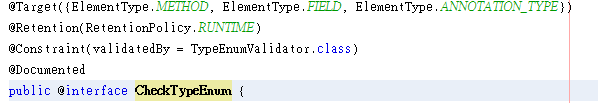
如何custom validation annotation

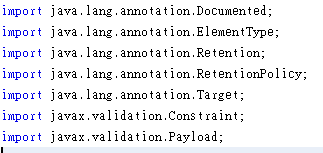
1. 先建立annotation
2. 建立對應的Validator
3. 更新annotation 上的Constraint validatedBy資訊
4. 在VO上面使用定義好的annotation
5. 自行決定定義訊息
6. 錯誤訊息 設定相關

* Annotation如何建立
  1. 建立 空殼

Annotation 使用@interface



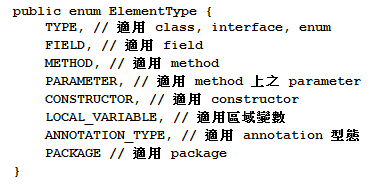
* 1. 使用到的class



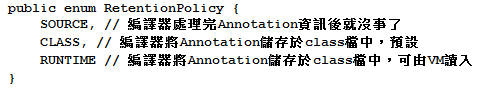
* 1. annotation屬性說明

1. target是說明這個annotation可以使用在哪邊

如果再沒有宣告的地方使用會發生錯誤



1. retention是指何時會用到



1. documented 是指可不可以被java doc

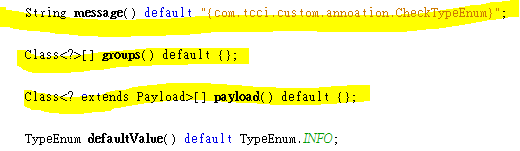
* 1. annotation 屬性設定

基本上長得像下面

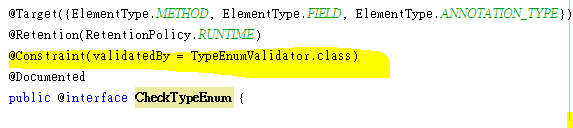
\*黃色的是如果要用來驗證的 需要設定這些屬性

\*預設值不能為空

\*沒設預設值的使用@ 就必須給值



* 1. 如果要拿來驗證使用的 在validator寫好之後加上

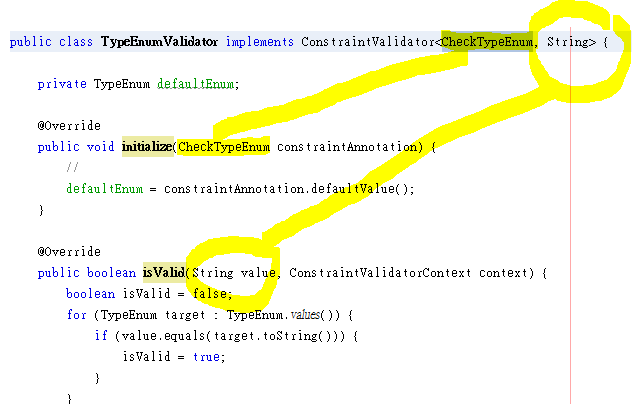


* Validator 如何建立

1. 實作javax.validation.ConstraintValidator



1. 設定相關對應的annotation及輸入的 物件型態



* 在VO上面使用建立好的annotation



* 透過validator驗證

基本上constraintViolations.size 為空代表沒有錯誤

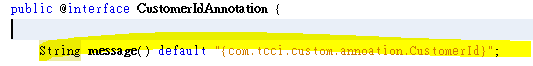
不為空 才有錯誤訊息可以取得constraintViolations.iterator().next().getMessage()

有錯誤訊息使用才不會噴錯



* 自行決定 定義訊息

驗證用的annotation雖然宣告時 有要設定default



但其實在validator時 可以自己改變 不單純只是回傳true / false



此外針對驗證 可以多個錯誤訊息 不見得 只能有一個

* 錯誤訊息設定相關

1. 如果是設定 長得像以下格式

{com.tcci.custom.annoation.CustomerId.IdLength}

會去訊息檔抓

1. 但如果直接在annotation 上直接設定寫死

應該就不會去設定檔抓了



但其實不太建議 因為這樣會沒有i18n

下面這樣就會去抓i18n



1. 錯誤訊息設定在ValidationMessages.property

針對語系請自己增加對應的檔案

1. 錯誤訊息 格式 設定
2. 固定訊息 沒有參數

com.tcci.custom.annoation.CustomerId = customer input error .

1. 固定訊息 使用參數



黃色區塊基本上就是annotation上面宣告的method(屬性)

基本上可以透過名稱指定

1. 固定訊息 但要顯示輸入值



基本上參數目前看起來 如果不額外處理 可能大都只能從annotation上取得

但有些非annotation取得的 可能就只能從

