AndroMDA 入門ガイド

第１．３版

Part 3 of 4

２００５／０８／１２

株式会社エクサ 技術部

＜Part 1 of 4＞

０　はじめに --MDA について-- 5
１　AndroMDA とは 6
２　セットアップ 7
２．１　前提ソフトウェアについて 7
２．１．１　Java(J2SDK) のセットアップ 7
２．１．２　Maven のセットアップ 7
２．１．３　JBoss のセットアップ 9
２．１．４　MagicDraw UML のセットアップ 12
２．１．５　環境変数の設定 15
２．２　AndroMDA の新規プロジェクト作成 16
２．３　AndroMDA サンプルアプリケーション 18
２．３．１　AndroMDA のソースをダウンロード及び解凍 18
２．３．２　Animal Quiz のビルド及びデプロイ 18
２．３．３　JBoss の設定変更 19
２．３．４　JBoss の起動及びテーブルの作成 20
２．３．５　Animal Quiz の実行 21
２．３．６　Animal Quiz のモデル図 25
２．４　Animal Quiz の Java スケルトン生成 29

＜Part 2 of 4＞

３　AndroMDA を用いた開発 31
３．１　Hello World アプリケーション 31
３．１．１　プロジェクトの作成 31
３．１．２　モデリング 33
３．１．３　ビルド&デプロイ 43
３．１．４　アプリケーション実行 45

＜Part 3 of 4＞

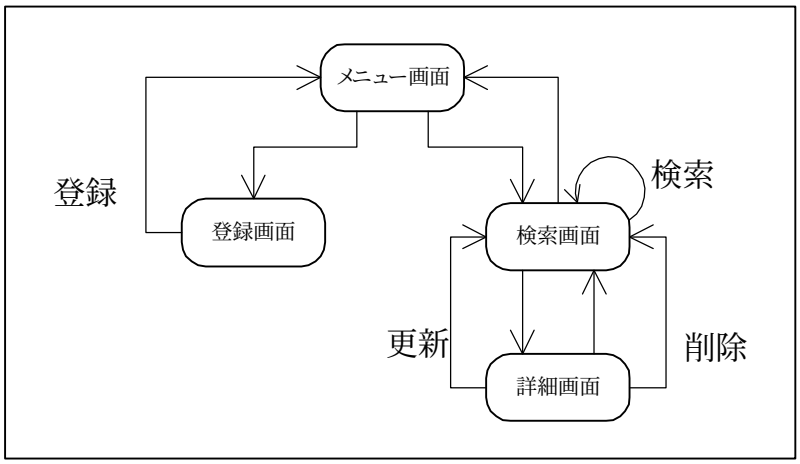
３．２　商品管理アプリケーション　46
３．２．１　プロジェクトの作成　46
３．２．２　モデリング　48
３．２．３　ビルド&デプロイ　68
３．２．４　アプリケーション実行　73

＜Part 4 of 4＞

３．３　事例紹介：バグ追跡掲示板アプリケーション　78
３．３．１　作成手順　82
３．３．２　アーキテクチャ　82
３．３．３　モデル図　84
３．３．４　実装ファイル　92
３．３．５　アプリケーションの実行及び操作方法　93
３．３．６　使用したステレオタイプとタグ付き値　102
４　AndroMDA を用いる場合の制約事項　103
４．３　画面レイアウトの制限　103
４．４　日本語は文字化けする　104
４．５　データベーステーブルの項目設定の制限　104
４．６　OCL の使用制限　104
５　まとめ　105

３．２　商品管理アプリケーション

このサンプルアプリケーションは、商品(Goods)の登録・検索・更新・削除といった管理を行うアプリケーションである。画面遷移は下記の通りである。



作成手順は「３．１HelloWorld アプリケーション」と同様

・プロジェクトの作成
・モデリング
・ビルド＆デプロイ
・アプリケーション実行

となる。

３．２．１　プロジェクトの作成

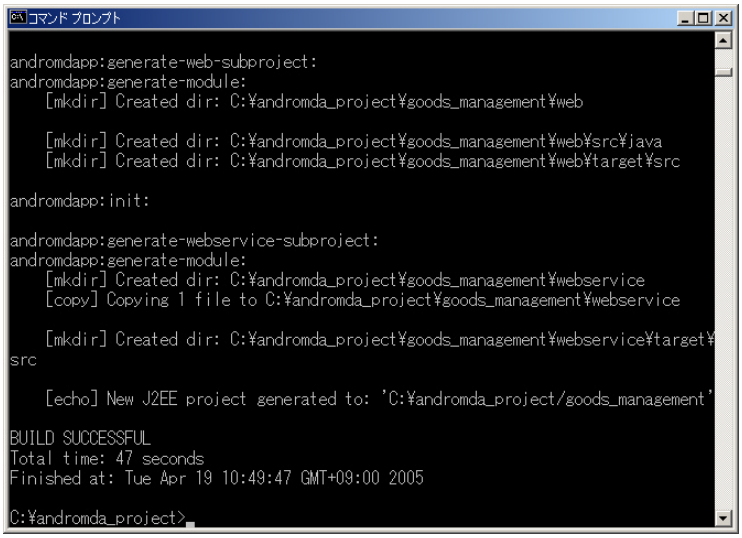
maven を使用して新規プロジェクトを作成する。コマンドプロンプトを開き、「**プロジェクトルートディレクトリ**」へ移動する。その後、下記のコマンドを実行する。

```
>maven andromdapp:generate
```

