1. 业务接口：站在”使用者”角度设计接口

三个方面：

方法定义粒度：方法定义明确。

参数：越简练，越直接越好。

返回类型(return类型/异常)

综合上面的三个方面，所以新建了如下package,dto(业务传输对象),exception,service

|  |
| --- |
| 这里重点说下dto（之前我的理解可能一直有错误。因为我都觉得dto就是entity,其实不是这样的） |
| 在调用业务业务方法的时候，DTO对象数据是由Action生成(可能是用户输入的Form的数据，也有可能是其他情况)  在我目前理解来看，dto是做表示层（展示给用户，或者是用户需要的），而实体是数据对象(表)。表示层dto可能包含多个实体，也可能是一个实体的一部分，或者多个实体的组成部分。总而言之，dto是面向对象的，实体是面向关系数据库的。  dto其实就是简单的JAVABEAN,实现Serializable借口，可以在网络间传输  dto是为了解决entity bean可能很庞大，影响网络传输性能而产生的一种想法 比如entity bean印射到一个表字段很多，但实际使用中可能只要取得几个值，比如ID等什么的，那么就定义一个class(类） getID() { }之类的，这个class在服务器端，调用entity的local接口，或直接用jdbc操作表，所以不会对网络传输产生不利，然后网络传输这个class给客户端(可序列化），是不是减小了网络传输。  你说的没错，，其实你说的dto，就是Value Object。  作用，主要是减少setXXX()/getXX()的调用，一次性全部用一个valueobject传递回来。就是这个作用。 |

下面来看下具体的部分设计代码

Service:

|  |
| --- |
| /\*\*  \* 业务接口:站在"使用者"角度设计接口  \* 三个方面:方法定义粒度,参数,返回类型（return类型/异常）  \* **@author** sucan  \*  \*/  **public** **interface** SeckillService {    /\*\*  \* 查询所有秒杀记录  \* **@return**  \*/  List<Seckill> getSeckillList();    /\*\*  \* 查询单个秒杀记录  \* **@param** seckillId  \* **@return**  \*/  Seckill getById(**long** seckillId);    /\*\*  \* 秒杀开启时输出秒杀接口地址，  \* 否则输出系统时间和秒杀时间  \* **@param** seckillId  \*/  Exposer exportSeckillUrl(**long** seckillId);    /\*\*  \* 执行秒杀操作  \* **@param** seckillId  \* **@param** userPhone  \* **@param** md5  \*/  SeckillExecution executeSeckill(**long** seckillId,**long** userPhone,String md5)  **throws** SeckillExceprion,SeckillCloseException,RepeatKillException;  } |

dto

|  |
| --- |
| /\*\*  \* 暴露秒杀地址DTO（数据传输对象）  \* **@author** sucan  \*  \*/  **public** **class** Exposer {  //秒杀是否开启  **private** **boolean** exposed;  //一种加密措施  **private** String md5;  //id  **private** **long** seckillId;  //系统当前时间(ms)  **private** **long** now;  //秒杀开启时间(ms)  **private** **long** start;  //秒杀结束时间(ms)  **private** **long** end;    **public** Exposer(**boolean** exposed, String md5, **long** seckillId) {  **super**();  **this**.exposed = exposed;  **this**.md5 = md5;  **this**.seckillId = seckillId;  }  **public** Exposer(**boolean** exposed, **long** now, **long** start, **long** end) {  **super**();  **this**.exposed = exposed;  **this**.now = now;  **this**.start = start;  **this**.end = end;  }  **public** Exposer(**boolean** exposed, **long** seckillId) {  **super**();  **this**.exposed = exposed;  **this**.seckillId = seckillId;  }  **public** **boolean** isExposed() {  **return** exposed;  }  **public** **void** setExposed(**boolean** exposed) {  **this**.exposed = exposed;  } |

exception

|  |
| --- |
| /\*\*  \* 重复秒杀异常(运行期异常)  \* **@author** sucan  \*  \*/  **public** **class** RepeatKillException **extends** SeckillExceprion {    **public** RepeatKillException(String message){  **super**(message);  }    **public** RepeatKillException(String message,Throwable cause){  **super**(message,cause);  }  } |