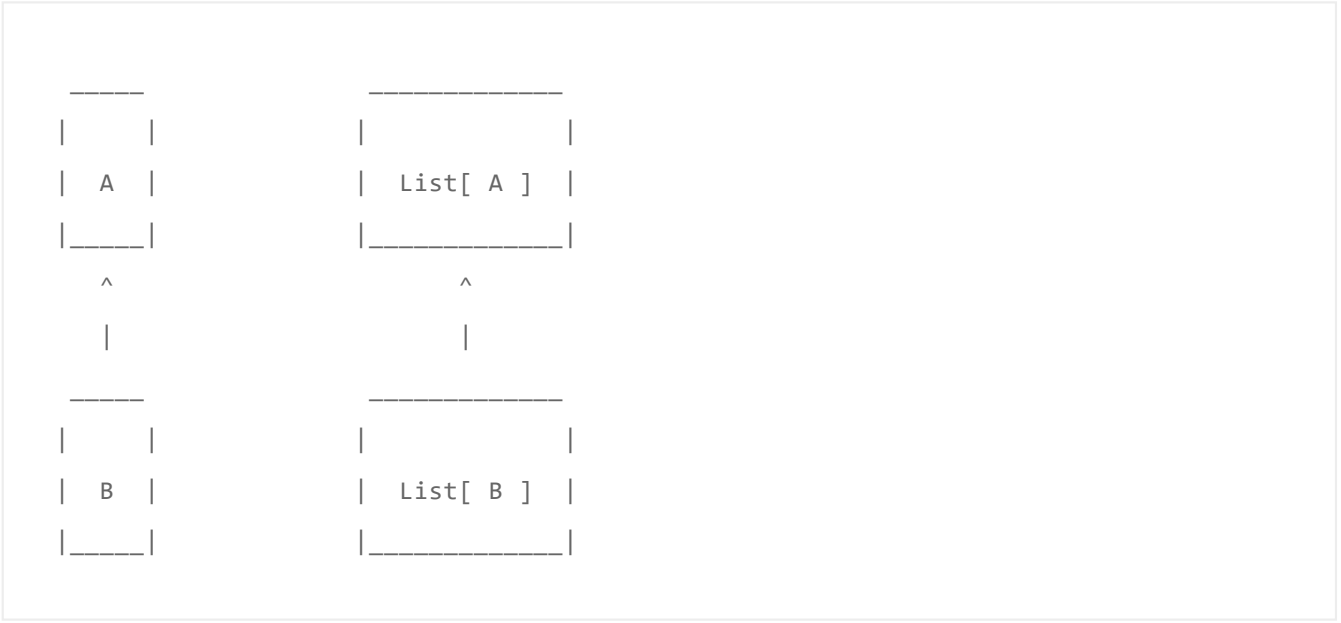
写点什么

有关jvm,scala与后端架构

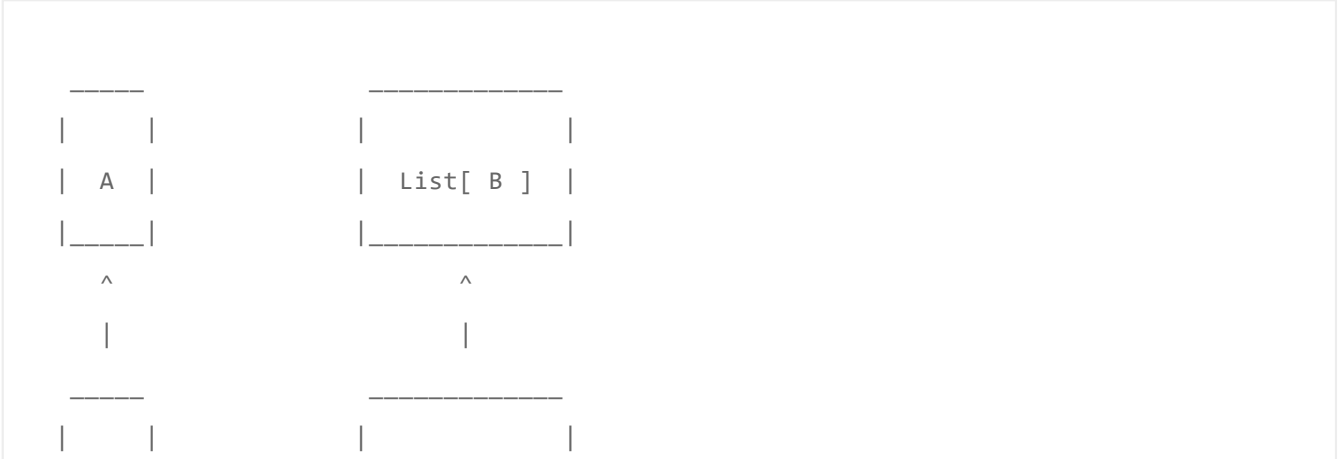
scala类型系统：15) 协变与逆变

对于一个带类型参数的类型，比如 `List[T]`，如果对A及其子类型B，满足 `List[B]`也符合 `List[A]`的子类型，那么就称为**covariance(**协变**)**，如果 `List[A]`是 `List[B]`的子类型，即与原来的父子关系正相反，则称为**contravariance(**逆变**)**

