链滴

系统重构时优化关于枚举和字符串

作者：felayman
原文链接：https://ld246.com/article/1456937914972
来源网站：链滴

最近在重构同事写的代码,代码设计的非常好,由于涉及到的比较多,我这里就以一个枚举类来记录关于枚举和字符串的优化(不涉及到业务部分)
代码如下：
<pre class="prettyprint">package com.vcg.community.data
constant;
public enum InvokerTypeEnum {
    //读操作
    ReadEntityById {
        public String cmdClassName() {return "com.vcg.community.data.command.read.EntityBydCmd";}
    },
    ReadEntityByIndex {
        public String cmdClassName() {return "com.vcg.community.data.command.read.EntityByndexCmd";}
    },
    ReadEntityByLink{
        public String cmdClassName() {return "com.vcg.community.data.command.read.EntityByLnkCmd";}
    },
    ReadEntityByColumn{
        public String cmdClassName() {return "com.vcg.community.data.command.read.EntityByolumnCmd";}
    },
    ReadCount {
        public String cmdClassName() {return "com.vcg.community.data.command.read.CountCd";}
    },
    ReadCountByLink {
        public String cmdClassName() {return "com.vcg.community.data.command.read.CntByLikCmd";}
    },
    ReadStateByLink {
        public String cmdClassName() {return "com.vcg.community.data.command.read.StateByLnkCmd";}
    },
    Search {
        public String cmdClassName() {return "com.vcg.community.data.command.read.Searchmd";}
    },

Msg {
    public String cmdClassName() {return "com.vcg.community.data.command.read.MsgCmd";}
},
ReadEntityBySql {
    public String cmdClassName() {return "com.vcg.community.data.command.read.EntityBySqCmd";}
},
SearchBySql {
    public String cmdClassName() {return "com.vcg.community.data.command.read.SearchBySqCmd";}
},
QueryCountBySql {

```
    public String cmdClassName() {return "com.vcg.community.data.command.read.QueryCoun
BySqlCmd";}
},
QueryIdsBySql {
    public String cmdClassName() {return "com.vcg.community.data.command.read.QueryIdsB
SqlCmd";}
},
//写操作
InsertEntitys{
    public String cmdClassName() {return "com.vcg.community.data.command.write.InsertEntit
sCmd";}
},
DeleteEntitys{
    public String cmdClassName() {return "com.vcg.community.data.command.write.DeleteEnti
ysCmd";}
},
DoLinks{
    public String cmdClassName() {return "com.vcg.community.data.command.write.DoLinksC
d";}
},
UndoLinks{
    public String cmdClassName() {return "com.vcg.community.data.command.write.UndoLink
Cmd";}
},
WriteOther{
    public String cmdClassName() {return "com.vcg.community.data.command.write.WriteOthe
Cmd";}
},
ChangeCount{
    public String cmdClassName() {return "com.vcg.community.data.command.write.ChangeCo
ntCmd";}
};
public abstract String cmdClassName() ;

}
</span></pre>

<p><span style="font-size: x-small;">上下文代码</span></p>
<pre class="prettyprint"><span style="font-size: x-small;">InvokerTypeEnum invokeType = i
putObject.getInvokeType();
        if (invokeType == null) {
            return new OutputObject(OutputObject.CODE_ERROR, "the attr invoke_type can not b
 null");
        }
       <span style="color: #ff0000;"> String className = invokeType.cmdClassName()</span>;
       <span style="color: #ff0000;"> AbstractCommand dp = (AbstractCommand) SpringUtils.
etBeanByClassName(className);</span>
        outputObject = dp.checkInputAttr(inputObject);
        if (outputObject.getCode() == OutputObject.CODE_ERROR) {
            return outputObject;
        }
       <span style="color: #ff0000;"> outputObject = dp.execute(inputObject);</span>
        return outputObject;
    }</span></pre>
```

<pre class="prettyprint">public abstract class AbstractCo
mand {
    @Autowired
    protected RedisStrategy redisStrategy;
    @Autowired
    protected MysqlStrategy mysqlStrategy;
    @Autowired
    protected SearchStrategy searchStrategy;
    public abstract OutputObject execute(InputObject inputObject);
    public abstract OutputObject checkInputAttr(InputObject inputObject);
}</pre>
优化后的代码：
<pre class="prettyprint">public enum InvokerType {
    //读操作
    ReadEntityById,
    ReadEntityByIndex,
    ReadEntityByLink,
    ReadEntityByColumn,
    ReadCount,
    ReadCountByLink,
    ReadStateByLink,
    Search,
    Msg,
    ReadEntityBySql,
    SearchBySql,
    QueryCountBySql,
    QueryIdsBySql,
    //写操作
    InsertEntitys,
    DeleteEntity,
    DoLinks,
    UndoLinks,
    WriteOther,
    ChangeCount;
    String cmdClassName(){return this.getClass().getPackage().getName()+".command."+this.n
me()+"Cmd";}
}</pre>
这样一来,是不是更合理了呢？我们来分析一下

1.首先从代码的数量来说,是不是更精简了一些。</spa
>
2.形式来说,避开了抽象枚举,是不是更接近普通的枚举
计。
3.性能来说,同事的方式会让JVM产生更多
类似“com.vcg.community.data.command.read/write.xxxxx”的字符串常量(不是ne
出来的字符串都会在编译期间存入常量池中)
4.
统扩展性,当系统可能会新增该枚举类型时,优化后的方式更利于系统扩展,仅需新增一个枚举值即可。</
pan>
5.系统稳定性，减少可能由于字符输入(大小写,非
制手敲等原因)错误而造成spring无法加载模板实例的可能性

<span sty
e="font-size: x-small;">最终,同事在我的说服下同意了我的修改。