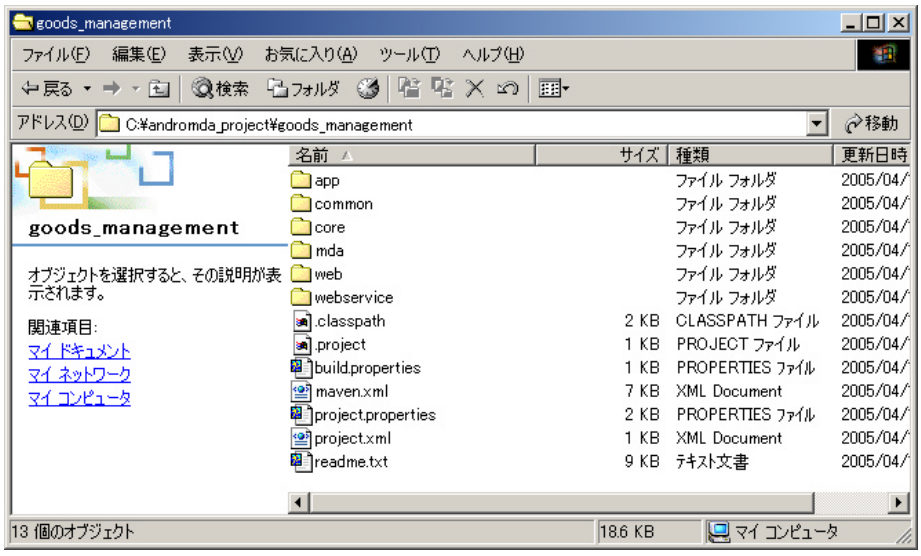
作成途中、いくつか質問されるので答える(太字は本ガイドでの入力例)。

Please enter your first and last name (i.e. Chad Brandon):
>Wouter Zoons ※開発者名
Please enter the name of your J2EE project (i.e. Animal Quiz):
>Goods Management　※プロジェクト名
Please enter the id for your J2EE project (i.e. animalquiz):
>goods_management ※プロジェクト ID 名
Please enter a version for your project (i.e. 1.0-SNAPSHOT):
>1.0-sample ※バージョン
Please enter the base package name for your J2EE project (i.e. org.andromda.samples):
>goodsmanagement.sample ※パッケージ名

Please enter the type of transactional/persistence cartridge to use (enter 'hibernate', 'ejb', or 'spring'):
>**hibernate** ※使用するカートリッジ
Would you like a web application? (enter 'yes' or 'no'):
>**yes** ※Web アプリケーションを適用するか
Would you like to be able to expose your services as web services? (enter 'yes' or 'no'):
>**yes** ※Web サービスを公開するか



「BUILD SUCCESSFUL」と表示されればプロジェクトの作成は成功である。



プロジェクトは、「**プロジェクトルートディレクトリ**\goods_management」というパスで作成される。以後、本ガイドではこのパスを「**GoodsManagement プロジェクト**」と記述する。

３．２．２　モデリング

モデリングは以下の手順で行う。

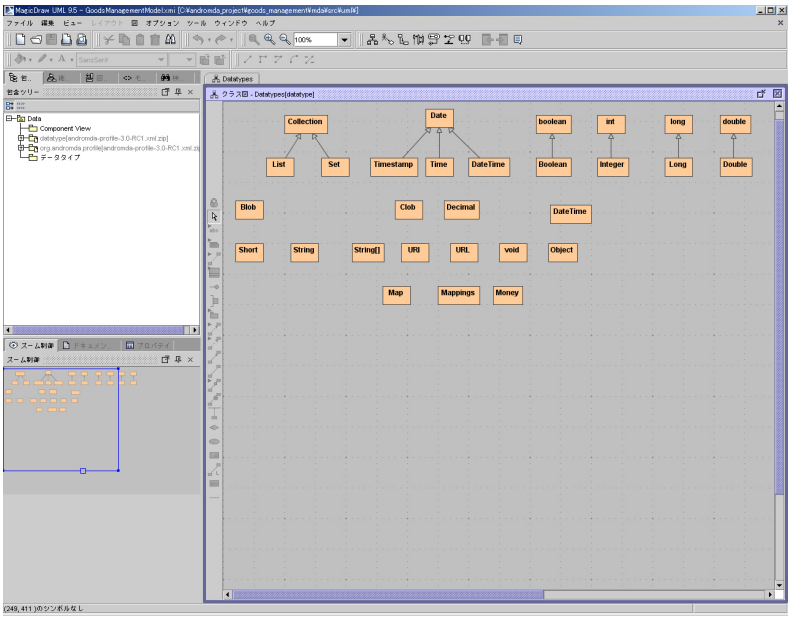
- MagicDraw でモデルファイル(XMI)を読込
- ドメイン側のクラス図
- Web 側のクラス図作成
- 画面遷移図であるアクティビティ図の作成

MagicDraw UML を起動し、初期状態の XMI ファイルである、

「**GoodsManagement プロジェクト**\mda\src\uml\GoodsManagementModel.xmi」を開く。

読み込みの途中で警告が表示される場合がある。そのときには、「２．３．６」と同様「andromda-profile-3.0－RC1.xml.zip」を指定する。

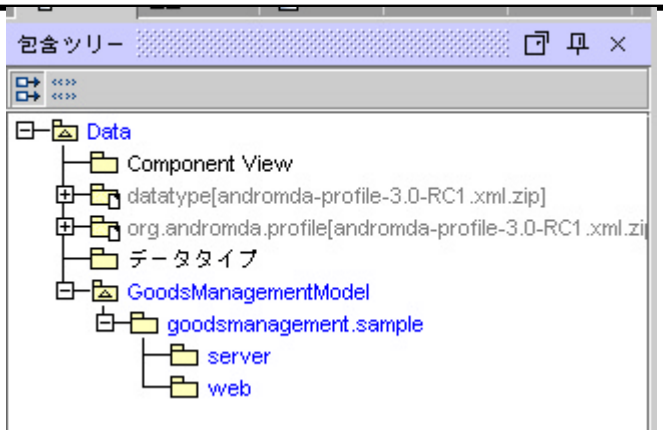
これで、XMI ファイルが読み込まれる。



次にモデリングを行う。

まず、包含ツリーの「Data」で右クリックメニューの「新規エレメント」→「モデル」を選択し、任意の名前をつける(例：GoodsManagementModel)。さらに、作成した「モデル」で右クリックメニューの「新規エレメント」→「モデルパッケージ」を選択し、ここでは、３．２．１で入力したパッケージ名をつける(例：goodsmanagement.sample)。

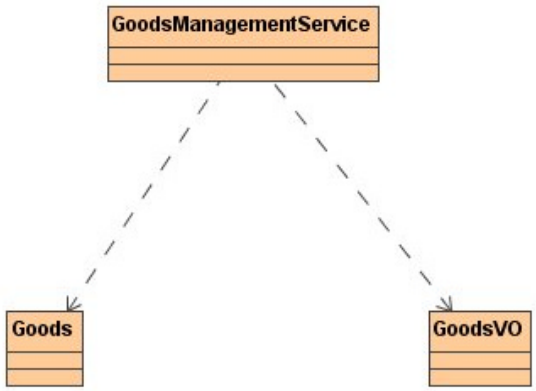
本アプリケーションは３．１と違い DB アクセスも行うので、画面側(Web)・ドメイン側(Server)とパッケージを分ける。「goodsmanagement.sample」の下に「server」と「web」という「モデルパッケージ」を追加する。



