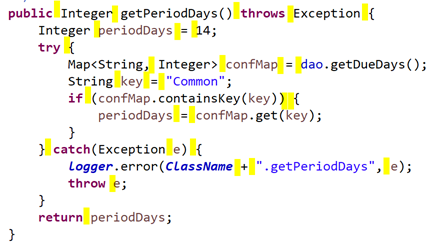
[](https://camo.githubusercontent.com/08b66a085fb85fb630de029b5a6e2646f7f63c4e203fc7e5ffa4c82cf3cac752/68747470733a2f2f692e696d6775722e636f6d2f5176736c5a6a642e706e67)

**2.3. Special characters**

**2.3.1. Whitespace characters**

除了每行的結束符號， ASCII 空格符號 (0x20) 是唯一存在於源碼檔案中的空白符號，目的是為了閱讀上的舒適性。這意味著：所有的字串以及符號間皆要使用空白。

Example:

[](https://camo.githubusercontent.com/76229774339ee88f0963491d70427fd6a1dd2194aa221f5fed2fe06582f97f0a/68747470733a2f2f692e696d6775722e636f6d2f6d546e74486a702e706e67)

**2.3.2. Special escape sequences**

所有可搭配跳脫符號的字符 (\b、\t、\n、\f、\r、\”、\’ 以及 \ ) 都直接使用，而不必要轉換成相應的八進位 (如：\012) 或是 Unicode (如：\u000a) 使用之。

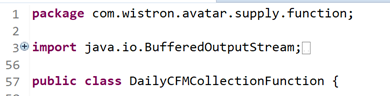
**3. Source file structure**

一個源碼檔案裡需要包括 (有順序性)：

1. Package 敘述
2. Import 敘述區
3. 獨立的最高層級類別 (top-level class)

以上各個區段之間要使用一空行 (exactly one blank line) 隔開。

Example:

[](https://camo.githubusercontent.com/38f6e40fd785864b0c660f5b59a54f3a588fdab9c9b58a92b72b63e48979b958/68747470733a2f2f692e696d6775722e636f6d2f34526c656170722e706e67)

**3.1. Package statement**

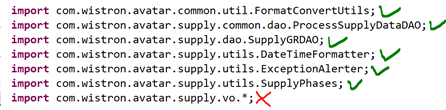
Package 敘述不換行 (not line-wrapped)。 意即每一行的最大字數限制 (請見 4.4. Column limit: 100) 不適用於此。

**3.2. Import statements**

**3.2.1. No wildcard imports**

Import 敘述區裡的每一個 import，皆不要使用萬用符號 (\*) 修飾子。

Example:

[](https://camo.githubusercontent.com/812c8249761be28441ea5660a12b087982765141961e528a53ab9c5346f2fd07/68747470733a2f2f692e696d6775722e636f6d2f534d5973464d6e2e706e67)

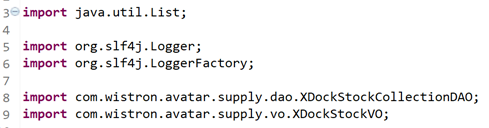
**3.2.2. No line-wrapping**

Import 敘述區裡的每一個 import 不換行 (not line-wrapped)。 意即每一行的最大字數限制 請見 4.4. Column limit: 100 不適用於此。

**3.2.3. Spacing**

Import 敘述區如有兩種以上不同的library import，可用一空行做為分群呈現，分群裡每行 import 敘述皆不要再空行隔開。

Example:

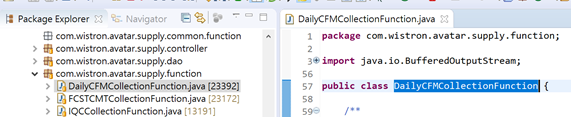
[](https://camo.githubusercontent.com/65f1b9fc6b6351470fc4bd1bed14fd14b00f05400626e55d9476c3a2875fff9d/68747470733a2f2f692e696d6775722e636f6d2f6b6c33336b576b2e706e67)

**3.3. Class declaration**

**3.3.1. Exactly one top-level class declaration**

每個最上層級類別都應該存在於自己的源碼檔案中。

Example:

[](https://camo.githubusercontent.com/b57b7e3a28b1e36b4e1397866c31f5dd7bf00899ab99d03b45fc865f0004fc08/68747470733a2f2f692e696d6775722e636f6d2f696358493875732e706e67)

