MODEL Criteria/Criterion/RS

COMPARATORS

CRITERIA	SQL
EQUAL	=
NOT_EQUAL	<>
ALT_NOT_EQUAL	!=
GREATER_THAN	>
LESS_THAN	<
GREATER_EQUAL	>=
LESS_EQUAL	<=
LIKE	LIKE
NOT_LIKE	NOT LIKE
ILIKE	ILIKE
NOT_ILIKE	NOT ILIKE
CUSTOM	CUSTOM
DISTINCT	DISTINCT
IN	IN
NOT_IN	NOT IN
ALL	ALL
JOIN	JOIN
BINARY_AND	&
BINARY_OR	|
ASC	ASC
DESC	DESC
ISNULL	IS NULL
ISNOTNULL	IS NOT NULL
CURRENT_DATE	CURRENT_DATE
CURRENT_TIME	CURRENT_TIME
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP
LEFT_JOIN	LEFT JOIN
RIGHT_JOIN	RIGHT JOIN
INNER_JOIN	INNER JOIN

ResultSet (RS) Methods

getResource() getCursorPos()
setFetchmode() getRow()
getFetchmode() getRecordCount()
isLowerAssocCase() close()
next() get()
previous() getArray()
relative() getBoolean()
absolute() getBlob()
seek() getClob()
first() getDate()
last() getFloat()
beforeFirst() getInt()
afterLast() getString()
isAfterLast() getTime()
isBeforeFirst() getTimestamp()

E.g. (criteria):

```
$c = new Criteria();
$c->clearSelectColumns();
$c->addSelectColumn(userPeer::ID);
$c->addSelectColumn(userPeer::USERNAME);
$c->add(userPeer::USERNAME, "${username}%",
                            Criteria::LIKE);
$c->setLimit(5);
$c->setIgnoreCase(true);
$rs = userPeer::doSelectRS($c);
while($rs->next()){
    $users[$rs->getInt(1)] = $rs->getString(2);
}
```

E.g. (SQL):

```
$con=Propel::getConnection(DATABASE_NAME);
$sql="SELECT books.* FROM books WHERE NOT EXISTS
(SELECT id FROM review WHERE book_id = book.id)";
$stm = $con->createStatement();
$rs=$stm->executeQuery($sql, ResultSet::FETCHMODE_NUM);
```

CRITERIA

To find records, use a **Criteria object** in conjunction with one of the **Peer's methods**: doSelect(), doSelectOne(), doSelectJoinXXX() , doSelectJoinAll(), retrieveByPk()

GETTING A SPECIFIC RECORD

retrieveByPk(<*value | array*>) *Peer method*

SINGLE PRIMARY KEYS

```
// gets the record with primary key = 3
$obj = BookPeer::retrieveByPK(3);
```

COMPOST PRIMARY KEYS

```
// gets the book_id=1,author_id=2 record
$obj = BookAuthorXrefPeer::retrieveByPK(array(1,2));
```

Note: the order of keys is critical and must correspond to the order in which the columns are defined in the XML (schema).

GETTING ALL RECORDS

doSelect(<*empty criteria*>) *Peer method*

```
$questions = QuestionPeer::doSelect(new Criteria());
```

GETTING ONE RECORD

doSelectOne(<*object*>) *Peer method*

```
$c = new Criteria();
$c->add(UserPeer::NICKNAME, $nickname);
$user = UserPeer::doSelectOne($c);
```

SPECIFYING CONDITIONS USING COMPARATORS

add(<*column*>, <*value*>, <*comparator*>) *Criteria method*

```
// default comparator: =
$c = new Criteria();
$c->add(TableNamePeer::ID, 17);
$obj = TableNamePeer::doSelect($c);
```

```
// id <> 17
$c = new Criteria();
$c->add(TableNamePeer::ID, 17, Criteria::NOT_EQUAL);
$obj = TableNamePeer::doSelect($c);
```

CUSTOM:

```
$c = new Criteria();
$c->add(tableNamePeer::RATING, "rating=rating +1", Criteria::CUSTOM);
```

LIMIT THE NUMBER OF RECORDS RETURNED

setLimit(<*value*>) *Criteria method*

```
// first 5 results
$c = new Criteria();
$c->setLimit(5);
```

ORDERING RECORDS

addAscendingOrderByColumn(<*column*>)
addDescendingOrderByColumn(<*column*>) *Criteria methods*

```
// first 10 authors, alphabetically
$c = new Criteria();
$c->setLimit(10);
$c->addAscendingOrderByColumn(AuthorPeer::LAST_NAME);
```

