

# msgpack源码修改记录

被zhangjunfeng1添加，被zhangjunfeng1最后更新于八月 20, 2015

## 1、TemplateRegistry

修改方法：

### ①lookup方法：

去掉 `synchronized` 修饰符；

只有是 `Throwable`、`List`、`Set`、`Map` 的子类时才查询父类模板。防止父类模板序列化子类时Filed不一致

### ②buildAndRegister

增加模板存在检查，存在时直接返回，防止重复注册引起的栈溢出问题

### ③lookupSuperclasses

在目标类是接口并且对应的模板是 `AnyTemplate` 时重新根据实际类注册

### ④lookupInterfaceTypes

在目标类是接口并且对应的模板是 `AnyTemplate` 时重新根据实际类注册

### ⑤getTemplateFromCache（新增）

解析泛型类互相嵌套时模板注册

## 2、MessagePack类

添加

```
private static final int POOL_SIZE = 500;
private LinkedBlockingQueue<BufferPacker> packerPool = new LinkedBlockingQueue<BufferPacker>(POOL_SIZE);
private LinkedBlockingQueue<BufferUnpacker> unpackerPool = new LinkedBlockingQueue<BufferUnpacker>(POOL_SIZE);
```

修改方法名：

```
createBufferPacker() 方法
byte[] write(T v)
write(T v, Template<T> template)
write(Value v)
read(ByteBuffer buffer)
read(byte[] bytes, T v, Template<T> tmpl)
read(ByteBuffer b, T v, Template<T> tmpl)
createBufferUnpacker()方法
read(byte[] bytes, int off, int len)方法
read(ByteBuffer buffer)
read(byte[] bytes, T v, Template<T> tmpl)
read(byte[] bytes, int off, int len, Class<T> c)
read(ByteBuffer b, T v, Template<T> tmpl)
去掉public MessagePack(MessagePack msgpack)构造函数
```

修改内容：

```
createBufferPacker方法，先从pool中取得，pool中没有时，new创建
createBufferUnpacker方法，先从pool中取得，pool中没有时，new创建
write和read方法调用BufferPacker和BufferUnpacker后，增加try{}finally{ pool.offer()}代码
```

## 3、AnyTemplate

修改write和read方法，在序列化数据中写入实际全路径类名。

## 4、MessagePackPacker

### ① addRef

序列化时记录重复引用的对象，如果重复引用时不再重新序列化，而是把引用的位置返回。序列化时把结果写进序列化数据中。

## 5、MessagePackUnpacker

新增方法：

① isRef

判断是否是引用数据

② setRef

把数据放到缓存中，记录其位置

③ readRef

根据位置取出实际的序列化数据

④ cachingClear

清除缓存的数据状态

## 6、DefaultBuildContext

此类是根据用户自定义的Bean来动态生成序列化模板的。

修改后生成的模板变动如下：

- ① 按照顺序存了Bean中的所有Field，并且开启了可访问权限
- ② 取得所有构造函数，并找出最高优先级的构造函数并保存在模板类中，供反序列化时实例化使用。
- ③ 根据保存的构造函数取得参数
- ④ 检查Field声明类型和实际类型的父子类检查，在Map中保存检查结果
- ⑤ 反序列化时检查数据的Field个数当前Bean的Field的个数，个数不一致时会按照少的进行处理。

增加检测重复引用和互相依赖机制，相同的对象只保存引用位置

## 7、TemplateReference

修改方法：

① validateActualTemplate

在实际模板类为空或者为TemplateReference时查找实际类对应模板


























② getActualTemplate

找到类实际的序列化模板

## 8、2014.10.10 ArrayTemplateBuilder

修改toTemplate方法，在找不到具体类型时增加动态注册模板功能

## 9、新增模板：

 CharacterArrayTemplate.java  
 CharArrayTemplate.java  
 InvocationTemplate.java  
 JSFAbstractTemplate.java  
 JSFCalendarTemplate.java  
 JSFCharsetTemplate.java  
 JSFClassTemplate.java  
 JSFExceptionTemplate.java  
 JSFFileTemplate.java  
 JSFListTemplate.java  
 JSFLocaleTemplate.java  
 JSFMapTemplate.java  
 JSFObjectArrayTemplate.java  
 JSFObjectTemplate.java  
 JSFSetTemplate.java  
 JSFSqlDateTemplate.java  
 JSFStackTraceElementArrayTemplate.java  
 JSFStackTraceElementTemplate.java  
 JSFTimestampTemplate.java  
 JSFTimeTemplate.java  
 JSFTimeZoneTemplate.java  
 JSFURITemplate.java  
 JSFURLTemplate.java  
 JSFUUIDTemplate.java  
 ResponseTemplate.java

---

无

---