

thread safe

Bean Norton Table View

..

Base

~ thread safe

Bean Parameters

..

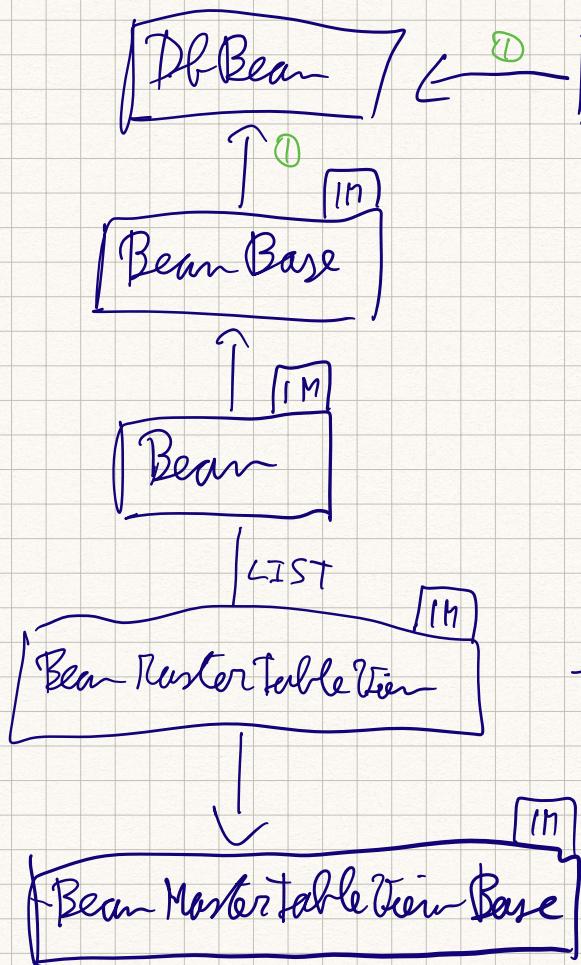
Base

Bean HTML View

..

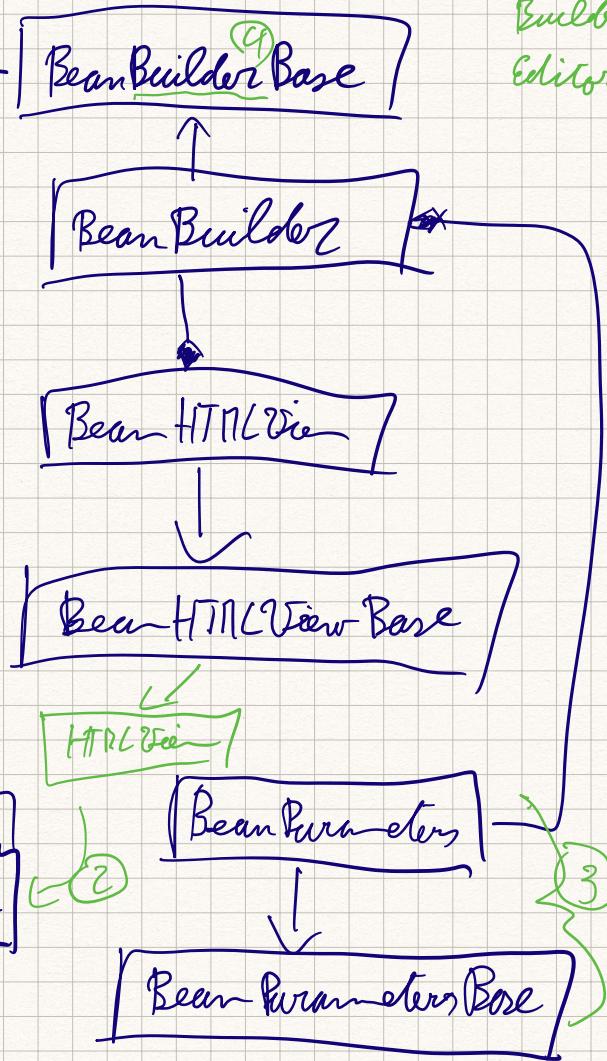
Base

Simpler variation



Questions

④ ne...?
Builder?
Editor?



