CROSSFIRE O/R for Eclipse

USER MANUAL

http://www.crossfire.jp

Index

1. CROSSFIRE O/R　とは?

“CROSSFIRE O/R は、JAVA のプログラムを人間の代わりに書くツールです。
あなたは、このプロダクトを使うことにより、JDBC による SQL を実行する JAVA プログラムを自動生成することができます。

1.1.　SQL に基づいた簡単なソフトウェア

このプログラムの使用は SQL に基づいています。あなたが、どんなプログラミング言語を使う場合でも、おそらくシステムを設計する際には DBMS (Database Management System)を使うことになるでしょう。そして、設計はデータ設計とそのデータ設計に対して使われる SQL と非常に強い関係のあるものになります。

もし、あなたが SQL を利用するシステムのデザインを行った経験があるのならば、このプロダクトを使うことはとても簡単でしょう。なぜならば、このソフトウェアは SQL に基づいた仕様になっているからです。

あなたは、ほぼ、直感的にこのソフトウェアを使うことが出来るでしょう。

1.2.　ソフトウェアがあなたの代わりに複雑なソースコードを書いてくれます

Java のプログラムの開発時に、基本的な部分を作成する際、非常に複雑なソースコードを書かねばなりません。SQL を設計した後、その SQL を実行するモジュールを実装するのに実際、かなりの時間がかかります。

しかし、CROSSFIRE O/R を使うことによってほんの 1 秒足らずでそれらのモジュールを自動生成することが出来ます。

1.3.　コンピュータは単純ミスを犯しません

人間がプログラムを作成すると、沢山の単純ミスを犯し、またそれによるバグをチェックするのに多大な時間と労力を費やします。しかし、コンピュータは絶対に単純ミスを犯しません。

CROSSFIRE O/R を使うことによって、人間が犯してしまうミスを回避することが可能です。

2. システム環境

CROSSFIRE O/R は Eclipse プラグインとして実装されているため、必要となるシステム環境は Eclipse に準拠したものになります。

※Eclipse に関しては下のページを参照して下さい。

Eclipse official page

http://www.eclipse.org/

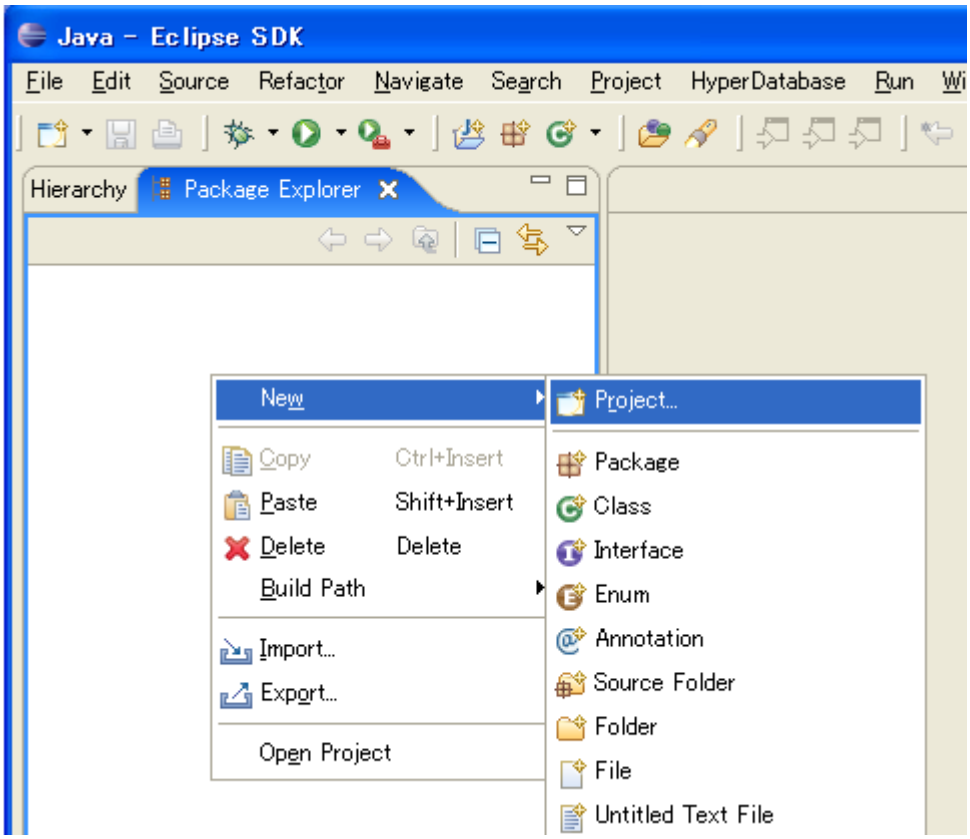
Service	Software	Comment
OS	Windows2000 or XP	Where Eclipse3.1 works
Java SDK	J2SE ver1.4 2 or later	
Eclipse	Version 3.X	

3. プロジェクトのセットアップ

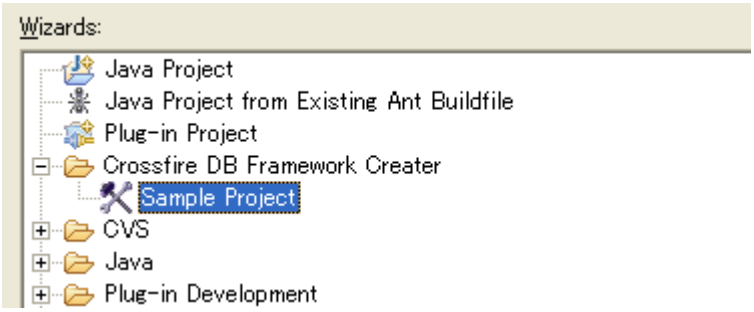
3.1. ウィザードから新規プロジェクトを作成

New Project Wizard から新規プロジェクトを作成することが出来ます。.

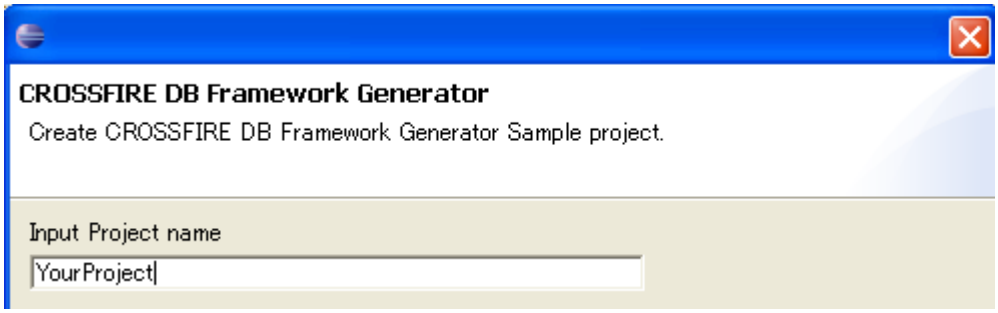
プロジェクトを作成するには、“Navigator” もしくは “Package Explorer”から “New” -> “Project”と選択してください。



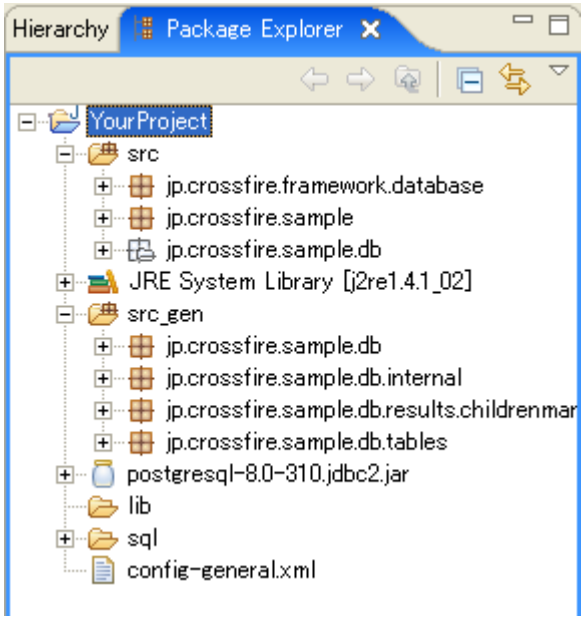
すると、ダイアログが現れますので “CROSSFIRE O/R の中にある”“Sample Project”を選択してください。



そして、プロジェクト名を聞かれますので任意の名前を入力してください。.



“Finish”ボタンを押した後、新しいプロジェクトが自動的に作成されます。



このプロジェクトは、CROSSFIRE O/R に関する設定は自動的に行われています。
しかし、次節以降で CROSSFIRE O/R の動作について詳しく知るため、解説を行います。

