Tern的基本使用

1. 对象的持久化

使用Tern实现对象持久，需完成以下步骤：

1. 要持久的对象（BO）需要继承“持久业务对象基类”，并传入其业务依赖接口（Di）

如：**public** **class** Order **extends** AbstractPersistent<StoreDi>

1. BO要持久的属性增加@Resource注解

如：@Resource

**protected** Date m\_CreateTime;

1. 当对象（要持久的属性）发生变化时需要主动调用markPersistenceUpdate();，一般在构造及set属性方法都会调用
2. 一般在Service实现里创建业务对象的持久器（可理解为业务对象的类工厂）

如：

**final** Persister<Order> m\_PsOrder;

m\_PsOrder = m\_DataHub.createPersister(Order.**class**, m\_StoreDi);

1. Spring配置文件添加保存对象的存储卷的路径及各存储目录，存储目录使用对象名的全小写（例子：order），多个之间使用分号分隔

如：

<!-- 数据存储的db -->

<constructor-arg index=*"0"* value=*"storeonline.db"* />

<!-- 对象数据存储节点 -->

<constructor-arg index=*"1"* value=*"order;order\_doc;goods"* />

持久器常用方法：

1. m\_PsOrder.get(id)：根据id取对象。返回一个对象。
2. m\_PsOrder.startsWith(**key**)：key为null时取全部对象，否则取id以key开头的所有对象。返回分页结果集。
3. m\_PsOrder.remove(id)：删除对象。
4. 持久对象的查询

查询一共有以下两种方式

1. 持久器startWhit

常用于固定从属关系的查询，通过先作id加入参数作为开头合成，查询时直接使用持久器m\_PsOrder.startsWith(**key**)即可返回key开头的所有对象。

1. 搜索器search，需要完成以下步骤

布置步骤：

1. Service实现里创建对象的搜索器

如：

**final** Searcher m\_OrderSearcher;

m\_OrderSearcher=m\_DataHub.createSearcher("order");

1. Di增加获取搜索器的方法

如：

Searcher getOrderSearcher();

1. 业务对象定义reindex()重索引方法，按业务需要组织好能搜索到此业务对象的一组关键词，调用搜索器的updateElement方法，建立关键字索引

如：

List<IndexKeyword>ks=Arrays.*asList*(IndexKeywords.*newKeyword*(getUserName(),0));

getBusinessDi().getOrderSearcher().updateElement(IndexElement.*valueOf*(getId().getOrdinal()), ks);

1. 在属性变化的方法调用reindex方法

如：构造或set或其他属性有变化的业务操作方法里，最后调用reindex

1. Spring配置文件添加保存对象索引的存储节点，使用对象名的全小写+“\_doc”（例子：order\_doc）

如：

<constructor-arg index=*"1"* value=*"order;****order\_doc****;goods"* />

使用方法：

1. 先整理要查询的关键字列表
2. 把关键字列表传进搜索器的search方法，返回结果集索引
3. 把结果集索引转换成分页对象，即刻获取关键字查询到的结果。

如：

List<IndexKeyword>ks=Arrays.*asList*(IndexKeywords.*newKeyword*(getUserName(),0));

IndexResults ir=m\_OrderSearcher.search(ks,Searcher.*OPTION\_NONE*);

**return** ResultPageWrap.*wrap*(ir, m\_PsOrder);

分页排序：

1. 定义比较器常量

如：

/\*\*

\* 最新的排在最前

\*/

**public** **static** Comparator<Order> *ORDER\_TIME\_NEW* = **new** Comparator<Order>() {

@Override

**public** **int** compare(Order o1, Order o2) {

**if** (**null** == o1 || **null** == o2) {

**return** -1;

}

**return** o1.getCreateTime().after(o2.getCreateTime())?0:1;

}

};

1. 有需要的话可使用ResultPages.*toSortResultPage*方法排序

如下：

rp = ResultPages.*toSortResultPage*(rp, Order.*ORDER\_TIME\_NEW*, ResultPage.*LIMIT\_NONE*);