协变：



逆变：



```
|  B  |            |  List[ A ]  |
|_____|            |_____________|
```

如果一个类型支持协变或逆变，则称这个类型为**variance(**翻译为可变的或变型**)**，否则称为**invariant(**不可变的**)**

在Java里，泛型类型都是invariant，比如 `List<String>` 并不是 `List<Object>` 的子类型。Java并不支持声明点变型(declaration-site variance，即在定义一个类型时声明它为可变型，也称definition-site)，而scala支持，可以在定义类型时声明(用加号表示为协变，减号表示逆变)，如:

```
trait List[+T] // 在类型定义时(declaration-site)声明为协变
```

这样会把`List[String]`作为`List[Any]`的子类型。

不过Java支持使用点变型(use-site variance)，所谓“使用点“，也就是在声明变量时:

```
List<? extends Object> list = new ArrayList<String>();
```

scala为了兼容java泛型通配符的形式，引入存在类型(existential type，后边再讲)时，也支持了使用点变型(use-site variance)

```
scala> val a : List[_ <: Any] = List[String]("A")
a: List[_] = List(A)
```

要注意variance并不会被继承，父类声明为variance，子类如果想要保持，仍需要声明:

```
scala> trait A[+T]

scala> class C[T] extends A[T]  // C是invariant的

scala> class X; class Y extends X;

scala> val t:C[X] = new C[Y]
```

```
<console>:11: error: type mismatch;
 found   : C[Y]
 required: C[X]
Note: Y <: X, but class C is invariant in type T.
You may wish to define T as +T instead. (SLS 4.5)
```

必须也对C声明为协变的才行：

```
scala> class C[+T] extends A[T]

scala> val t:C[X] = new C[Y]
t: C[X] = C@6a079142
```

对于协变与逆变的使用，参考《Effective Java》的PECS原则。这里有篇旧文也可供参考。

本条目发布于2013-09-10 [http://hongjiang.info/scala-covariance-and-contravariance/] 。属于scala分类，被贴了 scala、type-system 标签。

《scala类型系统：15) 协变与逆变》上有8条评论

chen
2013-11-3020:39

遇见一个这样的问题
class In[+A]{def fun(x:A){}}
会提示error: covariant type A occurs in contravariant position in type A of value x
class In[+A]{def fun(x:A){}}
^
而这样不会出现问题
class In[-A]{def fun(x:A){}}
想请教一下这是什么原因呢？