3.2. 必要な設定

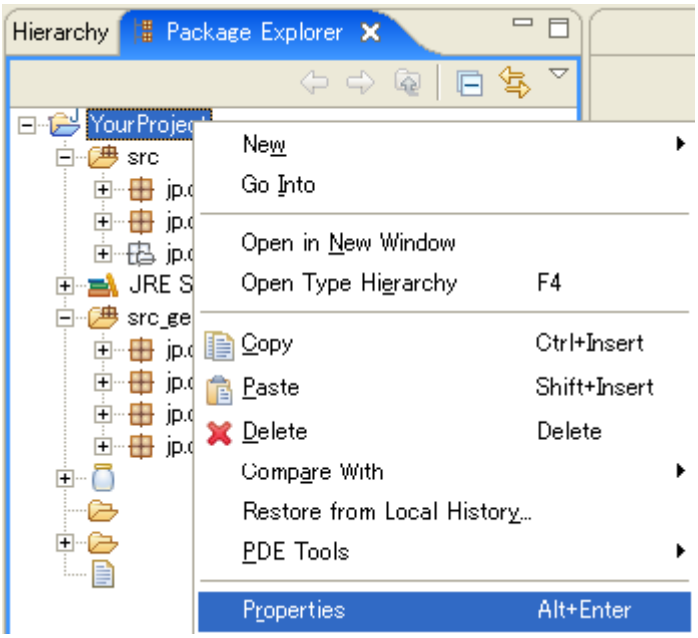
CROSSFIRE O/R に対して必要な設定は大きく分けて次の二つです。

- Project の設定
- フレームワーク

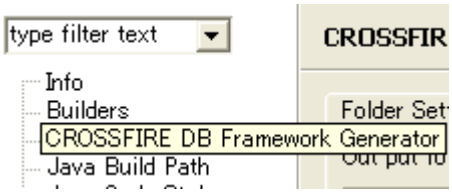
そして、これらで全てです。

3.3. Project の設定

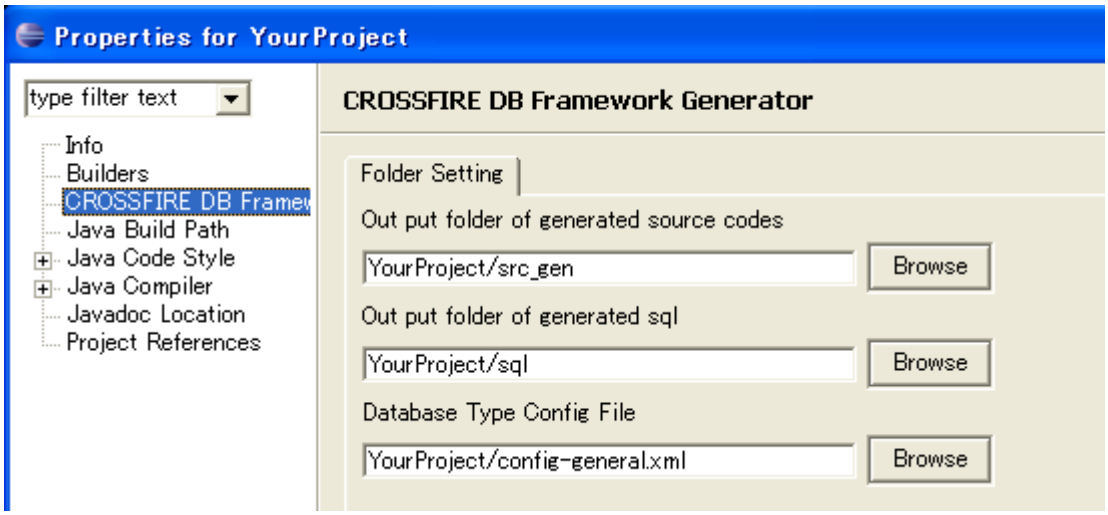
プロジェクトを選択後、右クリックし、メニューから “Property”を選択します。



「プロジェクトのプロパティ」のカテゴリ中の “CROSSFIRE O/R” を選択します。



すると、次のような設定項目が現れます。



設定項目は全部で３項目あります。.

“Out put folder of generated source codes”

CROSSFIRE O/R によって自動生成されるソースコードの出力先フォルダを設定します。

“Out put folder of generated sql”

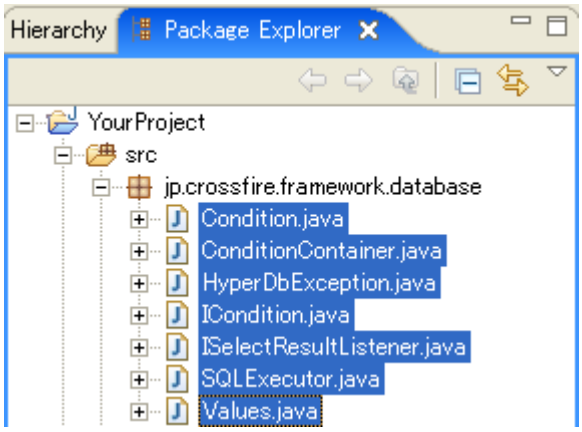
CROSSFIRE O/R によって自動生成される DDL の出力先フォルダを設定します。

“Database Type Config File”

データベースの型に関する設定情報を記述した XML ファイルの場所を指定します。
この XML ファイルで SQL の型と Java の型（クラス）のマッピングを設定することが出来、これにより CROSSFIRE O/R は様々なデータベースエンジンに対して対応することが出来ます。

3.4. フレームワーク

CROSSFIRE O/R によって自動生成されたソースコードを動作させるためにはフレームワークが必要です。しかし、このフレームワークは非常に小さなものになっています。
下図で表示された７つのファイルがフレームワークの全てです。



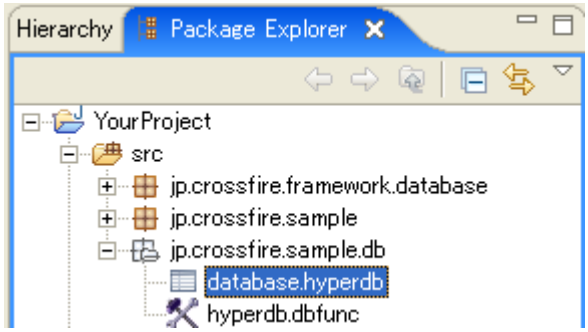
フレームワークは、CROSSFIRE O/R が自動生成するソースコードの出力先ではなく、手動で書くためのソースフォルダに配置してください。
また、このフレームワークはユーザーがカスタマイズできるようになっています。

4. テーブル情報の編集

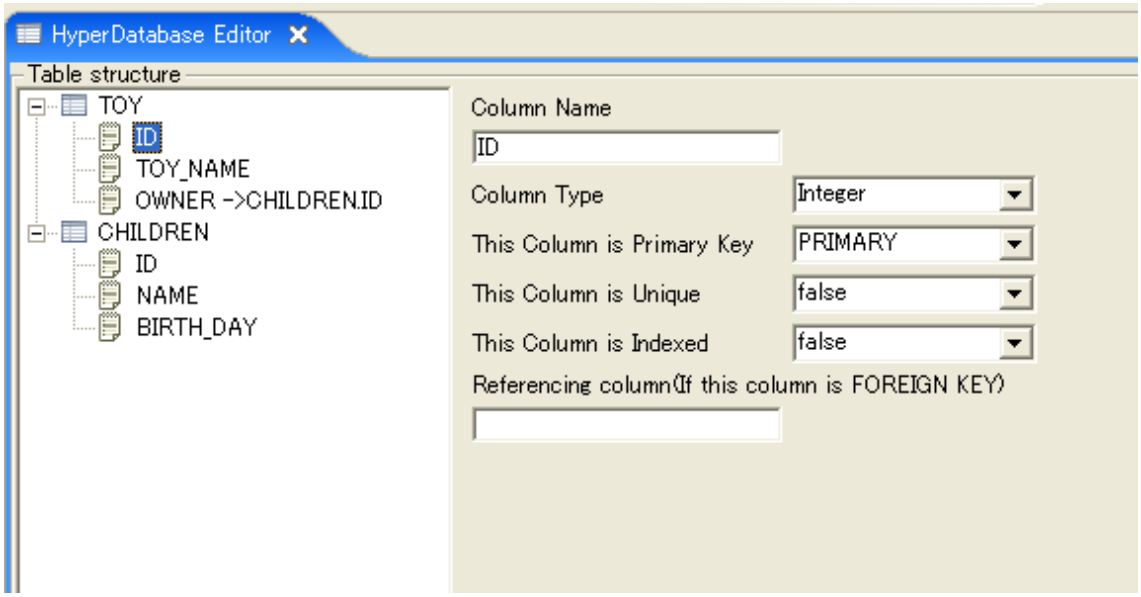
CROSSFIRE O/R を利用するためには、最初にテーブル設計の情報の入力が必要です。
また、テーブル設計の情報の入力後、DDL (SQL)を自動生成することが可能になります。

4.1. サンプルプロジェクトの “Hyper Database Editor File” を見る

サンプルプロジェクト作成後“Hyper Database Editor File”がソースフォルダに入っています。下図の“database.hyperdb” ファイルがそのファイルです。



ファイルをダブルクリックすることで、専用エディタで開くことが出来ます。



入力する項目は以下の通りです。

TABLE

Item	Explanation	Comment
name	Name of the Table	Necessary

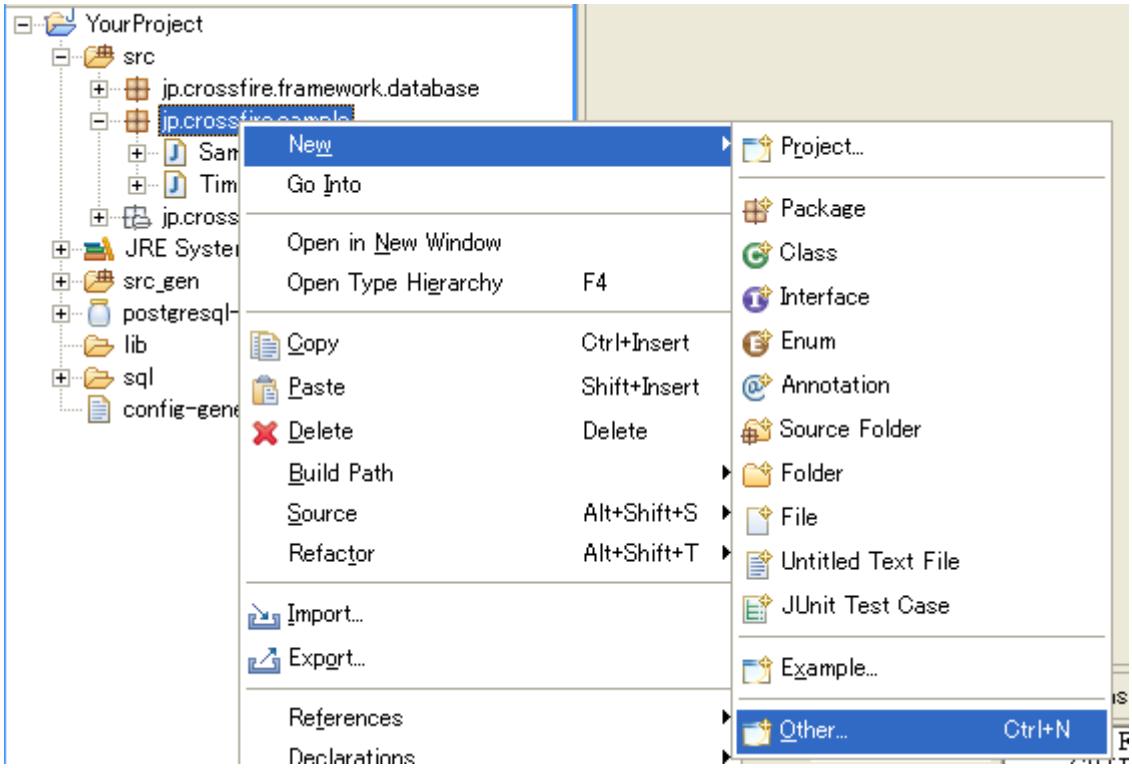
COLUMN

Item	Explanation	Comment
Column Name	Name of the Column	Necessary
Column Type	Type of the Column	Necessary
The Column is the Primary Key	If this column is the Primary Key, enter “PRIMARY”	
This Column is Unique	Whether this column is unique or not.	Necessary
This Column is Indexed	Whether this column is indexed or not.	Necessary
Referencing column(If this column is FOREIGN KEY)	If this column is the Foreign Key, enter the table and column this column refers. Ex. ‘CHILDREN.ID’	