**3.3.2. Ordering of class contents**

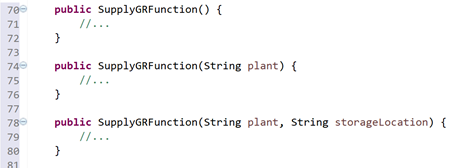
類別成員的排序對於能否很好的識別有極大的影響，但實作上並不存在唯一合適的通則。甚至是在不同的類別裡，其中的成員排序就都不同。

重點在於每個類別的維護者都要能夠按某個邏輯去排序其中的成員，並在被詢問時有辦法解釋之。舉例來說，在新增一個新的函式，我們就習慣的將之加在該類別的最末處，而這樣是不是就造就了一個「按照時間新增」的排序邏輯了呢。

**3.3.2.1. Overloads: never split**

當類別中有多個建構式 (constructors) 或是多個同名的函式，就讓他們依序的放在一起，中間就不要再穿插進不同成員來造成干擾。

Example:

[](https://camo.githubusercontent.com/54de20afa6012ae026be5c27f3b8bccf0651dfe486cf5ca2d1263b12a629c0f4/68747470733a2f2f692e696d6775722e636f6d2f3642316f3337572e706e67)

**4. Formatting**

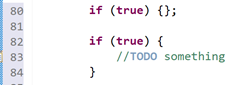
術語說明：類別中的建構式 (constructor) 或是函式 (method) 該以區塊結構 (block-like construct) 呈現。特別是像 4.6.3.1. 裡陣列的初始值，所有陣列的初始值只要是以區塊結構呈現，並不限制其格式都要一樣。

**4.1. Braces**

**4.1.1. Braces are used where optional**

在 if、 else、 for、 do 以及 while 這些敘述裡的程式碼，就算是空的，或是僅有一行，也都要使用大括號。

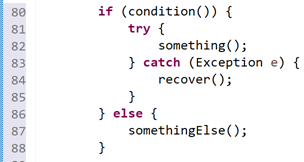
Example:

[](https://camo.githubusercontent.com/c5b80b0414dd213f735a7a164bd7f5bc40add98c00c2be4394f4ee5eaaa1ee9a/68747470733a2f2f692e696d6775722e636f6d2f396c4253394b682e706e67)

**4.1.2. Nonempty blocks: K & R style**

在內容為非空的區塊結構中，大括號使用參照Kernighan and Ritchie 風格 (Egyptian brackets)，如下 左大括號前不要換行 在左大括號後換行，並在右大括號前換行 當右大括號是用在一個敘述語法、函式、建構函式或是類別之後，就要換行。若是像其後接續的是 else 或是 catch 就不用。

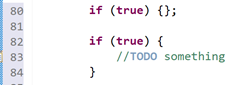
Example:

[](https://camo.githubusercontent.com/12b0e58356954b5449d2844e1776a950db5bc5ae3607e013e9a75f25e72cb473/68747470733a2f2f692e696d6775722e636f6d2f44776e795a71582e706e67)

**4.1.3. Empty blocks: may be concise**

當一個區塊結構為空時，可以將左右括號寫在一起，不需要在其中加入空白或是換行 ({}) 除非他是多區塊敘述語法的其中一部份(如 if/else-if/else 或是 try/catch/finally )。

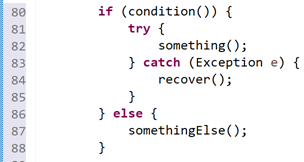
Example:

[](https://camo.githubusercontent.com/ece9cdc799025850307eeb51e0ec6a24a4b01b593f9dca5950ed1167f4e01f5f/68747470733a2f2f692e696d6775722e636f6d2f346339577054702e706e67)

**4.2. Block indentation: +1 tab**

每新增一個區塊結構時，便要在其開始就增加一個tab。當區塊結束時，其縮排則返回到和前一層的縮排一致。而縮排的層級適用於其間的程式碼與註解。

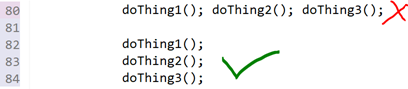
Example:

[](https://camo.githubusercontent.com/e405c7d82f7e986ca9904aafa92459855dcef49255f094c1d16e1514567aeb23/68747470733a2f2f692e696d6775722e636f6d2f4863756e676a512e706e67)