○　ドメイン側クラス図作成

「server」で右クリックメニューの「新規の図」→「クラス図」を選択し任意の名前をつける(例：GoodsManagement-Class)。

作成したクラス図に下記イメージのように、「Goods」、「GoodsVO」、「GoodsManagementService」クラスを新規追加し、「GoodsManagementService」から「Goods」、「GoodsVO」へ依存関係の線を引く。

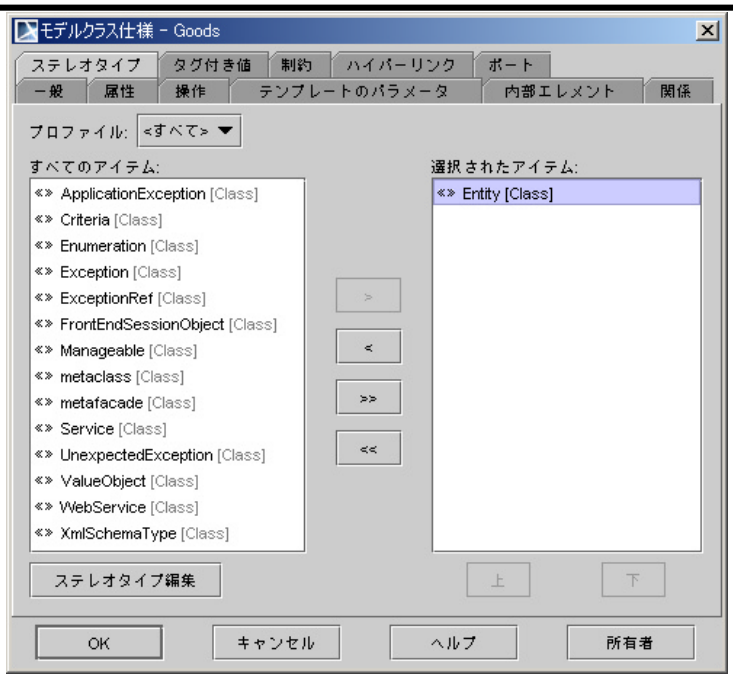


この３クラスに対して設定する。

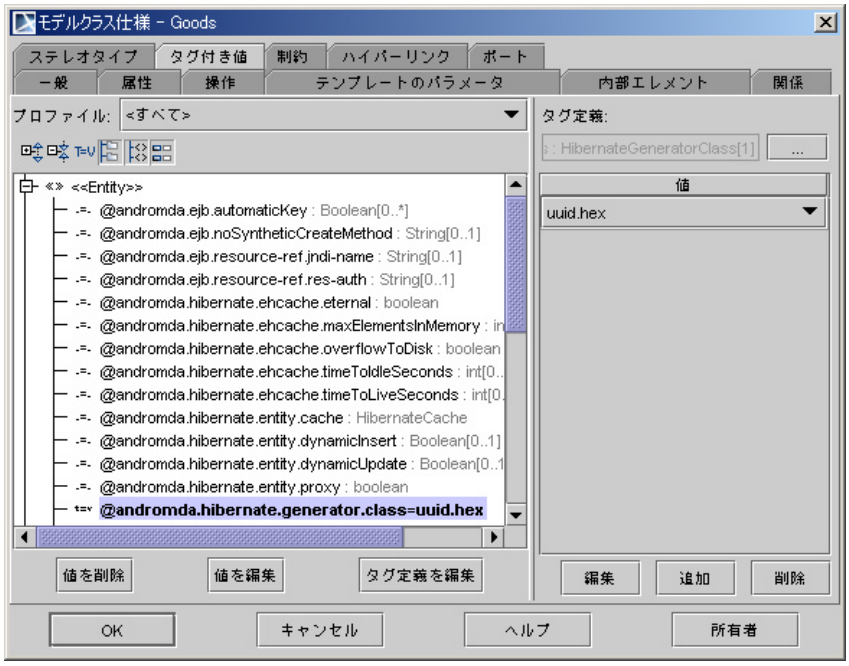
・ Goods クラスの設定

「Goods」クラスには、ステレオタイプ・属性３つ・操作２つを設定する。

「Goods」クラスの仕様ダイアログを開き、「ステレオタイプ」タブで「Entity」を追加する。



次に、「タグ付き値」タブでタグ付き値
「@andromda.hibernate.generator.class=**uuid.hex**」を設定する。



次に、「属性」タブで以下の３つを追加する。

属性名	可視性	タイプ
id	public	String
name	public	String
price	public	int

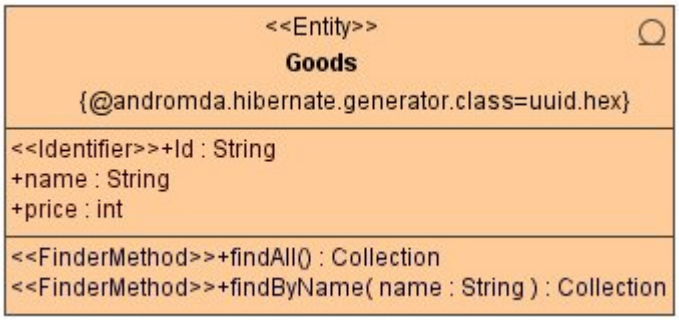
属性「id」を PrimaryKey にするため、属性「id」の仕様ダイアログの「ステレオタイプ」タブで「Identifier」を設定する。

次に「操作」タブで以下の２つの操作を追加する。

操作名	戻りの型	パラメータ	
		名前	タイプ
findAll	Collection	なし	なし
findByName	Collection	name	String