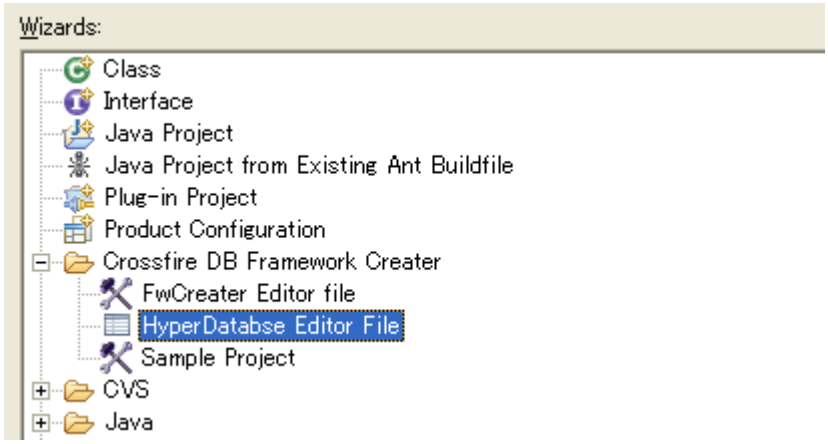
4.2. “Hyper Database Editor File”の新規作成

“Hyper Database Editor File”をプロジェクト上に作成します。

任意の“package”もしくはソースフォルダを選択後、右クリックし“New” -> “Other”と選択します。



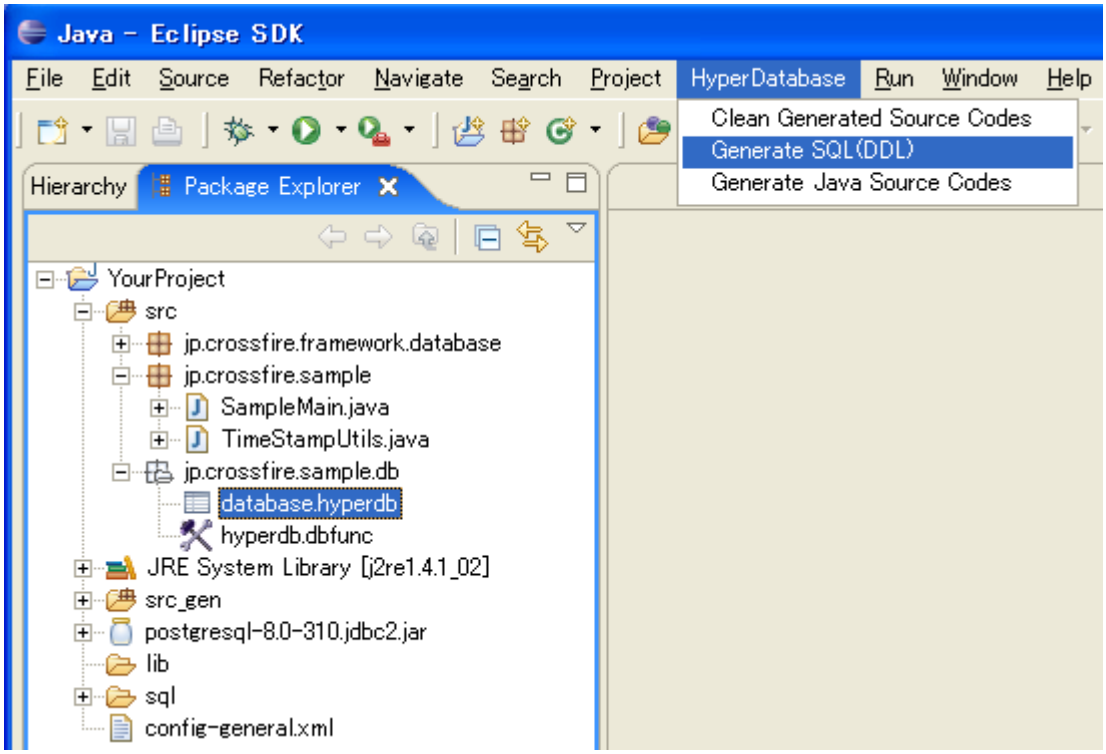
次に‘Hyper Database Editor File’を選択します。



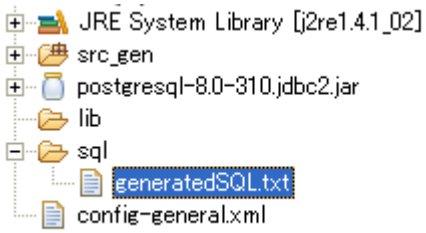
選択後、ダイアログボックスが表示されますので、“Finish” ボタンを押します。

4.3. SQL の自動生成

Eclipse のメニューから “Hyper Database” -> “Generate SQL (DDL)” と選択します。



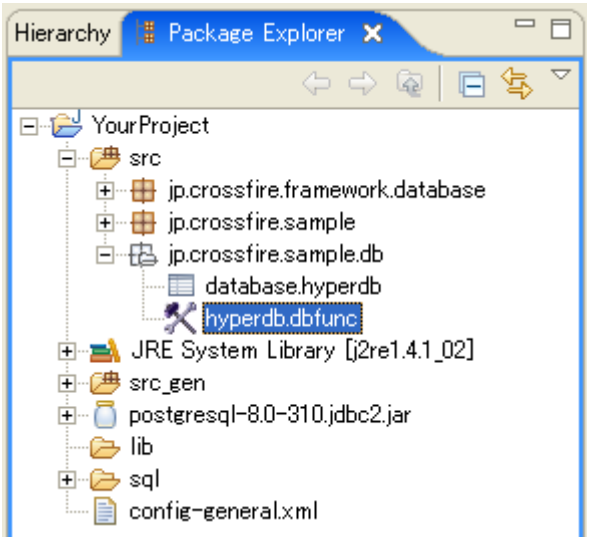
“generatedSQL.txt” と言う名前のファイルが、プロジェクトのプロパティで設定した“Out put folder of generated sql”に出力されます。



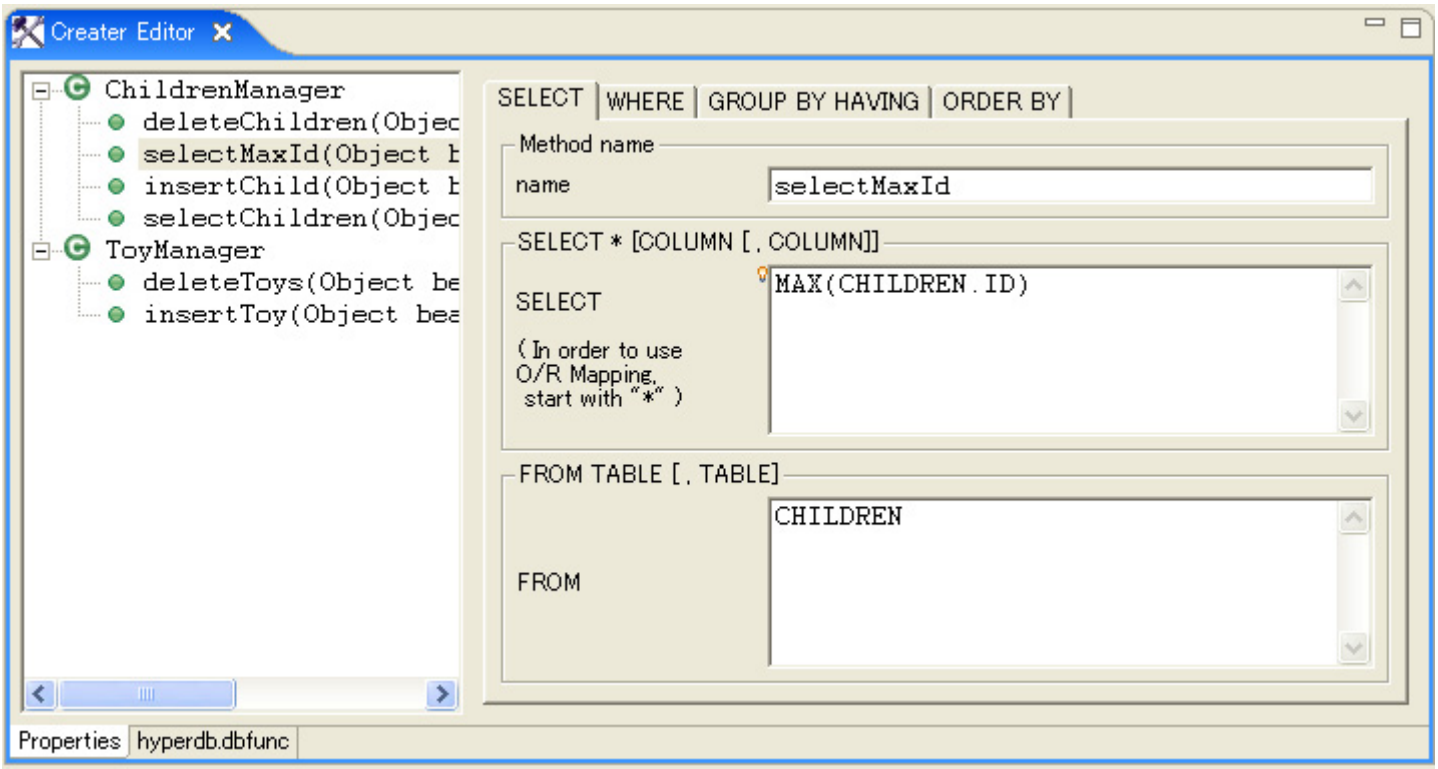
5. SQL を実行する Java メソッドの設定方法

5.1. サンプルプロジェクトの “FwCreator Editor File” を見る

サンプルプロジェクト作成後、 “FwCreator Editor File”がソースフォルダにあります。このファイルは下図の“hyperdb.dbfunc”がそのファイルに当たります。



“hyperdb.dbfunc”をダブルクリックすることで、専用エディタで開くことが可能です。



5.2. FwCreator Editor とは？

FwCreator は、SQL を実行する Java クラスおよびメソッドを設定するためのエディタです。 実行できる SQL は、SELECT, UPDATE, INSERT, DELETE の４タイプです。
エディタの各フィールドに値を入力するだけで SQL を設計していくことができます。また、各テキストフィールドは、簡単なスクリプトになっています。

5.3. Script in FwCreator's のテキストフィールドについて

5.3.1. 動的変数

メソッド実行時に、プログラムからメソッドに値を入力することが出来ます。
(例： WHERE 句の条件文で比較する値など)

For example 次のような SQL を設定したとします。.