① = héritage ou fonction
stratégies appelées
par Bean et BB?

③ = quelle(s) intégration(s)
BeanBuilder :)

② = sur le modèle (:)
↳ HTML Helper précisé ici (!)

Mots

→ min. version = java 11

→ changes in item-order ?

→ how to know if a Bean is no longer up-to-date ?

↳ last-update? (optional?)

↳ backward compatibility

→ interface generation ? pas de 1^{ère} itér.

↳ single web app modified

↳ new web app → bootstrap 5

↳ tailwind + alpine.js

↳ command line

→ labels ?

↳ keep resource files for now? X

↳ go all in with default integration of label table

=> creation des tables beamaker- ?

beamaker-language

beamaker-labels

beamaker-label-data

X

↳ or keep labels class system

(backward compatibility)

↓

→ système de gérer / projets

↳ label par défaut, préfixer gr les tables, etc.

} pas de 1^{ère}
} itér

→ question code cleaner / plus lisible entre
BeanBase & BeanBuilderBase ? *yes I problem*

↳ ok, or pas facile ?

↳ classe abstraite entre BeanBuilderBase & PBBean :

⇒ content les deux. X = problématique

↳ = BeanData X *me, because it's final
for Bean !!*

→ Constructeurs Bean

↳ 5 constructeurs actuels

*mutable pour
BeanBuilder !!*

→ Constructeurs BeanBuilder

↳ BB()

↳ BB(long id)

↳ BB(Bean b) { this(b.getId()) }

→ version java ?

↳ //

→ // A needed classes !!! *no 'll' suffix*

→ HTMLFormHelper

↳ = interface

Implementation ↳ LegacyHTMLFormHelper = V1

↳ Bootstrap3HTMLFormHelper

= V1 sans les factories uniques
qui utilisent pas HF parameters

-> MasterTableView ->

L) reimplanter sur le modèle de BeanHTMLView

L) add ... Cell (String key [, data])

au lieu de get ... Cell ([<data>])

=> modifications importantes du BaseMasterTableView ?
worth it ?

L) pas dans la 1^{ère} itération

-> JSONView / XMLView, etc.