hongjiang 文章作者

2013-12-0101:40

你的问题请看这里：http://hongjiang.info/scala-pitfalls-10/

Pingback引用通告： scala雾中风景(10): 逆变点与协变点 | 在路上

mars

2014-01-0712:01

楼主你好，最近看了你所有的文章，感触很深，国内研究scala的人实力真强，希望你继续努力推广scala
我一直在研究用scala+akka构建一个通用mmo游戏框架

```
/*
* ©2013 Mars Hsu (marsxu1984@gmail.com)
*
* All rights reserved.
*
*/

package io.marsdigtal.coremodel_module.cache

import io.marsdigtal.coremodel_module.db._

import com.google.protobuf.ByteString
import com.googlecode.concurrentlinkedhashmap.ConcurrentLinkedHashMap
import com.mongodb.casbah.Imports._
import io.marsdigtal.coremodel_module.model.dto.Common.ResourceType

/**
* resource cache.
*
*/
trait ResourceCache {

val store = new ConcurrentLinkedHashMap.Builder[Int, CacheData]()
.maximumWeightedCapacity(5000)
.concurrencyLevel(8)
.build()
```

```
val resourceConfig = List[ResourceConfig]()

val resourceColl = BaseMongoCollections.getInstance.ResourceColl

def as[A](indexId: Int): Option[A] = {
findResource(indexId).bean.asInstanceOf[Option[A]]
}

def dict(indexId: Int): Option[DictDTOData] = {
findResource(indexId).dto
}

def dicts(indexIds: Set[Int]): Option[List[DictDTOData]] = {
Option(indexIds.map { each =>
val cache = findResource(each).dto
cache match {
case Some(ca) => {
ca
}
}
}.toList)
}

def resourceType(indexId :Int): ResourceType = {
findResource(indexId).resourceType
}

def findResource(indexId :Int): CacheData = {
if (store.containsKey(indexId)) {
store.get(indexId)
} else {
resourceColl.findOne(MongoDBObject("_id" -> indexId)) match {
case Some(res) => {
if (res.containsField("r_ty")) {
val resourceType = res.as[Int]("r_ty")
resourceConfig.find(_.resourceType == resourceType) match {
case Some(cfg) => {
val obj = cfg.actionClazz.newInstance.asInstanceOf[MergeData[AnyRef]]
val template = cfg.mongoConvert.asInstanceOf[
MongoDBObjectConvert[AnyRef]].convertFromDBObject(res)
obj.insertBond(template)
obj.insertBlood
putResourceToCache(indexId, obj, cfg, ResourceType.valueOf(resourceType))
store.get(indexId)
}
case None => throw new ResourceTypeNotFindException
}
```

```
} else {
throw new ResourceTypeNotFindException
}
}
case None => throw new ResourceIdNotFindException
}
}
}

def putResourceToCache(id: Int,
obj: MergeData[AnyRef],
config: ResourceConfig,
resourceType: ResourceType) = {
val dto = config.dtoConvert.asInstanceOf[ProtobufConvert[AnyRef]].convertToDTO(obj.data)
val data = DictDTOData(id, config.dtoClazzName, dto.toByteString)
store.put(id, CacheData(Option(obj), Option(data), resourceType))
}
}

case class ResourceConfig(resourceType: Int,
mongoConvert: Any,
dtoClazzName: String,
dtoConvert: Any,
templateClazz: Class[_ <: AnyRef],
actionClazz: Class[_ <: AnyRef])

case class CacheData(bean: Option[AnyRef], dto: Option[DictDTOData], resourceType:
ResourceType)

case class DTOData(className: String, data: ByteString)

case class DictDTOData(indexId: Int, className: String, data: ByteString)

class ResourceTypeNotFindException extends RuntimeException

class ResourceIdNotFindException extends RuntimeException
```

mars

2014-01-0712:02

如果有机会发gmail给我，加强交流

mars
2014-01-0712:05

```
/*
* ©2013 Mars Hsu (marsxu1984@gmail.com)
*
* All rights reserved.
*
*/

package io.marsdigtal.XXX.cache

import io.marsdigtal.coremodel_module.cache.{ResourceConfig, ResourceCache}
import io.marsdigtal.coremodel_module.model.dto.Common.ResourceType
import io.marsdigtal.XXX.dto.{SQDTONamespace, QuestDTOConvert, EquipmentDTOConvert,
CardDTOConvert,
CityDTOConvert, QuestionDTOConvert}
import io.marsdigtal.XXX.model.{Equipment, Quest, Question, Card, City}
import io.marsdigtal.XXX.model.template.base.{CardTMongoBasicConvert, CardT,
QuestTConvert, CityTConvert,
QuestionTConvert, EquipmentTMongoBasicConvert, EquipmentT, CityT, QuestT, QuestionT}

object SQResourceCache extends ResourceCache {

override val resourceConfig = List[ResourceConfig](
ResourceConfig(
ResourceType.CHARACTER_VALUE,
CardTMongoBasicConvert,
SQDTONamespace.CardDTO,
CardDTOConvert,
classOf[CardT],
classOf[Card]),
ResourceConfig(
ResourceType.EQUIPMENT_VALUE,
EquipmentTMongoBasicConvert,
SQDTONamespace.EquipmentDTO,
EquipmentDTOConvert,
classOf[EquipmentT],
classOf[Equipment]),
```

```
ResourceConfig(
ResourceType.QUESTION_VALUE,
QuestionTConvert,
SQDTONamespace.QuestionDTO,
QuestionDTOConvert,
classOf[QuestionT],
classOf[Question]),
ResourceConfig(
ResourceType.CITY_VALUE,
CityTConvert,
SQDTONamespace.CityDTO,
CityDTOConvert,
classOf[CityT],
classOf[City]),
ResourceConfig(
ResourceType.QUEST_VALUE,
QuestTConvert,
SQDTONamespace.QuestDTO,
QuestDTOConvert,
classOf[QuestT],
classOf[Quest])
)
}
```

amos.zhou
2014-06-1800:18

正在看《快学scala》 本来在想 :与逆变 的关系与区别。 最开始感觉在限制方向上一致的，那么2者的区别是什么，琢磨的半天。总感觉有那么一点地方不同，但是又好像不能用语言组织出来。

看了你的博客，豁然开朗。 把我想的区别，表达出来了。 一个是使用变型，一个是声明时变型~

真大神！

Pingback引用通告： Scala Convariance and Contravariance(逆变与协变) | 小白菜