

接口层

Spring bean 1(单例)

业务侧自实现的bean, 需要继承Abstract Load
Get() --- 根据key获取想要的缓存data
Load() --- 自定义将数据对照某种key放入缓存
Init() --- 自定义的key结构
.....

Spring bean 2(单例)

业务侧自实现的bean, 需要继承Abstract Load
Get() Load() Init()

⋮

Spring bean n(单例)

业务侧自实现的bean, 需要继承Abstract Load
Get() Load() Init()

业务层

业务侧可自由调用接口层的get/load等方法,
线程安全在基础层已经有了对应控制

基础层

Abstract Load

对缓存体本身的一些刷新/增/删/改/查操作
Reload() --- 需要保证线程安全的reload
Get ()
Clean()
Is Empty()
keyset()
Get Key Class()
Get Data Class()
Modify Value()
.....

Abstract Initialization

对缓存体本身的一些初始化和基本属性的操作
Init() Eff Switch()

Local Cache Map

一级key是String 一级key的value 是子map,
key (二级key) 是Object,value是实际缓存值
结构: `Map<String, Map<Object, List<T>>>`
Add Key()
Key Set()
Key Value Set()
Get()
Put()
Is Empty()
.....

界面

Local Cache View Display

Bean Map Name1

key Level1 name
key Level2 name

Bean Map Name2

key Level1 name
key Level2 name

Bean Map Name N

key Level1 name
key Level2 name

Show cache Value<T> Object to
Json String here

Reload

Clean

Save

这里有一个基于左侧结构的专门用于展示/手动刷新/手动修改缓存的界面, 有如下功能;
1根据的Spring bean name 1—n 反射找bean, 并以三级树和input框的形式展示缓存数据
树1级节点为bean name 2级节点为1级key 3级节点为2级key
input框为2级key对应的value <T>的Jason String形式
2 对缓存的重新加载;
3 对缓存的清楚;
4 对缓存的部分数据的编辑和保存, (将新的Jason String反转成新的value <T>, 再用这个新值替换原缓存值)
(2/3/4功能主要都是通过反射找到Abstract Load. 然后使用其中的方法接口)