CASE SENSITIVITY

setIgnoreCase(<*true|false*>) *Criteria method*

```
// find all authors named "max", case-insensitive
$c = new Criteria();
$c->add(AuthorPeer::FIRST_NAME, "max");
$c->setIgnoreCase(true);
```

JOINS

addJoin(<*column_T1*>, <*column_T2*>, <*LEFT_JOIN|RIGHT_JOIN|INNER_JOIN*>) *Criteria method*

```
$c = new Criteria();
$c->addJoin(UserPeer::ID, InterestPeer::USER_ID, Criteria::LEFT_JOIN);
$c->add(InterestPeer::QUESTION_ID, $this->getId());
$obj = UserPeer::doSelect($c);
```

RETURNING SPECIFIC COLUMNS

clearSelectColumns()
addSelectColumn(<*column*>) *Criteria methods*

```
$c = new Criteria();
$c->clearSelectColumns();
$c->addSelectColumn(userPeer::ID);
$c->addSelectColumn(userPeer::USERNAME);
$c->add(userPeer::USERNAME, "${username}%", Criteria::LIKE);
$c->setLimit(5);
$c->setIgnoreCase(true);
$rs = userPeer::doSelectRS($c);
while($rs->next()){   $users[$rs->getInt(1)] = $rs->getString(2); }
```

ADD AN ALIAS TO COLUMN (AS clause)

addAsColumn(<alias>, <ALIAS(TableNamePeer::ID>) *Criteria method*

```
$c = new Criteria();
$c->addAsColumn("numUsers", "COUNT(".UserPeer::ID.")");
```

CRITERION

SPECIFYING MULTIPLE CONDITIONS FOR A COLUMN

getNewCriterion(<*column*>, <*value*>, <*comparator*>) *Criteria method*

addOr(<*condition*>) *Criteria method*

addAnd(<*condition*>) *Criteria method*

```
$c = new Criteria();
$criterion = $c->getNewCriterion(AuthorPeer::FIRST_NAME, "Leo%", Criteria::LIKE);
$criterion->addOr($c->getNewCriterion(AuthorPeer::FIRST_NAME, "Leonardo", Criteria::NOT_EQUAL));
$c->add($criterion);
```

COMBINING CRITERIA OBJECTS

combine Criterion objects in order to specify logical relationships between clauses

```
// Find all authors with first name "Leo" OR last name of "Tolstoy", "Dostoevsky", or "Bakhtin"
$c = new Criteria();
$cton1 = $c->getNewCriterion(AuthorPeer::FIRST_NAME, "Leo");
$cton2 = $c->getNewCriterion(AuthorPeer::LAST_NAME, array("Tolstoy", "Dostoevsky", "Bakhtin"), Criteria::IN);
$cton1->addOr($cton2);  // combine them
$c->add($cton1);  // add to Criteria
```

Criteria Class

getIterator ()
getMap ()
clear ()
addAsColumn ($name, $clause)
getAsColumns ()
getColumnForAs ($as)
addAlias ($alias, $table)
getTableForAlias ($alias)
keys ()
containsKey ($column)
setUseTransaction ($v)
isUseTransaction ()
getCriterion ($column)
getNewCriterion ($col, $val, $comp=null)
getColumnName ($name)
getTablesColumns ()
getComparison ($key)
getDbName ()
setDbName ($dbName=null)
getTableName ($name)
getValue ($name)
get ($key)
put ($key, $value)
putAll ($t)
add ($p1, $value=null, $comparison=null)
addJoin ($left, $right, $operator=null)
getJoins ()
getJoinL ()
getJoinR ()
setAll ()
setDistinct ()
setIgnoreCase ($b)
isIgnoreCase ()
setSingleRecord ($b)
isSingleRecord ()
setLimit ($limit)
getLimit ()
setOffset ($offset)
getOffset ()
addSelectColumn ($name)
getSelectColumns ()
clearSelectColumns ()
getSelectModifiers ()
addGroupByColumn ($groupBy)
addAscendingOrderByColumn ($name)
addDescendingOrderByColumn ($name)
getOrderByColumns ()
clearOrderByColumns ()
clearGroupByColumns ()
getGroupByColumns ()
getHaving ()
remove ($key)
toString ()
size ()
equals ($crit)
addHaving (Criterion $having)
addAnd ($p1, $p2=null, $p3=null)
addOr ($p1, $p2=null, $p3=null)

USEFUL LINKS

http://propel.phpdb.org
propel reference

http://creole.phpdb.org
creole reference

http://propel.jondh.me.uk
online tool for convert pseudo-SQL to criteria

NOTE 1: Propel stores criteria in a hashtable, for performance reasons.

NOTE 2: some of the examples came from the Criteria Class Reference on the Propel website.