どちらの操作にも仕様ダイアログの「ステレオタイプ」タブで「FinderMethod」を設定する。

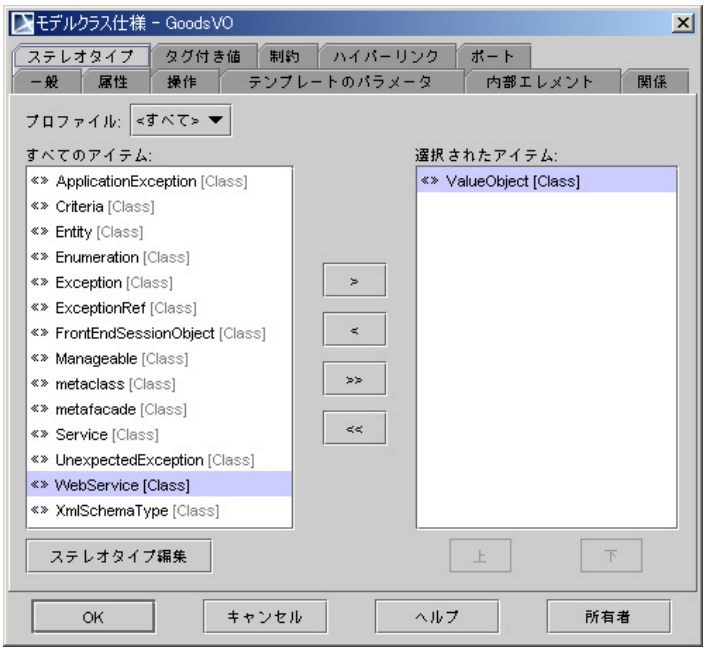
これで、Goods クラスの設定が完了する。



- GoodsVO クラスの設定

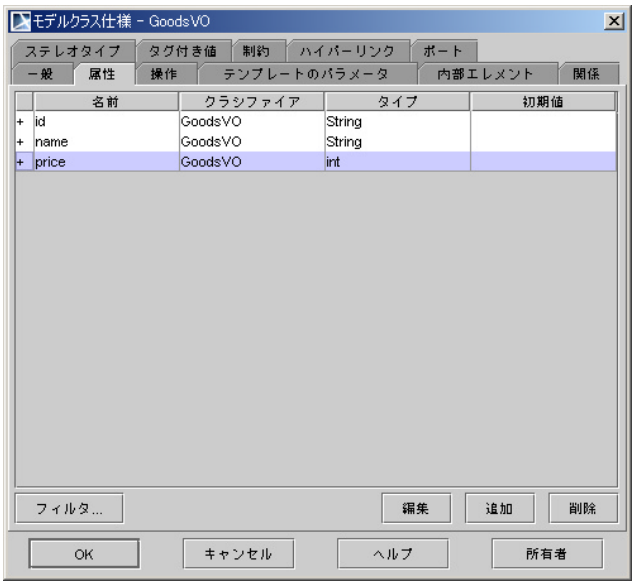
「GoodsVO」クラスには、ステレオタイプ・属性３つを設定する。

「GoodsVO」クラスの仕様ダイアログを開き、「ステレオタイプ」タブで「ValueObject」を追加する。

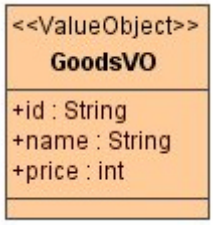


次に、「属性」タブで以下の３つを追加する。

属性名	可視性	型
id	public	String
name	public	String
price	public	int

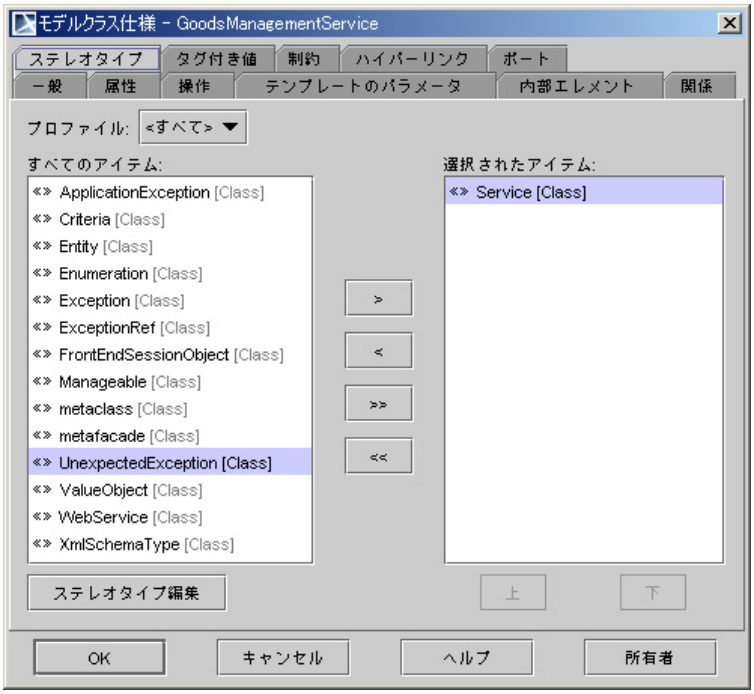


これで、GoodsVO の設定は完了する。



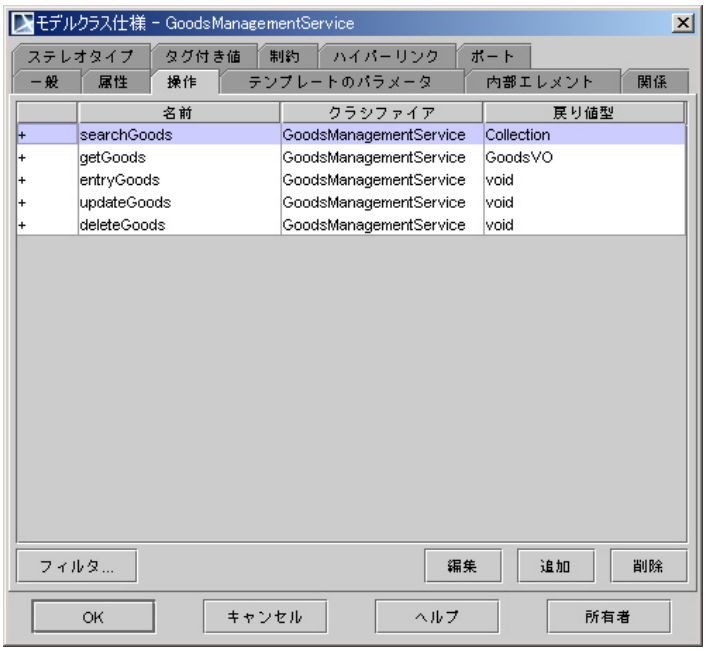
・ GoodsManagementService クラスの設定

「GoodsManagementService」クラスには、ステレオタイプ・操作５つを設定する。「GoodsManagementService」クラスの仕様ダイアログを開き、「ステレオタイプ」タブで「Service」を追加する。