SELECT * FROM TOY WHERE ID = '**$ID**'

$ID は、動的変数で、実際の値は、Java プログラムの実行時に決まります。
'$' から始まるものは動的変数で、Java プログラムからメソッドを実行する際に動的に入力された値に変換されます。

5.3.2. サブクエリー

SELECT 文の SQL を実行する、同じクラス内の他の Java メソッドををサブクエリーとして呼び出すことが出来ます。

For example 次のような SQL を設定したとします。.

UPDATE CHILDREN SET ID = (**@selectMax**) + 1

'selectMax' は、他の SELECT 文の SQL を実行する Java メソッド名です。'@' から始まる単語はサブクエリーを意味します. 仮に selectMax が実行する SQL が下の SQL だとします。

SELECT MAX(CHILDREN.ID) FROM CHILDREN

その場合、最初の SQL は次のように展開されます。

```
UPDATE CHILDREN SET ID = (SELECT MAX(CHILDREN.ID) FROM CHILDREN) + 1
```

5.4. Hyper DB View について

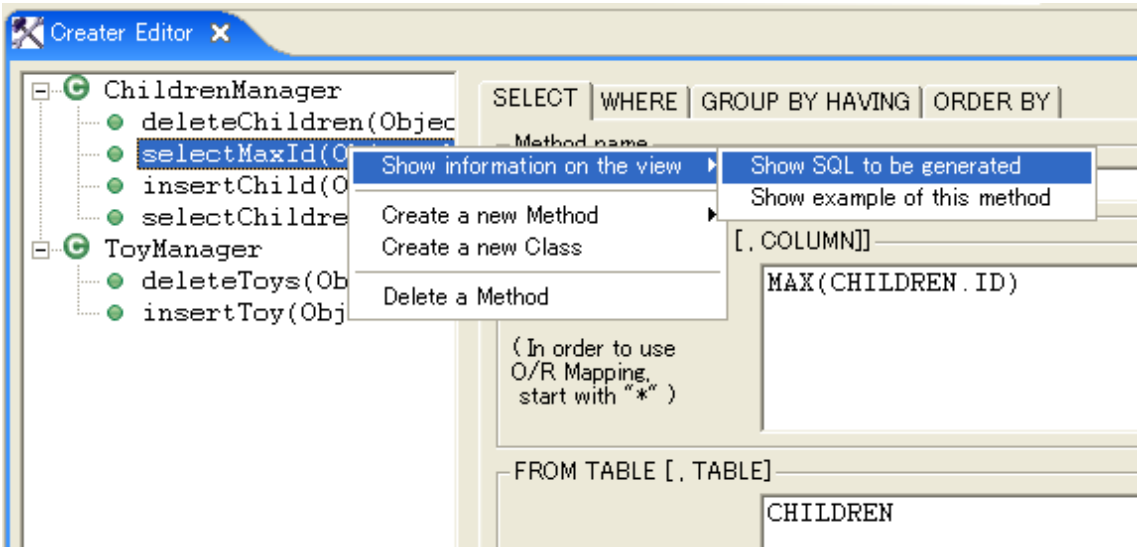
“Hyper DB View” はとてもパワフルなツールです。このビューにより

- SQL 全体の表示
- 自動生成された Java クラスのメソッドを使ったサンプルソースコード表示

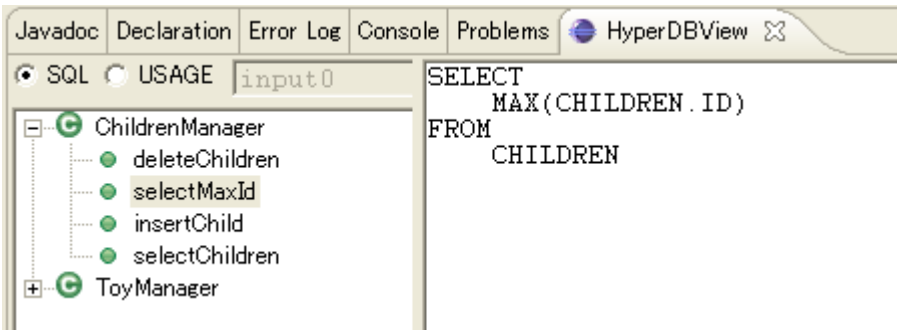
が可能です。

5.4.1. SQL 全体を表示する

“Hyper DB View”で SQL 全体を表示させることが出来ます。FwCreator Editor の任意のメソッドを右クリックし、“Show information on the view” -> “Show SQL to be generated”と選択します。

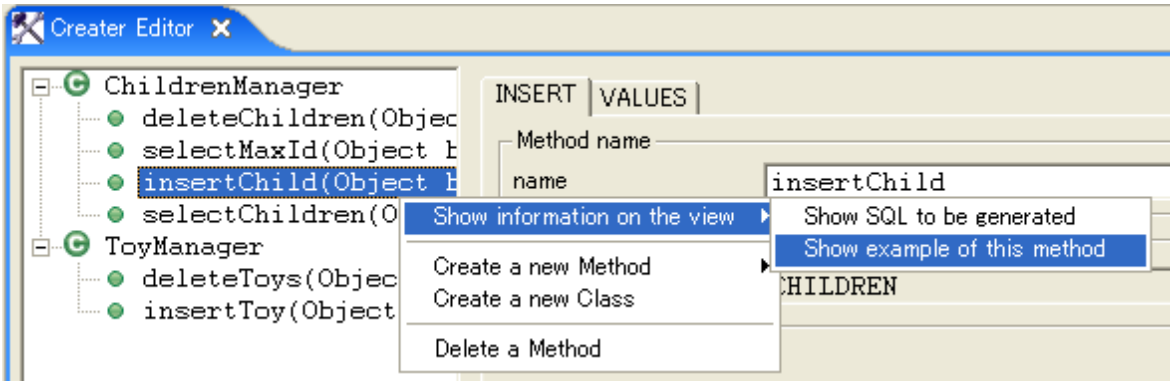


すると “Hyper DB View”に設定中の SQL が表示されます。

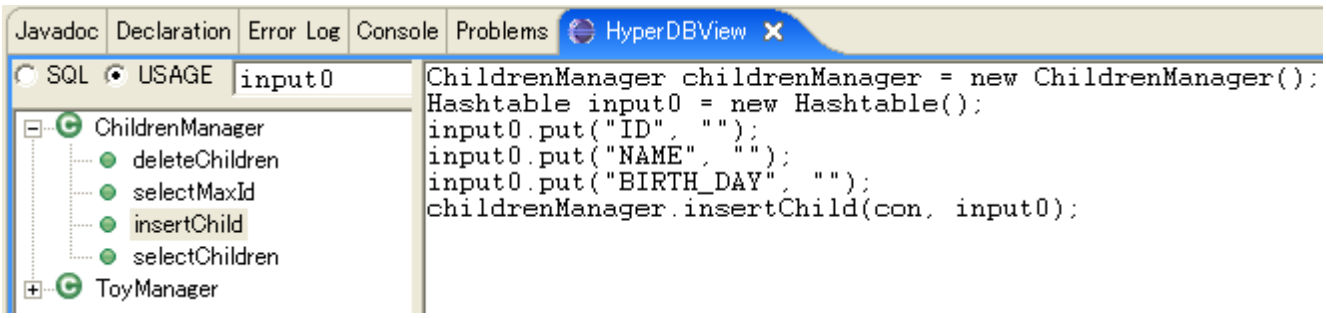


5.4.2. サンプルソースコードの表示

自動生成された Java クラスのメソッドを使ったサンプルソースコード表示が可能です。エディタ上で任意のメソッドを選択後 “Show information on the view” -> “Show example of this method”と選択します。



すると、“Hyper DB View” 上にサンプルソースコードが表示されます。

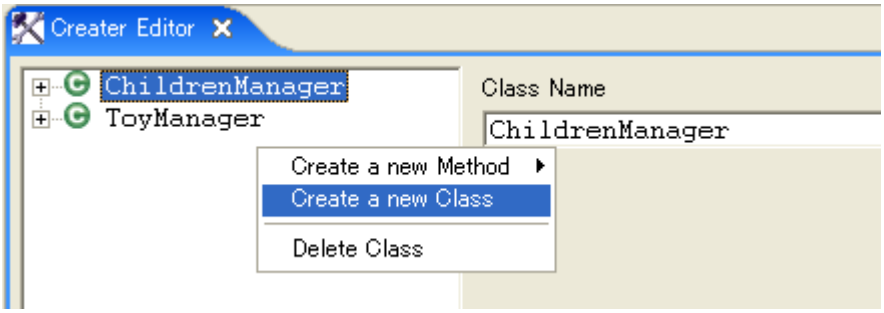


このとき、SQL 内で使われている動的変数がリストアップされてプログラム内に表示されます。
よってプログラムを開発する際には大変に役立ちます。
また、このサンプルコードをコピー&ペーストして使うことも可能です。