**4.3. One statement per line**

每行的敘述都要換行。

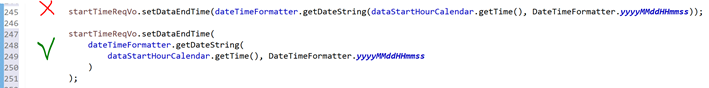
Example:

[](https://camo.githubusercontent.com/8c0f5b3cbe5f83ce052b6582e0fa62e77968a600bb2d63b955b2a25d67de1cc7/68747470733a2f2f692e696d6775722e636f6d2f614e48326243792e706e67)

**4.4. Column limit: 100**

每個專案皆可選擇自己要以每列 100 個字元為限制。除非是遇到下述例外，否則每行的字數當達到限制時就需要換行，換行的說明請見 4.5 Line-wrapping。

Example:

[](https://camo.githubusercontent.com/9b8911cf0491a540fe2135fc2b675b6fb7b5c9caf045f64ba666c160efc93ac8/68747470733a2f2f692e696d6775722e636f6d2f6e573659766b4f2e706e67)

例外：

1. 不可能滿足在一列的字數限制的條件 (比方說，在 Javadoc 裡很長 URL)。
2. package 以及 import 敘述 (請見 3.1. Package statement 以及 3.2. Import statements)。
3. 註解中，可能會被複製貼上的 shell 指令

**4.5. Line-wrapping**

術語說明：當程式碼可能在單行裡超過每列限制，而將之拆開到多行中，這個動作稱之為換行 (Line-wrapping)。

並沒有一種方程式可以完整、明確的告訴我們在所有情況中該如何換行。而有常有多種合理的方式去對同一段程式碼換行。

:::info Note: 雖然換行的原因是為了避免超出行限制，但即使是實際符合行限制的代碼也可能由作者自行決定是否換行。 :::

:::warning Tip: 精簡函式或是區域變數名稱長度，也許可以解決這種需要被換行的情形。 :::

**4.5.1. Indent continuation lines at least +1 tab**

每個接續在斷行後的第一行開始，都要比原行加上一個 tab 來縮排。

**4.6. Specific constructs**

**4.6.1. Enum classes**

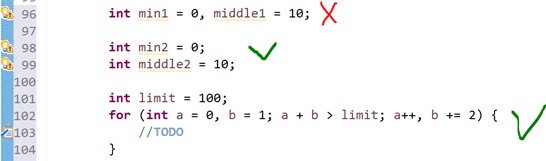
建議可直接使用Constants取代Enum。

**4.6.2. Variable declarations**

**4.6.2.1. One variable per declaration**

每個變數在宣告 (屬性 (field) 或是區域變數 (local)) 時，僅宣告一個變數，也就是說不要寫成 int a, b; 。 例外: 多變數在同一行的宣告可用於 for loop。

Example:

[](https://camo.githubusercontent.com/063fb17ca6ca6f8db4524fa73fdeeb3dd97b28d418db905b6b179967297c7f04/68747470733a2f2f692e696d6775722e636f6d2f427338624e4f382e706e67)

**4.6.2.2. Declared when needed**

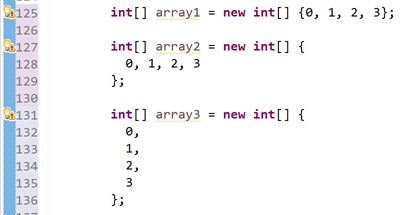
區域變數不要習慣在其區塊或是類區塊結構的一開始就全部宣告了。而是要在該區域變數第一次被使用前才宣告，盡量將他的範圍最小化。宣告區域變數時通常就會有初始值，或是在其宣告後立刻被初始化。

**4.6.3. Arrays**

**4.6.3.1. Array initializers: can be "block-like"**

所有陣列在初始化時，只要格式為「類區塊結構 (block-like construct)」皆可。

Example:

[](https://camo.githubusercontent.com/0ac0186cb897cb168778e8765fa86490fa78fd26ec3430bb60b987a6bfe7c6be/68747470733a2f2f692e696d6775722e636f6d2f63756b546176772e706e67)

**4.6.3.2. No C-style array declarations**

中括號是型別的一部份，而非變數的一部份： Srting[] args 而不是 String args[]。

**4.6.4. Switch statements**

術語說明：在 switch 敘述區塊的括號中，有一或多個敘述群組。每個敘述群組包含一個或多個 switch 標籤 (像是 case FOO: 或是 default: )，其後接續著一或多個程式敘述。