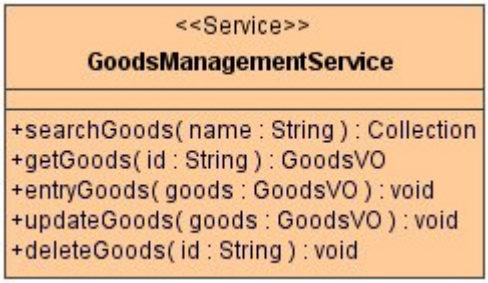


次に、「操作」タブで以下の５つを追加する。

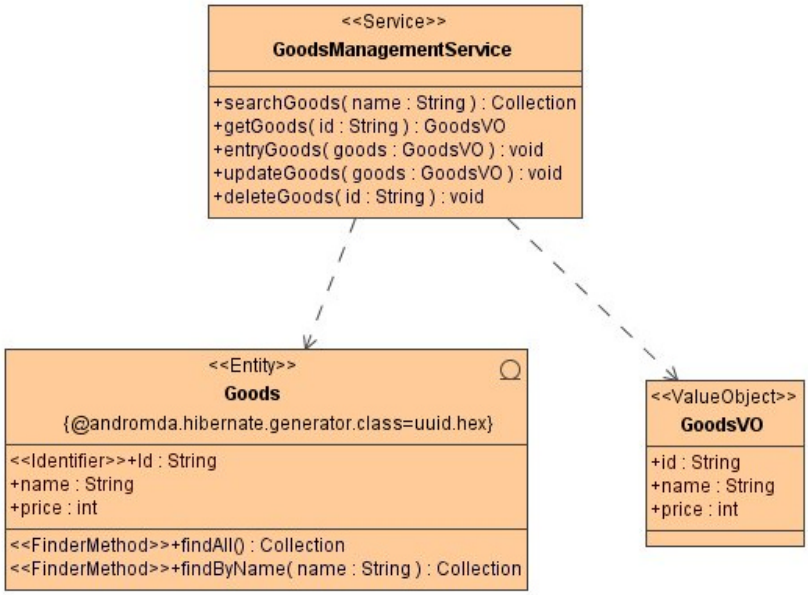
操作名	戻りの型	引数	
		名前	型
searchGoods	Collection	name	String
getGoods	GoodsVO	id	String
entryGoods	void	goods	GoodsVO
updateGoods	void	goods	GoodsVO
deleteGoods	void	id	String



これで GoodsManagementService クラスの設定は完了である。



以上でドメイン側の設定が完了である。



下記にこのクラス図で使用したステレオタイプとタグ付き値を説明する。

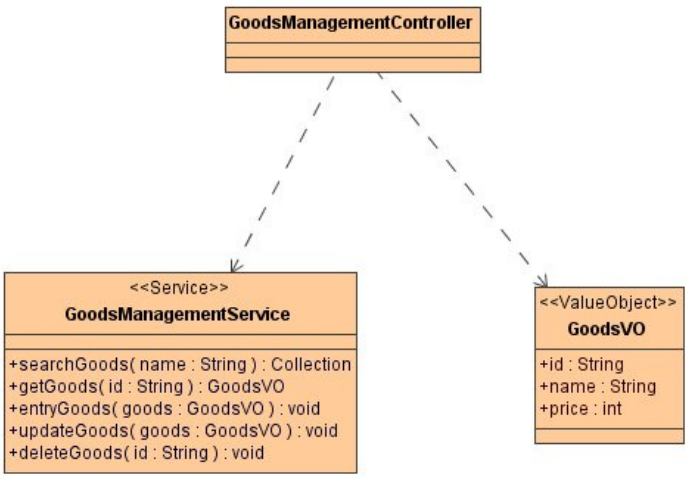
ステレオタイプ	内容
Entiry	テーブルをマッピングしたクラスを表す
ValueObject	値オブジェクトを表す
Service	ファサードの役割を行うクラスを表す
Identifier	Entity の id(PK)を表す
FinderMethod	検索メソッドを表す hibernate カートリッジでは、検索クエリが生成される

タグ付き値	内容
@andromda.hibernate.generator.class	Entity の id 生成方法について設定する。

○　Web 側のクラス図作成

ユーザーのアクションに応じてドメイン側の Service クラス(GoodsManagementService)の操作を呼び出すコントローラークラスを作成する。「web」で右クリックメニューの「新規の図」→「クラス図」を選択し任意の名前をつける(例：GoodsManagement-WebClass)。

作成したクラス図に下記イメージのように、「GoodsManagementController」クラスを新規追加し、作成済みの「GoodsManagementService」と「GoodsVO」を左側のツリービューからドラッグして追加する。「GoodsManagementController」から「GoodsManagementService」と「GoodsVO」へ依存の線を引く。