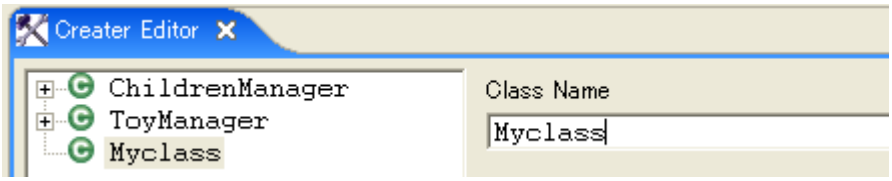
6. SQL を実行するメソッド及びクラスの作成

6.1. Java クラスの新規作成

FwCreator Editor の中で右クリックをし、“Create a new Class”を選択することで新規 Java クラスを作成することが出来ます。

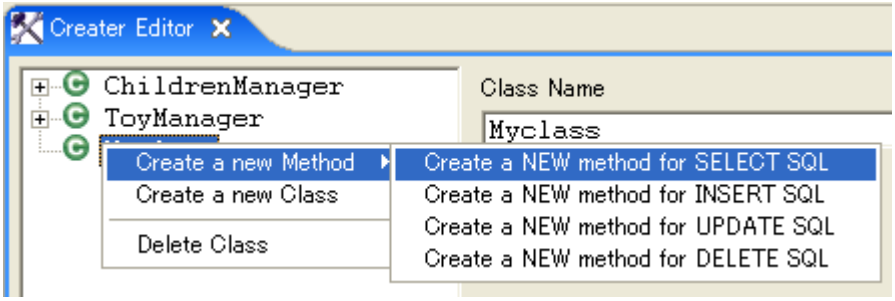


下図のようにクラスを選択し、名前を変更することが可能です。



6.2. SELECT 文を実行するメソッドの作成

作成したクラスで、SELECT 文を実行するメソッドを作成したいクラスを選択し、“Create a new Method” -> “Create a NEW method for SELECT SQL”と選択します。



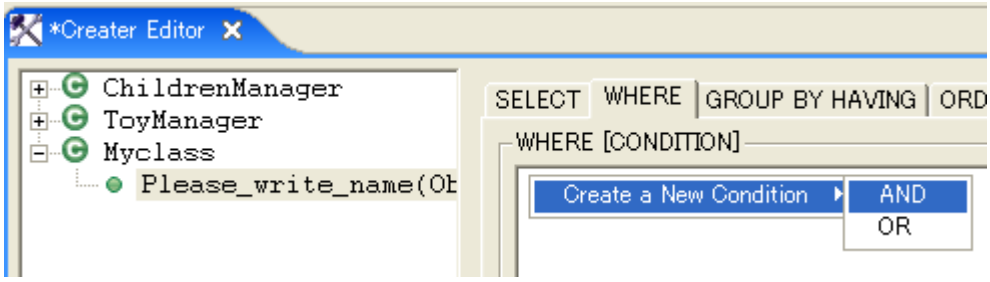
すると、SELECT 文を実行するメソッドが作成されます。
次に、メソッドの各フィールドを入力していきます。 各フィールドでは変数の使用が可能です。
(‘$’で始まる文字列は変数として扱われます。
次節では、WHERE 区の入力方法が特殊なので、入力方法について説明します。

6.3. ‘WHERE’ 句の設定

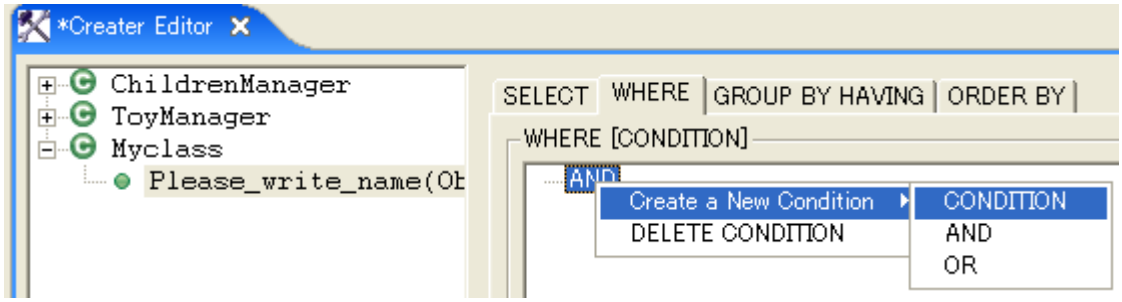
‘WHERE’ 句は、条件のツリー構造として表現されます。

初期状態では、何も存在しないため、まずは最初の条件を追加します。

FwCreator Editor 上で右クリックし “Create a New Condition” -> “AND” (“OR”)を選択します。

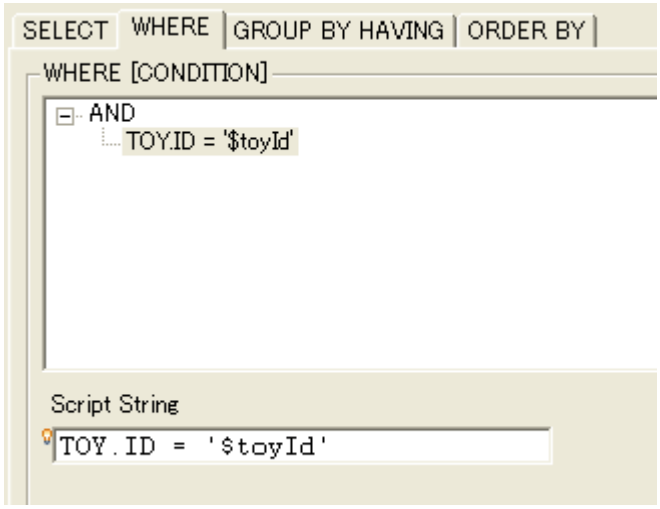


すると、最初の条件が作られます。次に条件を付けたしていきます。“Create a New Condition” -> “CONDITION”を選択します。



すると、条件が AND の中に作られます。

条件を選択することにより、条件を編集することが出来ます。

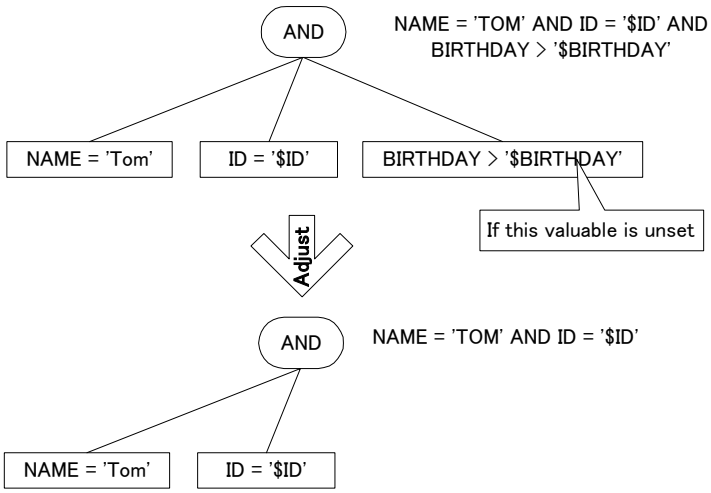


6.4.　条件文の自動調整機能

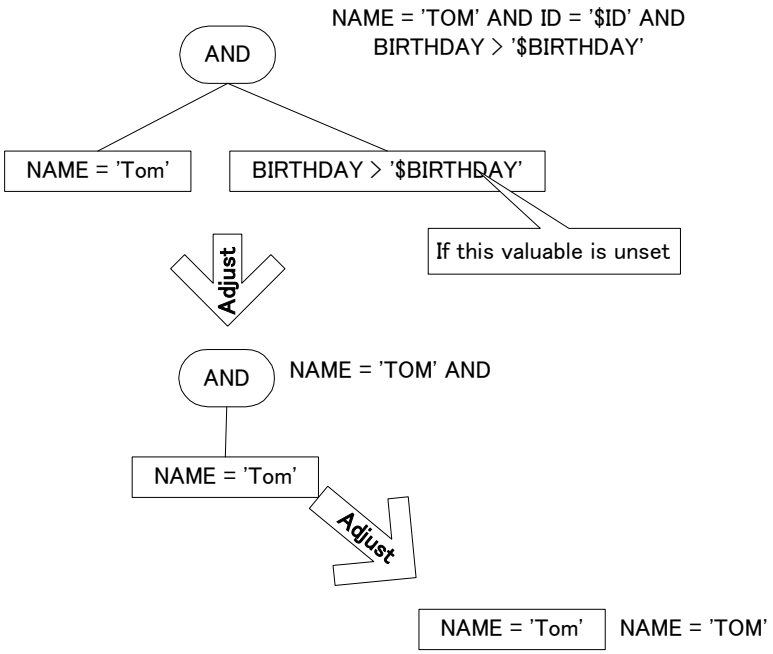
この機能は非常にパワフルな機能です。条件の内容で変数を使うことができますが、このメソッドが呼び出されたときに、変数が設定されない場合はどのようなことが怒るのでしょうか？