**4.6.4.1. Indentation**

和其他區塊一樣， switch 區塊的內容縮排都是 +1 個 tab。 在 switch 標籤後，每新的一行，若是該區塊的開始，就 +1 個 tab做為縮排層級的增加。並在下一個標籤退回到上一個縮排層級，以表示上一個區塊已經結束。

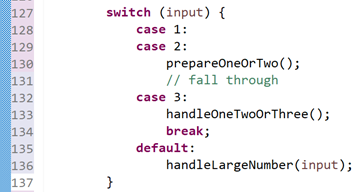
**4.6.4.2. Fall-through: commented**

在 switch 區塊中，每個敘述群組都會有個終止點 (像是 break 、 continue 、 return 或是拋出異常)，或是標上註釋以表將會或是可能繼續往下一個敘述群組執行。該註解只要足以表達出 fall-through 即可 (通常都是用 // fall through )。這特殊的註解不需要出現在 switch 區塊中的最後一個敘述。

**4.6.4.3. The default case is present**

每個 switch 敘述的群組中都一定包含 default ，就算裡面沒有程式碼。

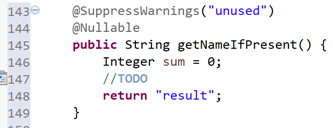
Example:

[](https://camo.githubusercontent.com/68aeb46e792a79021183c0b86583134ad3d099b15600d13b6479719ea1567179/68747470733a2f2f692e696d6775722e636f6d2f334169445750332e706e67)

**4.6.5. Annotations**

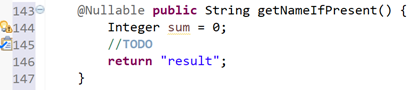
應用標注 (annotation) 的類別、函式與建構函式緊接於其文件區塊之後，每個標注皆自己獨立於一行 (意即一行一個標注) 。而這幾行的斷行並不需依照換行建議 (4.5. Line-wrapping，所以，縮排層級並不會增加。例如：

Example:

[](https://camo.githubusercontent.com/f14f4d9a23a58bc7c5f032ae3c972fbaca6382ca094c514478760b986c18b1f6/68747470733a2f2f692e696d6775722e636f6d2f3337526b3766642e706e67)

例外： 單個無參數的註解可以和其屬名的第一行放在一起。

Example:

[](https://camo.githubusercontent.com/120417d48ac49744678909e58a302b6e86e466364e9881be812f95ecdc2c7e74/68747470733a2f2f692e696d6775722e636f6d2f6e564f347362312e706e67)

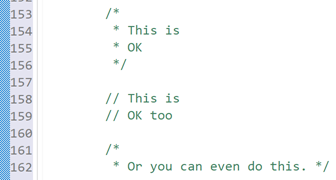
**4.6.6. Comments**

本節著重在implementation註解。

**4.6.6.1. Block comment style**

註解區塊的縮排，和其接連的程式碼同一層級。可用 / ... / 或 // ... 。若是這種註解風格 / ... / 有多行時，其子行的起始必需有 ，而該星號需對齊上一行的 。

Example:

[](https://camo.githubusercontent.com/b6133d7f9a952db236cd412389a27d5b3ac62d604123ae1468d252db7fa09c75/68747470733a2f2f692e696d6775722e636f6d2f53476f436542552e706e67)

**4.6.6.2. Documentation Comments**

採取Service的implement 及 Util需添加註解。

Sample：

一張含有 文字 的圖片

自動產生的描述

**4.6.7. Modifiers**

類別與成員的修飾詞，其呈現順序就依 Java 語言的規範：

public protected private abstract default static final transient volatile synchronized native strictfp`

**4.6.8. Numeric literals**

long 於整數型別後加上後綴大寫字母 L，千萬別用小寫字母 (避免和數字 1 )。例如，3000000000L 而不要寫成 30000000000l。

**4.6.9. DSL and SQL command**

使用 StringBuilder 取代 String 來組成 DSL 和 SQL，在呈現上會更清楚且效能更好。

**4.7. Write short methods**

好的函式應該夠短，當一個函式寫到 40 行，就容易造成理解上的困難，如有很長的函式，嘗試將其分成多個部分。

例外: 資料 mapping 的函式允許超過 40 行。

**5. Naming**

**5.1. Rules common to all identifiers**

名稱定義僅使用ASCII字母和數字，並且在下面提到的少數情況下使用下底線。

在 Avatar Style中，不使用特殊前綴或後綴。

例如，這些名稱不是 Avatar Style：name，mName，sname 和 kName。

**5.2. Rules by identifier type**

**5.2.1. Package names**

Package 名稱全部小寫，連續單字直接寫在一起(不用下底線())。例如， com.example.deepspace，不要用 com.example.deepSpace 或 com.example.deep\_space 。