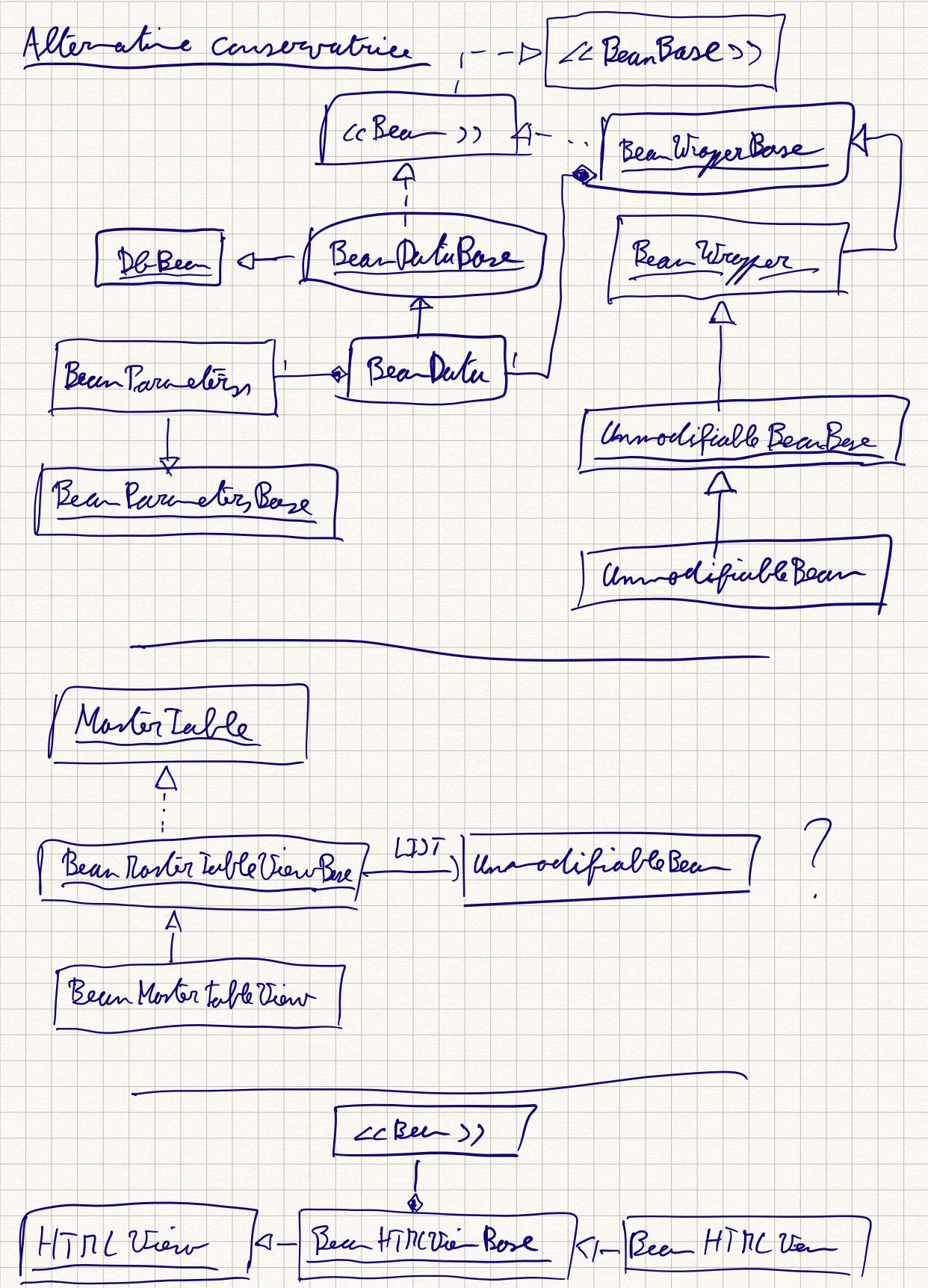
HTMLTableView

L) se préfère ignorer ou réimplanter
prudente des 1^{er} types (probl. JSONView problematic !)

Ideas diverses

↓ id

- hascode Bean : utiliser Object.hashCode (long)
(pas besoin equals() / hashCode() du BeanBuilder)
- fluent interface pr BeanBuilder/Editor



Renages d'erreur / validation des champs

field => 3 vérif

- ↳ required → no change
- ↳ OK ✘
- ↳ unique → no change

* dans BeanParameter (Box)

↳ liste de test de validation par chaque champ

↳ peut être vide

↳ test renvoie null ou le msg du libellé content
= OK le msg d'erreur = PAS OK

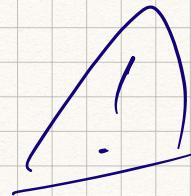
↳ `list<Function<[field type], String>>`

mais plus ^{plus} possible

Plus complexe : au lieu de String :

class FieldValidity Test Result

- boolean status
- String errorMessageLabelName
- list<Object> errorMessageParameters



↳ private constructor

↳ static OK {
 status = true
 errorMessageLabelName = null
 errorMessageParameters = null}

↳ static create (String errorMessageLabelName)

↳ static create (String errorMessageLabelName, list<Object> errorMessageParameters)
↳ static create (String errorMessageLabelName, list<Object> errorMessageParameters, Object... objects)

Paramètres & cache des labels

→ Bea Parameter = singletor (\neq over \Leftarrow héritage)
 Δ overload \Rightarrow BeaParameterBase

→ Map < String, DbBeaLabel > labelCache
 \Downarrow

Q : quel niveau de restriction, (if or g)

→ herient System = no exception

2 err. \rightarrow no label : HERE NO LABEL label_name
 \rightarrow no leg : HERE NO LANG key_id_label_name

Δ

↳ alternative = cache global à l'application ?!

↳ ds les 2 cas : il faut clearLabelCache()

Global Cache

