**Dozer (http://dozer.sourceforge.net)**

Служит для маппинга классов (т.е. создать класс выбрав поля из другого класса).

Attributes:

The wildcard attribute is set to true by default. This means that it will automatically try to map every property in the two objects. When the attribute is set to false it will only map explicitly defined fields.

**Cumulative vs. Non-Cumulative List Mapping (bi-directional)**

If you are mapping to a Class which has already been initialized, dozer will either 'add' or 'update' objects to your List. If your List or Set already has objects in it dozer checks the mapped List, Set, or Array and calls the contains() method to determine if it needs to 'add' or 'update'. This is determined using the relationship-type attribute on the field tag. The default is 'cumulative'. relationship-type can be specifed at the field mapping, class mapping, or global configuration leve

<mappings>

<configuration>

<relationship-type>non-cumulative</relationship-type>

</configuration>

</mappings>

**Note:** if you do not define custom *equals()* and *hashCode()* methods non-cumulative option will not function properly, as Dozer will fail to determine object equality and will rely on JDK generated object Ids. In default case two instances of a class are always treated as different and update will not occure.

(При non не будут добавлены элементы которые уже есть. 'cumulative – добавить все)

### Removing Orphans

Orphans are elements which exist in a destination collection that did not exist within the source collection. Dozer will remove orphans by calling the 'remove' method on actual orphans of the underlying destination collection; it will not clear all. To determine elements which are orphans dozer uses the contains() method to check if the results contains orphans. The default setting value is false.

(удалит элементы с dest листа которые отсутствуют в src)

<!-- orphan objects will always be removed from an existing

destination List -->

<field remove-orphans="true">

<a>srcList</a>

<b>destList</b>

</field>

run:

public static void main(String[] args)

{

List<String> mappingFiles = new ArrayList<String>(1);

mappingFiles.add("dozerBeanMapping.xml");

Mapper mapper = new DozerBeanMapper(mappingFiles);

SourceChildClass sourceObject = new SourceChildClass("1","2","3");

DestinationChildClass destObject = mapper.map(sourceObject, DestinationChildClass.class, "caseA");

}

class DestinationChildClass

private Integer prop1;

private Integer prop2;

private Integer prop3;

getters/setters

SourceChildClass

private String prop1;

private String prop2;

private String prop3;

getters/setters

dozerBeanMapping.xml:

<?xml version="1.0" encoding="UTF-8"?>

<mappings xmlns="http://dozer.sourceforge.net" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://dozer.sourceforge.net http://dozer.sourceforge.net/schema/beanmapping.xsd">

<mapping map-id="caseA">

<class-a>com.platinum.SourceChildClass</class-a>

<class-b>com.platinum.DestinationChildClass</class-b>

</mapping>

<mapping wildcard="true">

<class-a>com.platinum.SourceClass</class-a>

<class-b>com.platinum.DestinationClass</class-b>

<field>

<a>sourceCol</a>

<b>destCol</b>

<a-hint>com.platinum.SourceChildClass</a-hint>

<b-hint>com.platinum.DestinationChildClass</b-hint>

</field>

<field> <!--date-->

<a date-format="MM/dd/yyyy HH:mm:ss:SS">sourceDate</a>

<b>destDate</b>

</field>

</mapping>

</mappings>