此外 Package 應依照 com.wistron.avatar.{專案名稱}.{子專案名稱 (如果需要)}.{模組或功能名稱}.{責任分類} 命名。

Example:

com.wistron.avatar.supply.controller

com.wistron.avatar.supply.function

com.wistron.avatar.supply.dao

com.wistron.avatar.supply.vo

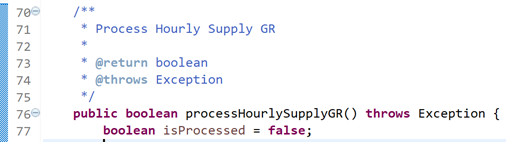
**5.2.2. Class names**

類別名稱採用大寫開始的駝峰命名法 (UpperCamelCase)。 類別名稱為名詞或是名詞片語。例如， Character 或 ImmutableList，介面 (interface) 也使用名詞或是名詞片語 (如： List)，但有時候會用形容詞或是形容詞片語取而代之 (如： Readable)。 測試類別的名稱以他們正在測試的類別名稱開頭，以Test結尾。 例如，HashTest 或 HashIntegrationTest。

**5.2.3. Method names**

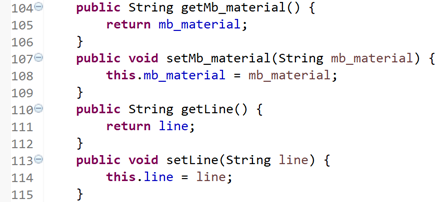
函式名稱採用小寫開始的駝峰命名法 (lowerCamelCase)。 函式名稱通常都是動詞或是動詞片語。例如， sendMessage 或是 stop 。 下底線可以做為 JUnit 的測試函式名稱與邏輯元件的分隔。一個典型的模式 test<MethodUnderTest><state>，範例： testPopemptyStack。這邊並沒有唯一的正確方式去命名測試函式。

Example:

[](https://camo.githubusercontent.com/9be68f6fe7a59674c9a75ab7fe7561bae88f8c37fa2ff47d237954102a3e631c/68747470733a2f2f692e696d6775722e636f6d2f696179744e734d2e706e67)

例外: VO (Value Object) 的 getter 和 setter, 會使用小寫開始的駝峰命名法搭配 Snake case 來命名，原因是現行 Avatar 存取的 ElastricSearch index, Oracle table, Redis 等欄位名稱, 皆使用小寫的 Snake case, 因此便於資料 mapping, VO 對應的 getter 和 setter 函式, 皆使用 Snake case。

Example:

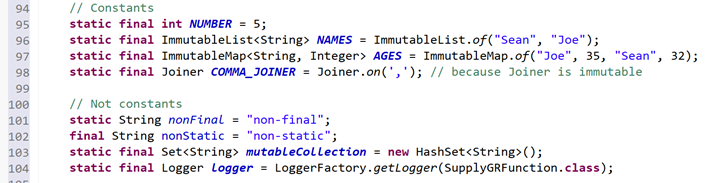
[](https://camo.githubusercontent.com/c4b4b24098fbc3aa1e951dbfc3483cc6ff3d792341f45c3957bdab69afb13c07/68747470733a2f2f692e696d6775722e636f6d2f39734952594b342e706e67)

**5.2.4. Constant names**

常數名稱採用 CONSTANTCASE ，全部採用大寫字母，使用下底線分隔。但究竟什麼是常數呢？

所以有常數的屬性 (Field) 皆是 static final， 但並非所有 static final 屬性的變數皆為常數。在確定變數是為常數前，需要先考慮他是否真的像一個常數。舉例來說，當所有在實作的觀察階段會改變時，那這個變數幾乎就可以肯定不是一個常數。而通常只是打算永不改變狀態是不夠的。

Examples:

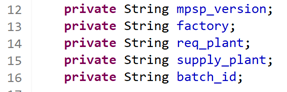
[](https://camo.githubusercontent.com/0ecb00384eb251cc09599b52bcd2e6b55fd9d6d5fbf27c4387efd436e90b53f7/68747470733a2f2f692e696d6775722e636f6d2f554d6f454a78712e706e67)