そのような場合には、自動生成されたプログラムが自動的に条件文を調整します。

Ex.1. 条件が３つ AND の中にあり、１つの条件の動的変数が未設定の場合

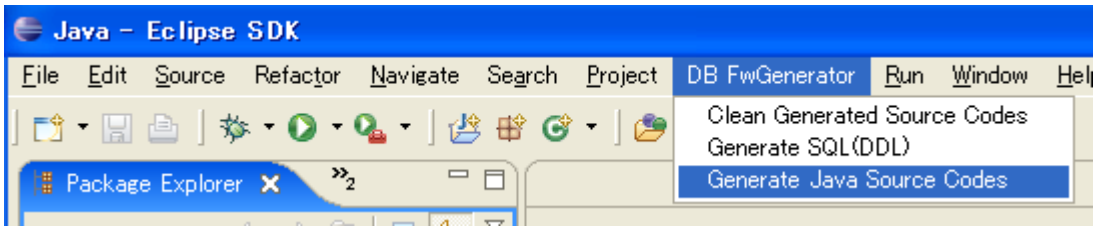


Ex.2. 条件が３つ AND の中にあり、２つの条件の動的変数が未設定の場合



6.5. Java ソースコードの自動生成

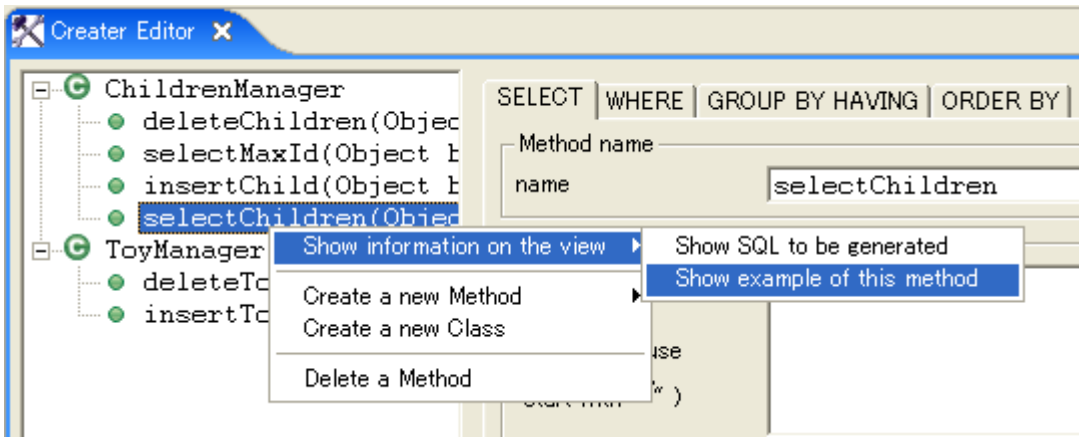
Eclipse のメニューから “DB FwGenerator” -> “Generate Java Source Codes”と選択します。



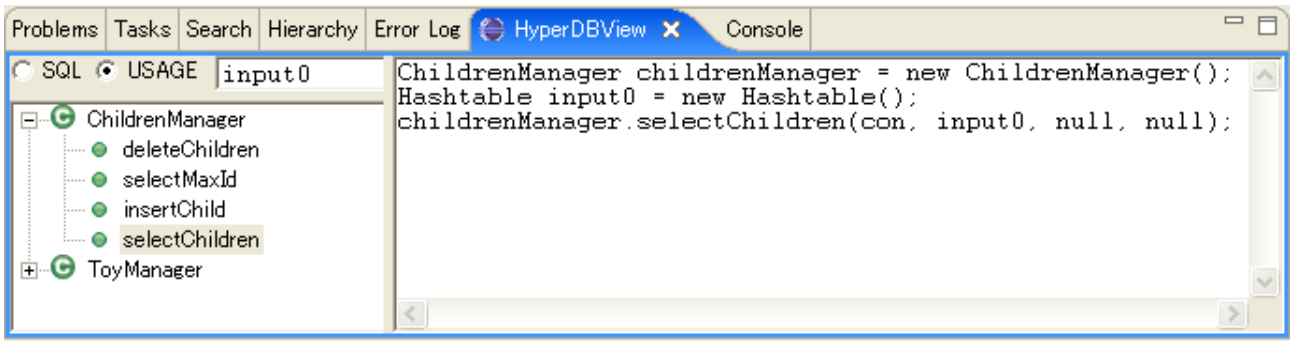
数秒後、プロジェクトの設定の“Out put folder of generated source codes”で指定したフォルダにそーすこーどが　出力されます。

これらのメソッドを利用するためには、どのように利用するかを知る必要があります。

“HyperDb View”を利用することで、メソッドの利用方法を知ることが出来ます。利用したいメソッドを選択し “Show information on the view” -> “Show example of this method”と選択します。



すると “Hyper DB View” が立ち上がり、メソッドの利用方法を表示します。



6.6. Java プログラム上での SELECT 文の結果取得

SELECT SQL を実行するメソッドは、 “java.util.List”型の戻り値を返します。
このリストは “java.util.Hashtable”型のオブジェクトのリストとなっており、下のソースコードのように結果を取得することが出来ます。

Sample Code

```
// Select
ChildrenManager childrenManager = new ChildrenManager();
Hashtable input = new Hashtable();
List result = childrenManager.selectChildren(con, input, null, null);

// You can get data from List of Hashtable
System.out.println("---------Check RECORDSET RESULT--------");
Iterator it = result.iterator();
while(it.hasNext()){
    Hashtable hash = (Hashtable)it.next();
    System.out.print(hash.get("name"));
    System.out.print(" " + hash.get("birth_day"));
    System.out.println(" " + hash.get("toy_name"));
}
```

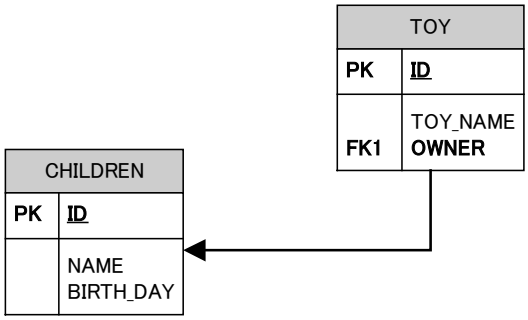
6.7. Java Bean Mapping (OR Mapping)

自動生成された SELECT SQL を実行するメソッドから結果を取得する方法はもう一つあります。

自動生成されたクラスは、データベースのテーブルに相当する JavaBean のリストを持っています。この JavaBean は、外部キーによる他テーブルへの参照と逆参照を持っています。

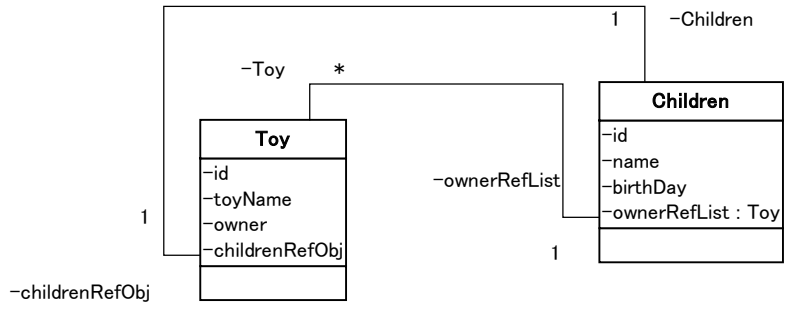
このサンプルでは、２つのテーブルがあり、TOY テーブルの OWNER カラムが CHILDREN テーブルを参照する外部キーとなっています。

Structure of Sample's DATABASE TABLES



このような構造に対して CROSSFIRE O/R は下図のような JavaBean を自動生成します。
CHILDREN テーブルに相当する Children Bean は、複数の Toy Bean を参照するための Toy 型のリストを持ち、TOY テーブルに相当する Toy Bean は Children 型の Children Bean に対する参照を持ちます。

Java Bean



JavaBean による結果の取得方法のサンプルは下のようになります。

Sample Code

```
// Select
ChildrenManager childrenManager = new ChildrenManager();
Hashtable input = new Hashtable();
childrenManager.selectChildren(con, input, null, null);

// You can also get data from Mapped Java Bean
// JavaBeans are automatically generated and Mapping
// Logic is generated too.
System.out.println("---------Check ORMAPPING RESULT--------");
Iterator it = childrenManager.getChildren().iterator();
while(it.hasNext()){
    Children child = (Children)it.next();
    System.out.println("Child Name : " + child.getName());

    Iterator toyIt = child.getOwnerRefList().iterator();
    while(toyIt.hasNext()){
        Toy toy = (Toy)toyIt.next();
        System.out.print("   having: " + toy.getToyName());
        System.out.println("   owner-> "
            + toy.getChildrenRefObj().getName());
    }
}
```

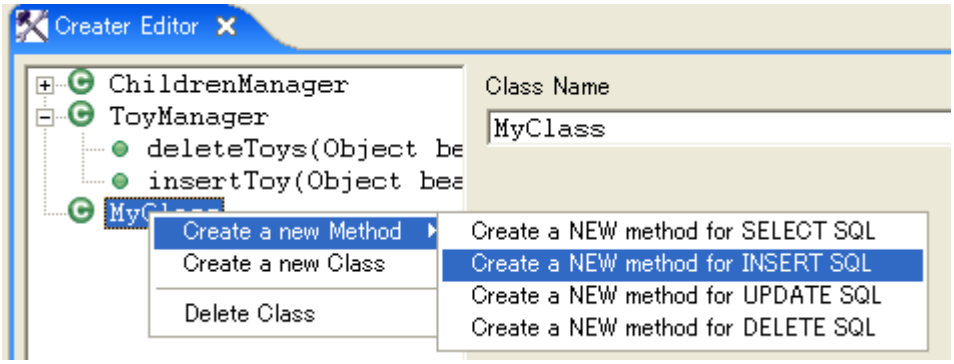
6.8. ORマッピングを使うために必要な条件

ORマッピングを使うためには次の条件を満たさなければなりません。.