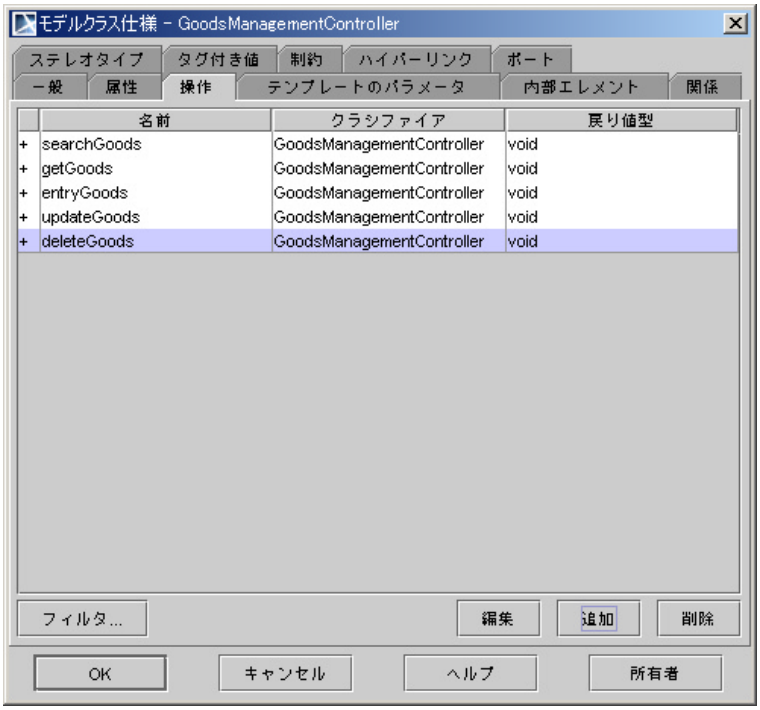


続いて「GoodsManagementController」クラスの設定を行う。

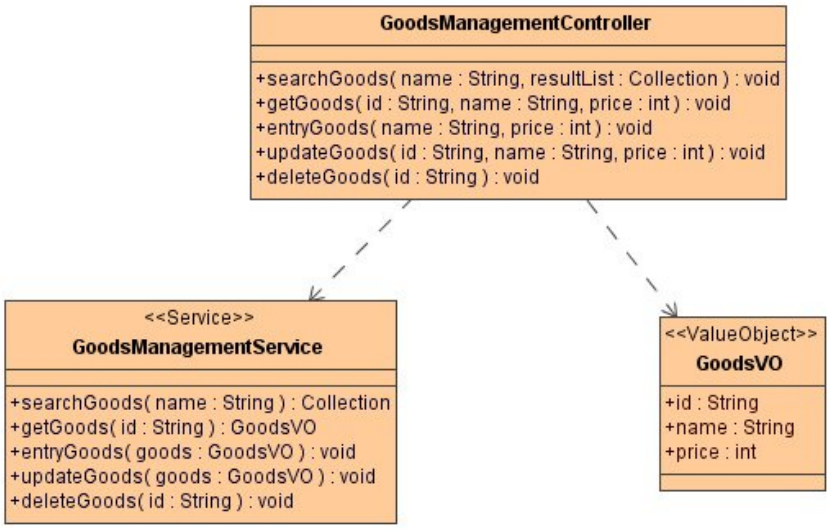
・ GoodsManagementController クラスの設定

「GoodsManagementController」の仕様ダイアログを開き「操作」タブで下記の操作を追加する。

操作名	戻りの型	引数	
		名前	型
searchGoods	void	name	String
		resultList	Collection
getGoods	void	id	String
		name	String
		price	int
entryGoods	void	name	String
		price	int
updateGoods	void	id	String
		name	String
		price	int
deleteGoods	void	id	String



これで Web 側のクラス図の設定が完了する。



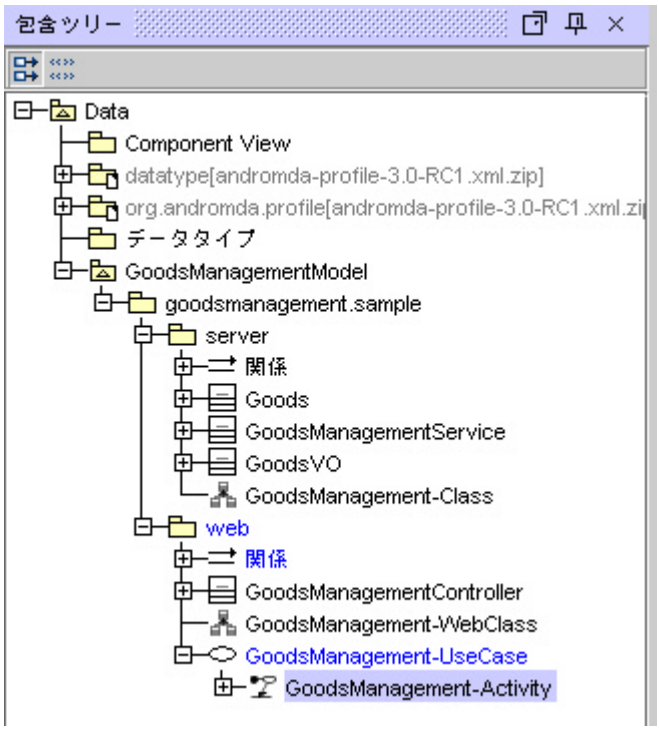
○ ユースケース及び画面遷移のアクティビティ図作成

「web」で右クリックメニューの「新規エレメント」→「ユースケース」を選択し任意の名前をつける。

例）GoodsManagement-UseCase

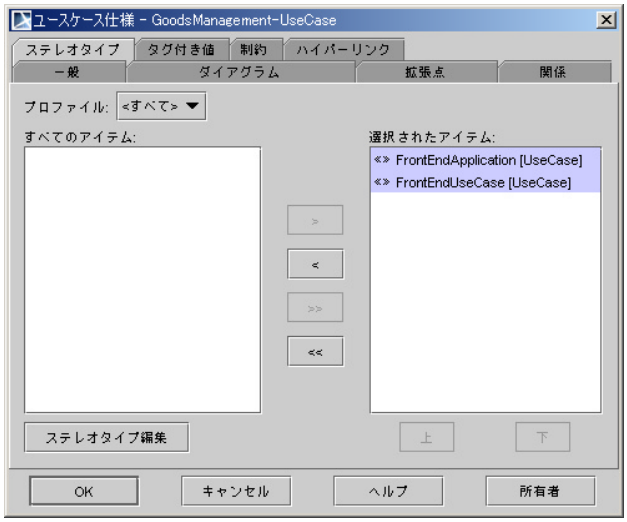
ユースケース「GoodsManagement-UseCase」で右クリックメニューの「新規の図」→「アクティビティ図」を選択し任意の名前をつける。

例）GoodsManagement-Activity



作成したユースケース「GoodsManagement-UseCase」にステレオタイプを設定する。ユースケース「GoodsManagement-UseCase」で右クリックメニューの「仕様」を選択する。表

示されたダイアログの「ステレオタイプ」タブで「FrontEndApplication」と「FrontEndUseCase」を選択されたアイテムへ追加し、「OK」ボタンを押下する。