**5.2.5. Non-constant field names**

非常數屬性的名稱採用小寫開頭的駝峰命名法 (lowerCamelCase)。 名稱通常為名詞或是名詞片語。

例外: VO (Value Object) 使用 Snake case 來命名，現行 Avatar 存取的 ElastricSearch index, Oracle table, Redis 等欄位名稱, 皆使用小寫的 Snake case, 以便於資料 mapping, VO 的非常數命名, 皆使用 Snake case。

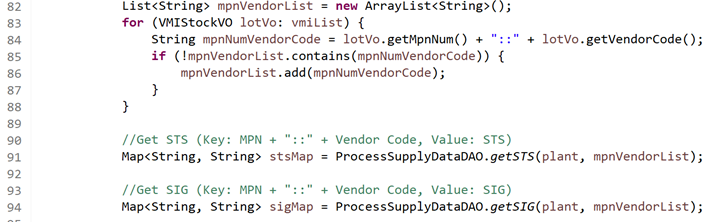
Example:

[](https://camo.githubusercontent.com/e572963bb0e4f54e8ec605013546df41451f6595dea1eefa9dcab87f89682a92/68747470733a2f2f692e696d6775722e636f6d2f423168453659672e706e67)

**5.2.6. Local variable names**

區域變數名稱採用採用小寫開頭的駝峰命名法 (lowerCamelCase)。 區域變數是不可為常數的，當然，也不該使用常數變數的風格。

Example:

[](https://camo.githubusercontent.com/be8c4dcf6bf1f94843e226bb981364467f4b2bfc31452e1ec6a1112b1bec7316/68747470733a2f2f692e696d6775722e636f6d2f6261797657455a2e706e67)

**5.3. Camel case: defined**

英文詞彙有時並非只有一種合理的駝峰命名表示方式，也有像「IPv6」或是「iOS」這樣的縮寫或是不尋常的表示法。為了改善使其規律，Avatar Style 將使用下方 (幾乎) 確定的方法。 以散文格式 (prose form) 為名稱的開頭：

1. 字詞皆改為 ASCII 碼並移除所有單引號，例如，「Müller’s algorithm」可以改變為「Muellers algorithm」。
2. 依上述步驟的結果，再依其中的空白以及其餘的符號 (通為會連字符號) 做為拆分點，拆成逐一的單字。 建議：若所有字都已經有其慣用的駝峰命名用法，仍是將其拆開(例：「AdWords」變成「ad words」。注意，像「iOS」這個並不是一個駝峰命名的形式，這個建議就不適用於這樣的例子。
3. 現在，將每個字母全部變成小寫(包含縮寫)，接著，只要將第一個字母改為大寫： 每個單字都改，為大寫開頭的駝峰 (upper camel case) 命名。 每個單字除了第一個單字不改，仍用小寫開頭的駝峰 (lower camel case) 命名。
4. 最後，將所有單字連成一個名稱定義。

需要注意的是，這邊的大小寫幾乎是已經無視原來的單字。

Examples:

| **Prose form** | **Correct** | **Incorrect** |
| --- | --- | --- |
| XML HTTP request | XmlHttpRequest |  |
| new customer ID | newCustomerId |  |
| inner stopwatch | innerStopwatch |  |
| supports IPv6 on iOS? | supportsIpv6OnIos |  |
| YouTube importer | YouTubeImporter |  |

:::info Note: 有些字在英語中，有無帶著連字符號都沒有錯，舉例來說「nonempty」以及「non-empty」二者皆對，所以方法若是命名成 checkNonempty 以及 checkNonEmpty 都是正確的。 :::

**6. Programming practices**

**6.1. @Override: always used**

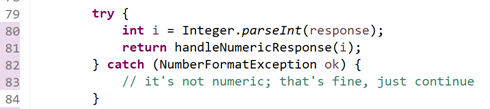
當一個被合法的標註了 @Override 的方法 (method)，他一定是覆寫了其父類別 (superclass) 的方式、實作了介面方法 (interface mthod) 以及介面中重新指定了其父介面 (superinterface) 的方法。

例外： @Override 當可以被省略時，代表其父類別的方法已被標示為 @Deprecated。

**6.2. Caught exceptions: not ignored**

下面這個異常 (Exception) 描述，他是少數在發生異常時，可以不做回應的異常處理。以標準的異常回應，是需要記錄下來的。當在 catch 區塊中，確實沒有任何動作的話，便用註解明確的說明其原委。