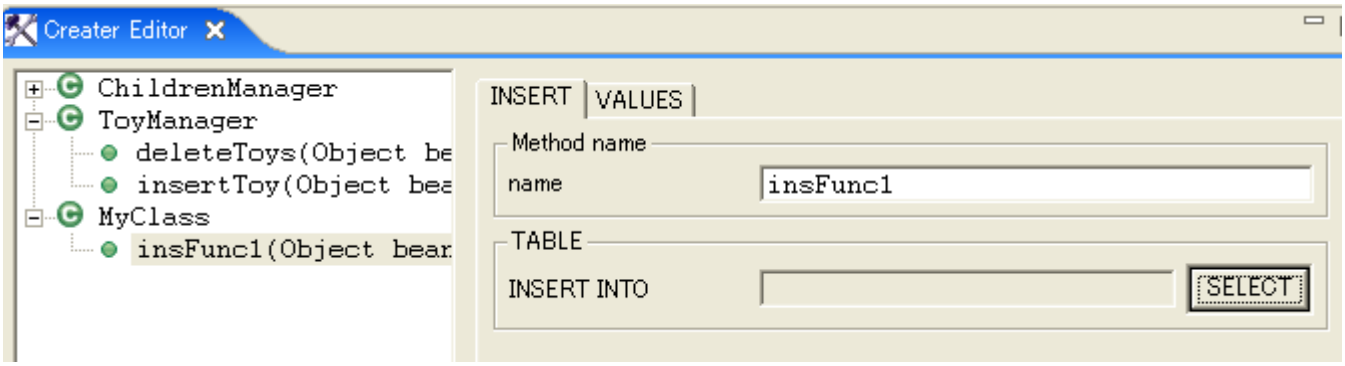
- **SELECT文のSELECT句が'*'から始まること**
- **'FROM'句の中で、MAPするテーブルが指定されていること**

6.9. INSERT文を実行するメソッドの作成

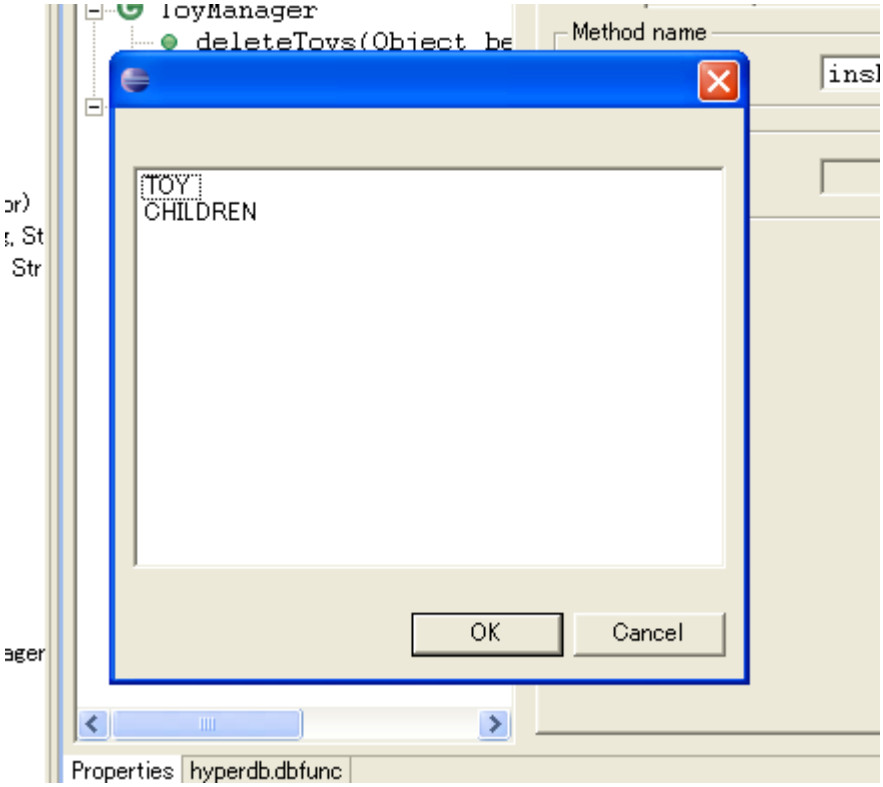
INSERT文を実行するメソッドを作成したいクラスを選択後、右クリックし、"Create a new Method" -> "Create a NEW method for INSERT SQL"を選択します。



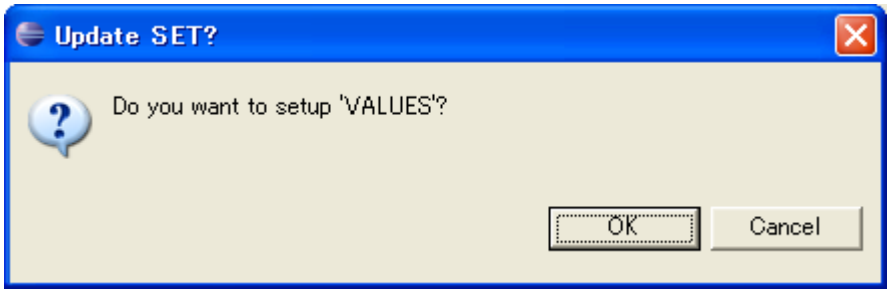
すると、 INSERT文を実行するメソッドが作成されます。
次に INSERTを行う対象となるテーブルを入力します。 "SELECT" ボタンを押します。



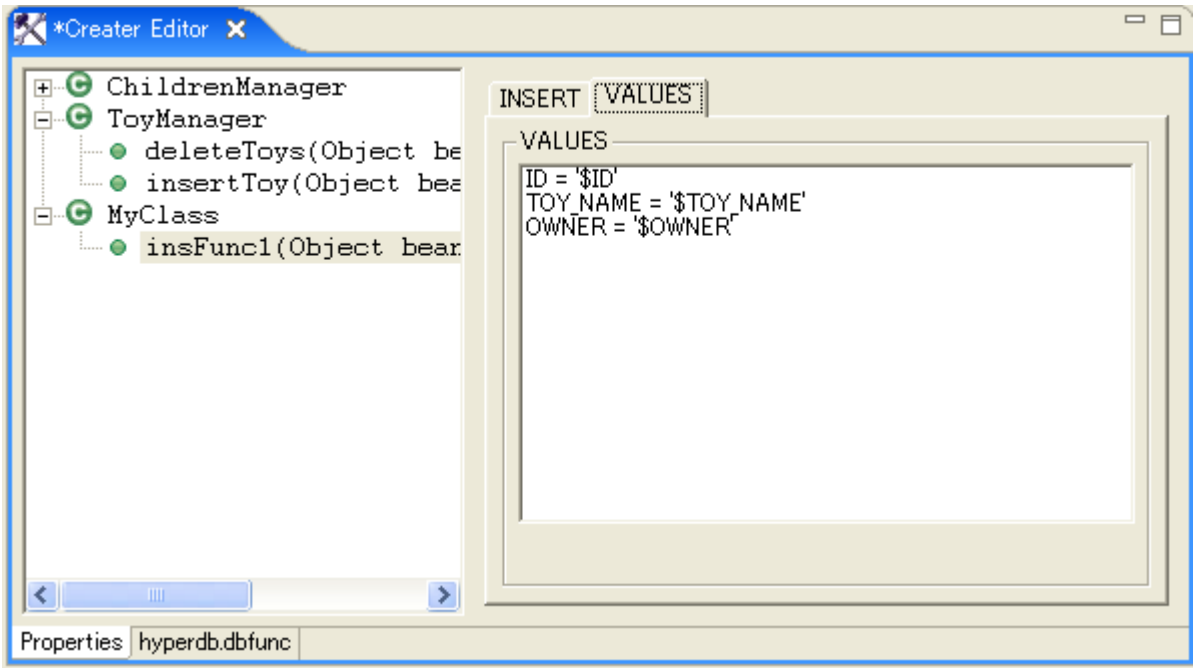
すると、テーブルを選択するダイアログが表示されますので、テーブルを選択します。



テーブル選択後、テーブルに対する VALUES 句のデフォルトの設定を行うかどうか聞くダイアログが出ますので、‘OK’ ボタンを押します。

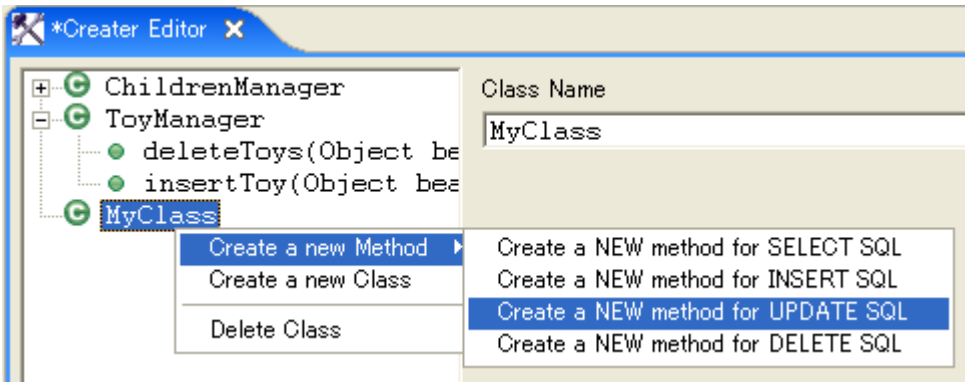


‘OK’ボタンを押すことによって、INSERT 文の‘VALUES’句が自動的に設定されます。
サブクエリーや関数の値をカラムに入力する場合は、スクリプトを編集してカスタマイズすることも可能です。

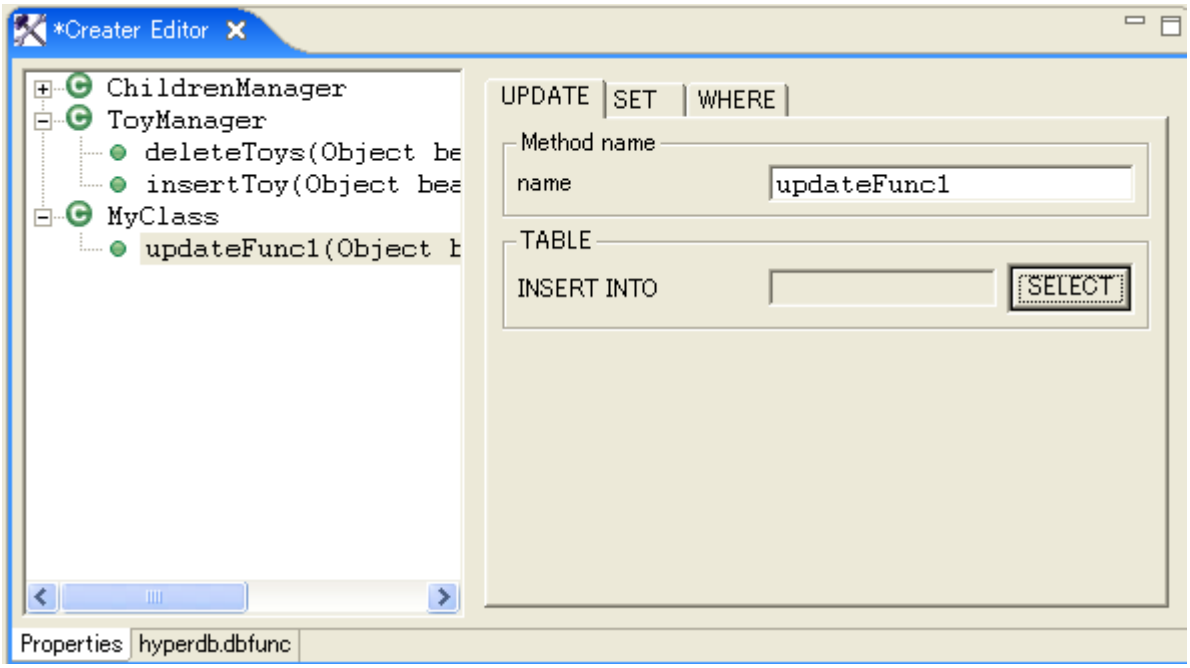


6.10. UPDATE 文を実行するメソッドの作成

UPDATE 文を実行するメソッドを作成したいクラスを選択後、右クリックし、“Create a new Method” -> “Create a NEW method for UPDATE SQL”を選択します。



