HOWTO: “JOIN”

foobar.

There are two entities, say, Room and Building, where Room belongs to a Building and Building has a name. In order to display a list of Rooms and the name of the Building they belong to, one would simply do a JOIN in ordinary SQL:

```
SELECT Room.name, Building.number FROM Room JOIN Building ON Room.building = Building.id
```

However, using the weave/bakery ORM it is not possibly to issue a custom query to the database engine. You can, however, do the following to obtain Rooms and the names of the Buildings they belong to in two steps:

1. Select the rooms, and
2. select the Buildings which id is in one of the Rooms building-references.

This can be written like this:

```
List<Room> $rooms = manager().getRoom(...FILTERS...);
$eq = new Function<Room,Filter>() {
       public Filter call(Room $r) {
              return eq(Person.id, $r.getBuilding());
       }
}
List<Building> $buildings = manager().getBuilding(any(map($eq, $rooms)));
```

This will be translated to exactly two queries, no JOIN is done at all. The idea of how to deal with this data in Template is the following:

1. Pass both lists to the template
2. Refer to the entities via their id

This may look like this (Java):

```
put(“room”, $rooms);
put(“building”, $buildings);
```

and the template like this (XSL-T):

```
<xsl:for-each select=”//item:room”>
       <xsl:value-of select=”@number” />
       located in
       <xsl:value-of select=”//item:building[@id = current()/@building]” />
</xsl:for-each>
```

Explanation

map() takes the anonymous `Function` created above which extracts the id form a room and returns a comparing `Filter` (which states “*I am true if Person.id equals the number I got in my second parameter*”). Since that function is mapped on the $rooms we obtain a list of filters which are than connected by any() - which means, that all buildings that do have any one of the ids mentioned in the list constructed by *map()* are included in the result.

This is illustrated quite more beautiful in the following piece of pseudo-code:

```
rooms <- { any rooms where FILTER applies }
buildings <- { any building which id is in map(func(r) -> get-building-id(r), rooms) }
```

Imports

You will need to import the following functions:

```
import static de.fu.bakery.orm.java.filters.Filters.*;
import static de.fu.junction.functional.F.*;
```