Example:

[](https://camo.githubusercontent.com/8cb67735487b728a3366e0ff319a274bab3f8abb853cedb8808ad14769393d5b/68747470733a2f2f692e696d6775722e636f6d2f35596e4d6758742e706e67)

**6.3. Static members: qualified using class**

引用靜態成員必需搭配著類別 (class) 名稱才是適當的用法，不是和一個物件類型或是描述句來使用。

Example:

Foo aFoo = ...;

Foo.aStaticMethod(); //good

aFoo.aStaticMethod(); //bad

**6.4. Duty of Controller, Service and Mapper**

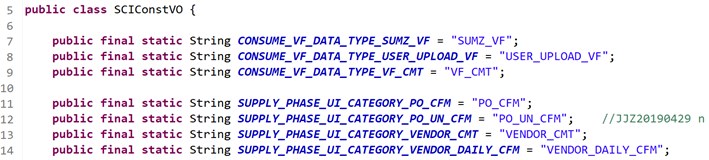
1. Controller，Service 以及 Mapper之間的權責和調用要明確：Controller 只能調用 Service，Service只能調用其他Service和共用的Mapper
2. Controller 負責 Token 驗證
3. Service負責處理業務邏輯和 Exception，不應該包含任何和 Database 相關的代碼
4. Mapper只負責對 Database 的 CRUD；函式一定簡潔，每一個函式只做一件事；把 Catch exception 留給 Service處理。

例外: 對於 Common service，如日期轉換或 Alert mail 等，不在此限。

**6.5. Single class for storage definition**

Database 如 Elasticsearch index/type name, Oracle table name, System Definition/User Definition 的 mapping type…，以 final static 的方式統一定義到一個class。

Example:

[](https://camo.githubusercontent.com/9c55feafc15c14ff69e67e368846b9790e1fe9e92d068d9b28ae1063d0effcc1/68747470733a2f2f692e696d6775722e636f6d2f576e416136344a2e706e67)

**6.6. Redis services**

每個Redis Key 的邏輯用單獨的Class來處理，包括 set, get, delete 和 set expired time等。

Example:

[RedisMpspSciProcessCtrlDao.java](https://github.com/WistronAvatar/Wistron-Avatar/blob/master/RedisMpspSciProcessCtrlDao.java)

**6.7. Async process for non-biz logic**

與業務邏輯無關的代碼，如 Write Log 和 Send Mail ，應該採取非同步觸發的方式來實現。因而不會因為這些邏輯有 exception，導致主程式執行不下去或影響效能。

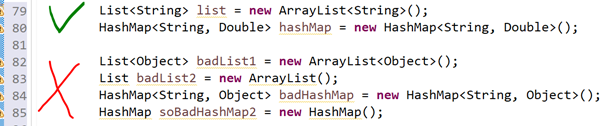
**6.8. Input parameter limit for method: 4**

一般情況下，函式的 Parameter 不要超過 5 個，如果超過應定義 Class 來封裝。

**6.9. Collection Type Must Be Declared (List, Map, Set)**

永遠都要宣告 collection type，但不要用 Object type，不要讓人去猜 collection要怎麼使用。

Example:

[](https://camo.githubusercontent.com/c2bab4152061f398157b5fdb347cab06425b75e1c8a87e40eda52ef0cf2674c8/68747470733a2f2f692e696d6775722e636f6d2f57536e70596a362e706e67)

**6.10. Replaced JsonObject with Java VO**

儘量使用 Java VO 替代 JsonObject。JsonObject 動態添加屬性的功能雖然靈活，但會增加程式閱讀和 tracking issue 的難度。

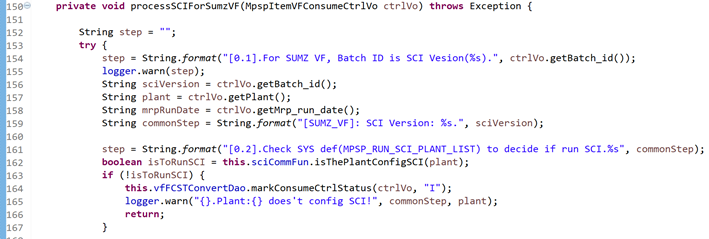
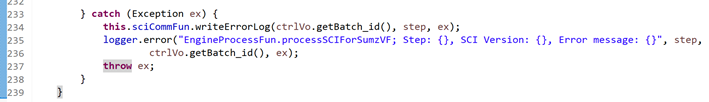
**6.11. Extend Data VO**