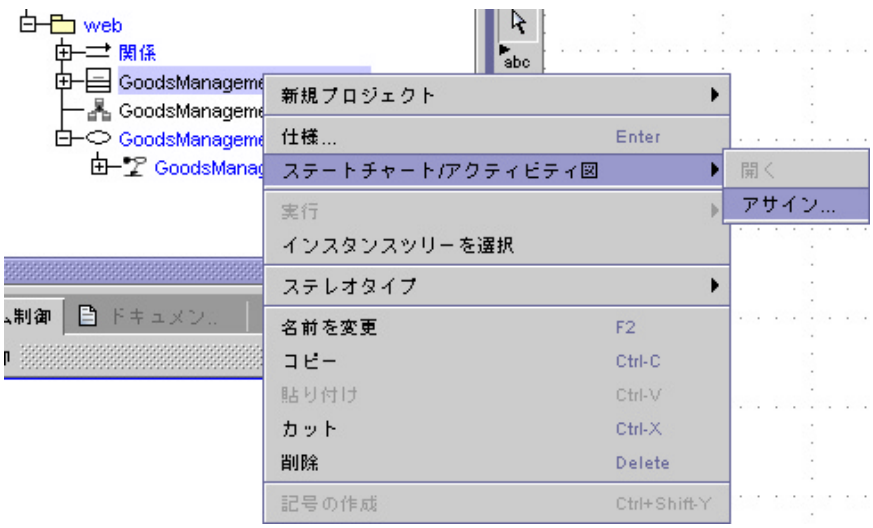


設定したステレオタイプの詳細は以下の通り

ステレオタイプ	内容
FrontEndApplication	アプリケーションの開始点を意味する
FrontEndUseCase	設定したユースケースをアプリケーションに含めることを意味する

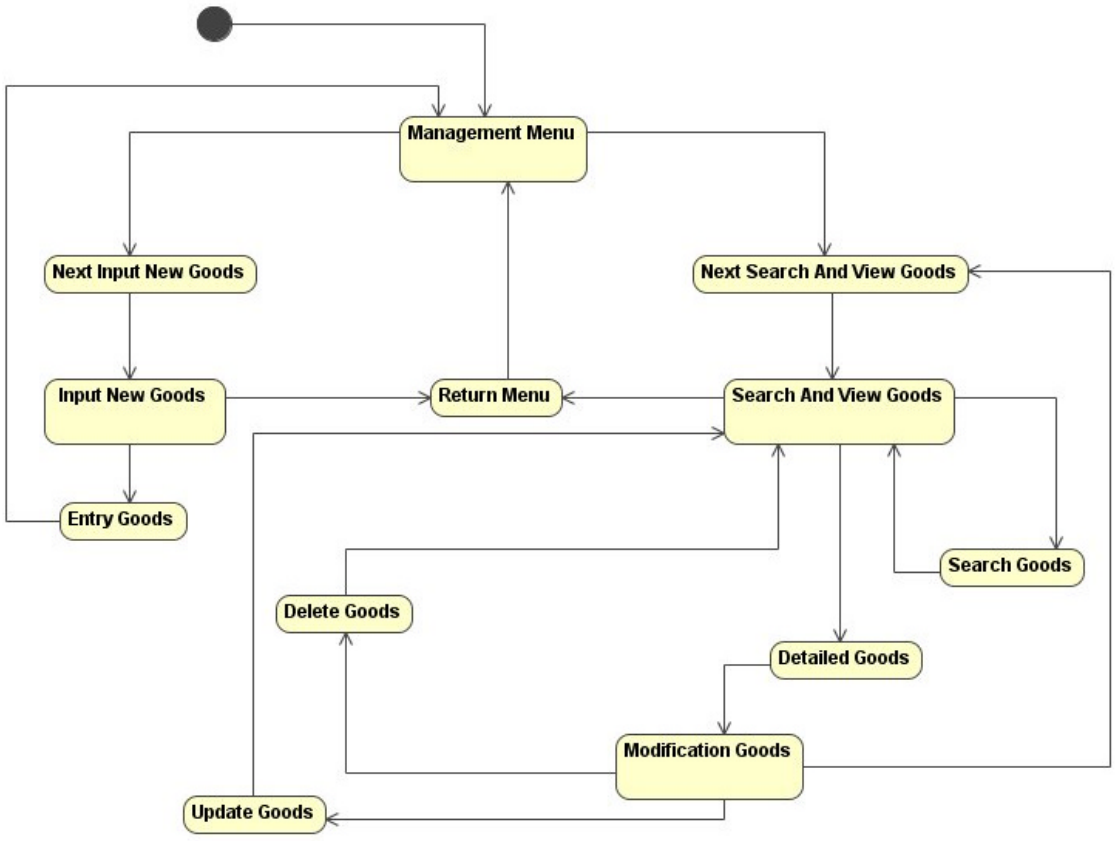
前述で作成した Web 側のクラス「GoodsManagementController」の操作をアクティビティ図のアクションでイベント呼出し操作として設定できるようにアサインする。

包含ツリーの「GoodsManagementController」で右クリックメニューの「ステートチャート/アクティビティ図」→「アサイン」を選択し、表示されたダイアログで「GoodsManagement-Activity」を選択して「アサインする」ボタンを押下し、「閉じる」ボタンでダイアログを閉じる。