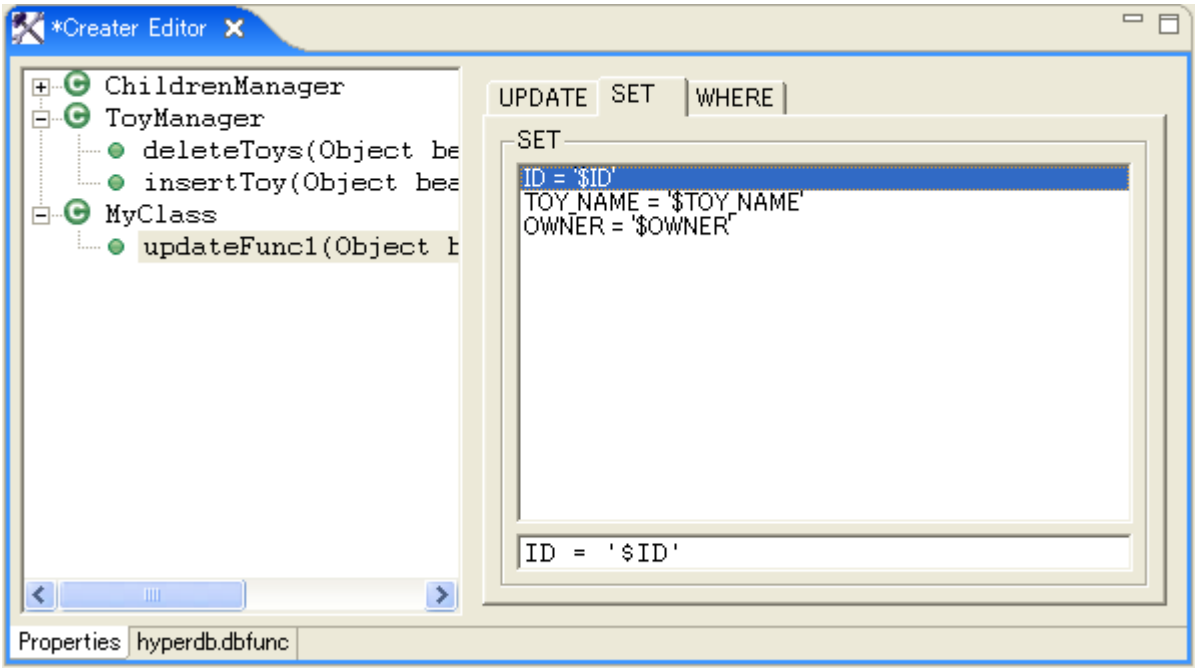
すると、UPDATE 文を実行するメソッドが作成されます。
つぎに UPDATE する対象となるテーブルを指定します。下図の“SELECT”ボタンを押すことでテーブルを設定することが出来ます。



テーブルを選択するダイアログが表示されますので、テーブルを選択します。

テーブル選択後、テーブルに対する SET 句のデフォルトの設定を行うかどうか聞くダイアログが出ますので、 'OK' ボタンを押します。

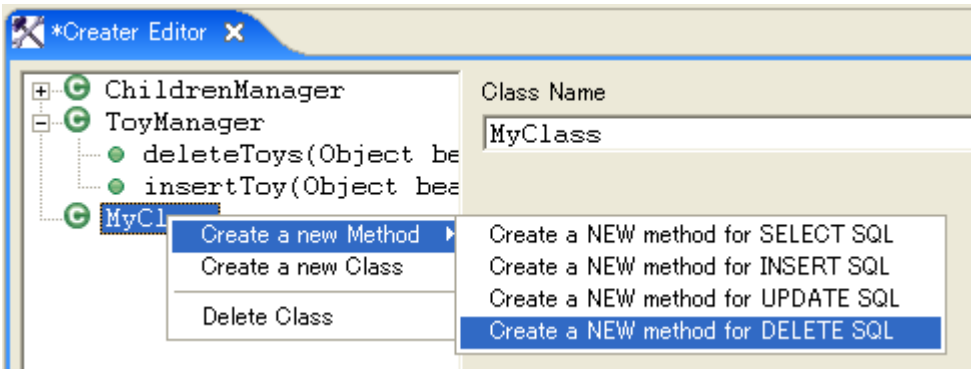
‘OK’ボタンを押すことによって、UPDATE 文の‘SET’句が自動的に設定されます。
サブクエリーや関数の値をカラムに入力する場合は、スクリプトを編集してカスタマイズすることも可能です。



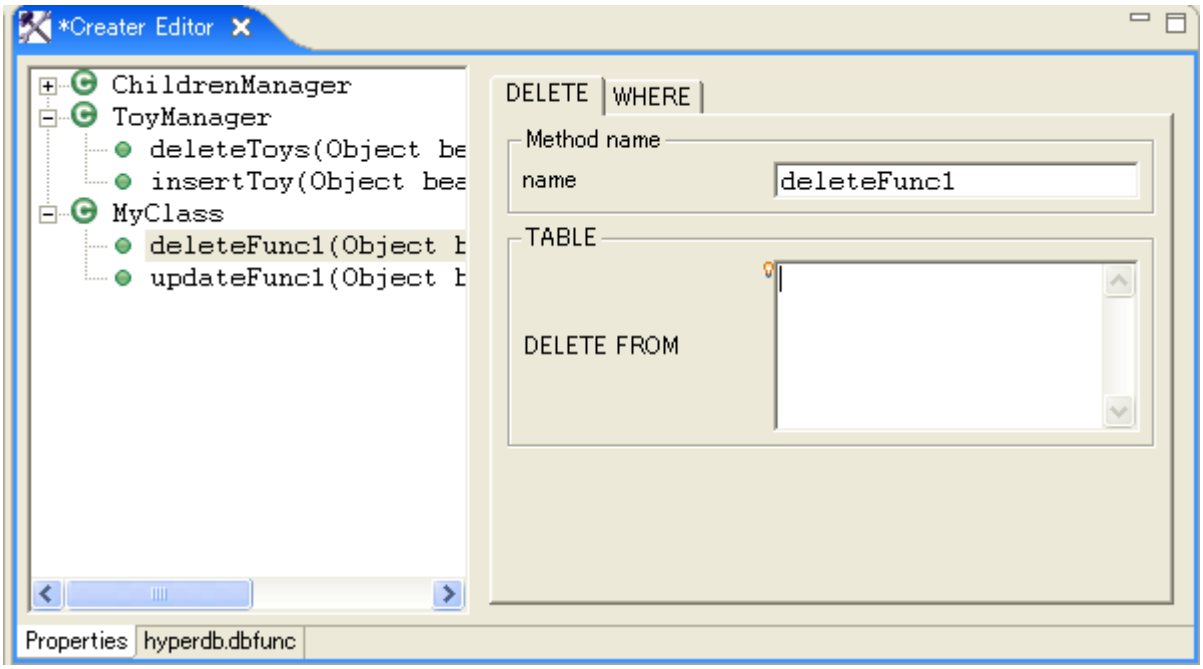
‘WHERE’ 句の設定は “SELECT SQL”のものと同じです。

6.11. DELETE 文を実行するメソッドの作成

UPDATE 文を実行するメソッドを作成したいクラスを選択後、右クリックし、“Create a new Method” -> “Create a NEW method for DELETE SQL”.を選択します。



すると、DELETE 文を実行するメソッドが作成されます。



‘WHERE’ 句の設定は “SELECT SQL”のものと同じです。

7. データベースの互換性について

7.1. 型の互換性について

“Database Type Config File”に、データベースの型と Java の型のマッピングを記述することが出来ます。“Database Type Config File”に関してはプロジェクトのプロパティで指定することが出来ます。フォーマットは下の様になります。

```
<?xml version="1.0" encoding="UTF-8"?>
<HYPERDBCONFIG>
    <TYPE name="[Type name to show in GUI]"
        dbType="[Type of database column]"
        javaType="[Corresponding Java Class]" />
</HYPERDBCONFIG>
```

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<HYPERDBCONFIG>
    <TYPE name="VARCHAR(16)" dbType="VARCHAR(16)" javaType="java.lang.String" />
    <TYPE name="VARCHAR(128)" dbType="VARCHAR(128)" javaType="java.lang.String" />
    <TYPE name="VARCHAR(255)" dbType="VARCHAR(255)" javaType="java.lang.String" />
    <TYPE name="TEXT" dbType="TEXT" javaType="java.lang.String" />
    <TYPE name="TIMESTAMP" dbType="TIMESTAMP" javaType="java.sql.Timestamp" />
    <TYPE name="DATE" dbType="DATE" javaType="java.sql.Date" />
    <TYPE name="Integer" dbType="INT" javaType="java.lang.Integer" />
    <TYPE name="REAL" dbType="REAL" javaType="java.lang.Double" />
    <TYPE name="BOOL" dbType="BOOL" javaType="java.lang.Boolean" />
</HYPERDBCONFIG>
```

7.2. LIMIT と OFFSET

LIMIT と OFFSET に対応していないデータベースではこの機能を使うことが出来ません。
対応していないデータベースを使う場合はSELECT文を実行するメソッドの第３、第４変数に必ずnullを入力するようにしてください。