如果需要在共用的 Data VO 加入特定 Project 的專屬使用欄位，請繼承該 Data VO 避免影響他人。

**6.12. Main Flow Log**

如果程式有主流程函式，請對每個步驟加上對應的 Log (Info level) 並且要記錄處理時間。

Example:

[](https://camo.githubusercontent.com/352d364830c2614b7c62e35568eda3da5ac4b2fd9fd9e767f53241eddcc50999/68747470733a2f2f692e696d6775722e636f6d2f574258497355592e706e67) [](https://camo.githubusercontent.com/093a88f1349b8451c0cf9fe22de23c1fd03659acfee06a55974c42109e27503b/68747470733a2f2f692e696d6775722e636f6d2f37523430654f362e706e67)

**6.13. Main Flow Log**

如果程式有主流程函式，請對每個步驟加上對應的 Log (Info level) 並且要記錄處理時間。

**6.14. Meaningful error**

如果程式有設定 try/catch，並且程式也在 catch 中寫下 error log 或是發送 alert mail，表示這個錯誤是需要被重視的，不要將無關痛癢的錯誤寫進 log 或發信，請適當的將其處裡 (忽略或是 re-try)。

**7. Package Definition & Input / Output Param**

**7.1. Package Definition**



1. Java Common Package：
   1. 放置共用的程式，包含共用的Controller、Util、Vo、Service及Mybatis的entity & mapper均統一放置於此Package之中。
   2. 其中的datasource，請依不同數據源名稱自行定義Folder Name。
2. Java Custom Package：
   1. 各Use Case自行依開發功能自行定義Folder Name。
   2. 當中含有該功能特有使用到的Controller(API接口)、Service(業務邏輯)及Vo(數據層)。
3. Resources Package：
   1. 放置yml檔、log相關xml設定。
   2. 其Sub-Folder：templates放置各種Template模板，如發送email的Template、Excel Sample Template...等等。

**7.2. Input Param**

1. Job傳入參數

URL：https://DomainName/ServiceName/FunctionName

ServiceName & FunctionName : 請採用首字小寫的駝峰式規則

VO：JobApiRequestVo<T>

格式：

{

"jobApi": "/ftpApi/stdpriceDtl",

"jobParameter": "[{"seq": "2020090111091586310710197"},

{"requestInfo": T}]"

}

Example：

https://xxxxxx/collectSfcsLine/genSfcLine

{

"jobApi": "/collectSfcsLine/genSfcLine",

"jobParameter": [

{"seq": "202108301413158632107173",

"requestInfo": ["F137"]}

]

}

1. 一般API傳入參數

URL：https://DomainName/ServiceName/FunctionName

ServiceName & FunctionName : 請採用首字小寫的駝峰式規則

VO：ApiRequestVo<T>

格式：

{

"userId": "10710197"

"token": "E0MqR2YciHYbgh16R2pv9Yg1uC0=",

"requestInfo": T

}

Example：

https://xxxxxx/costBomExecute/getPullDownMenu

{

"userId": "10710197"

"token": "E0MqR2YciHYbgh16R2pv9Yg1uC0=",

"requestInfo": {"sysParmType":"1"}

}

**7.3. Output Param**

1. Success

VO：ApiResponseVo<T>

格式：

{

"result": "0000000",

"message": "api return ok",

"responseObject": T

}

Example：

{

"result": "0000000",

"message": "api return ok",

"responseObject": [

"JPY",

"USD",

"NTD",

"RMB"

]

}

1. Failure

VO：ApiResponseVo<T>

格式：

{

"result": "900\*\*\*\*",

"message": Exception.getMessage(),

"responseObject": T

}

result統一讀取postgre db table (at\_ba\_sys\_error\_message)中的message\_id設定

message統一讀取postgre db table (at\_ba\_sys\_error\_message)中的descr\_en設定

Example：

{

    "result": "9000000",

    "message": "This base data already exists!",

    "responseObject": null

}

**Reference**

<https://google.github.io/styleguide/javaguide.html> <https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html> <https://blog.mosil.biz/2014/05/java-style-guide/> <https://standardjs.com/readme-zhtw.html> <https://docs.oracle.com/javase/tutorial/java/data/characters.html> <https://blog.codinghorror.com/new-programming-jargon/>