アクティビティ図へアサインすることによって、アクティビティ図のイベントでアサインしたクラスの操作を呼出し操作として設定することができる。

アクティビティ図に下記のイメージのようにモデルを配置する。



各モデルを設定する。

- JSP の設定

  下記のアクション状態に対してステレオタイプを設定する。

  a)「Management Menu」

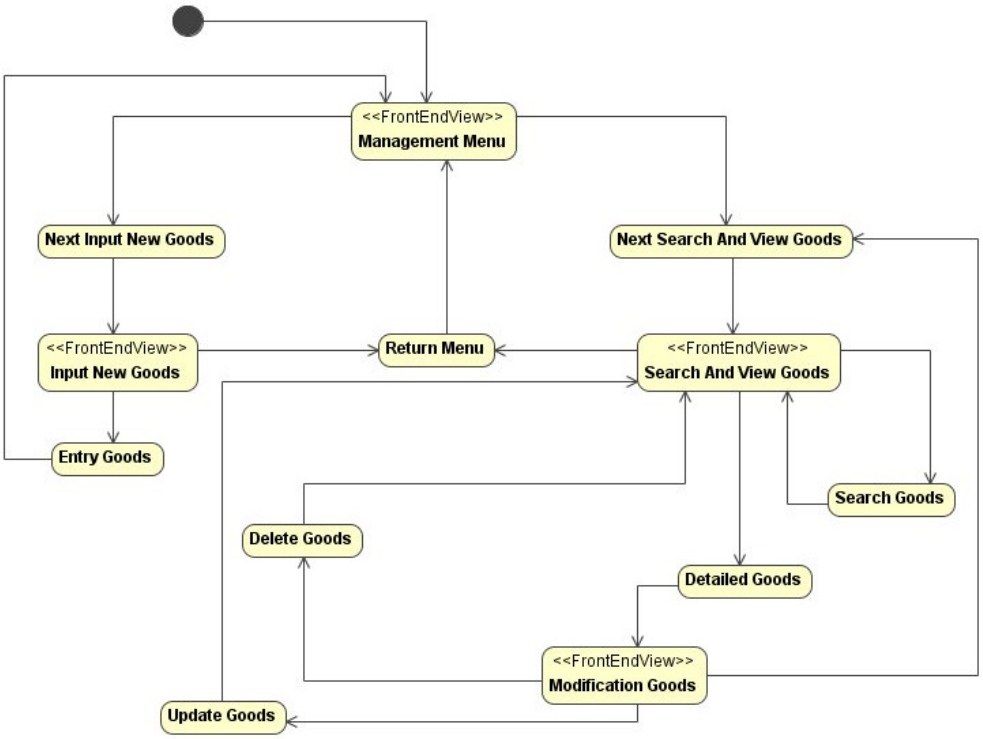
  b)「Input New Goods」

  c)「Search And View Goods」

  d)「Modification Goods」

  各仕様ダイアログを開き、「ステレオタイプ」タブで「FrontEndView」を設定する。

すべて設定すると以下のイメージのようになる。

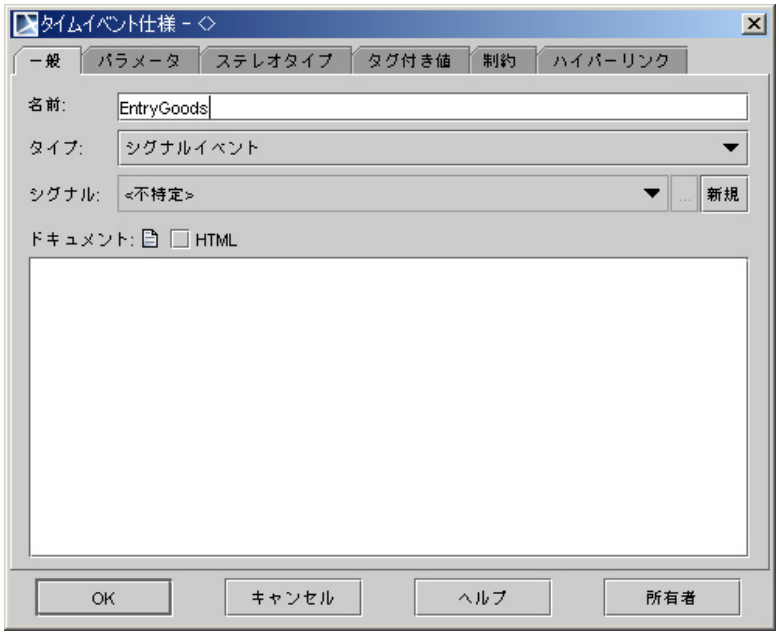


・ イベントの設定

下記の遷移の線に対してイベントを設定する。

遷移元アクション	遷移先アクション	イベント名
Management Menu	Next Input New Goods	EntryGoods
Input New Goods	Entry Goods	Entry
Input New Goods	Return Menu	ReturnMenu
Management Menu	Next Search And View Goods	SearchGoods
Search And View Goods	Search Goods	Search
Search And View Goods	Detailed Goods	Detail
Search And View Goods	Return Menu	ReturnMenu
Search Goods	Search And View Goods	Result
Modification Goods	Update Goods	Update
Modification Goods	Delete Goods	Delete
Modification Goods	Next Search And View Goods	Return

イベントの設定方法は、対象遷移の線の仕様ダイアログを開き、「一般」タブの「トリガー」の「編集」ボタンを押下する。表示されたダイアログの「名前」に上記の「イベント名」、「タイプ」に「シグナルイベント」を設定する。

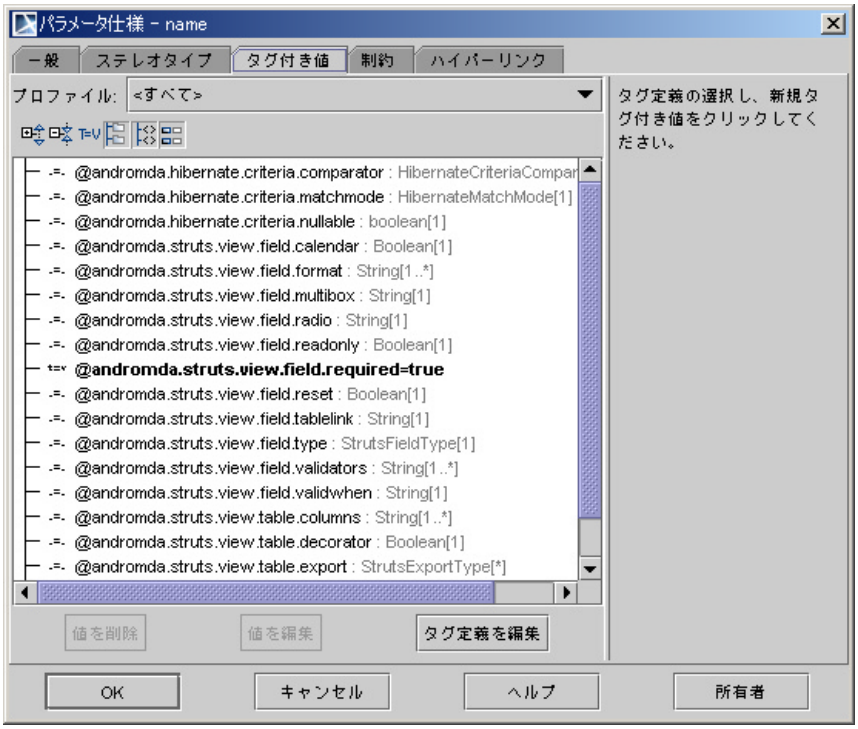


同様に他のイベントを設定する。今設定したイベントの中にはパラメータを持つイベントがある。下記のイベントについてパラメータ及びパラメータに付加するタグ付き値の設定をする。

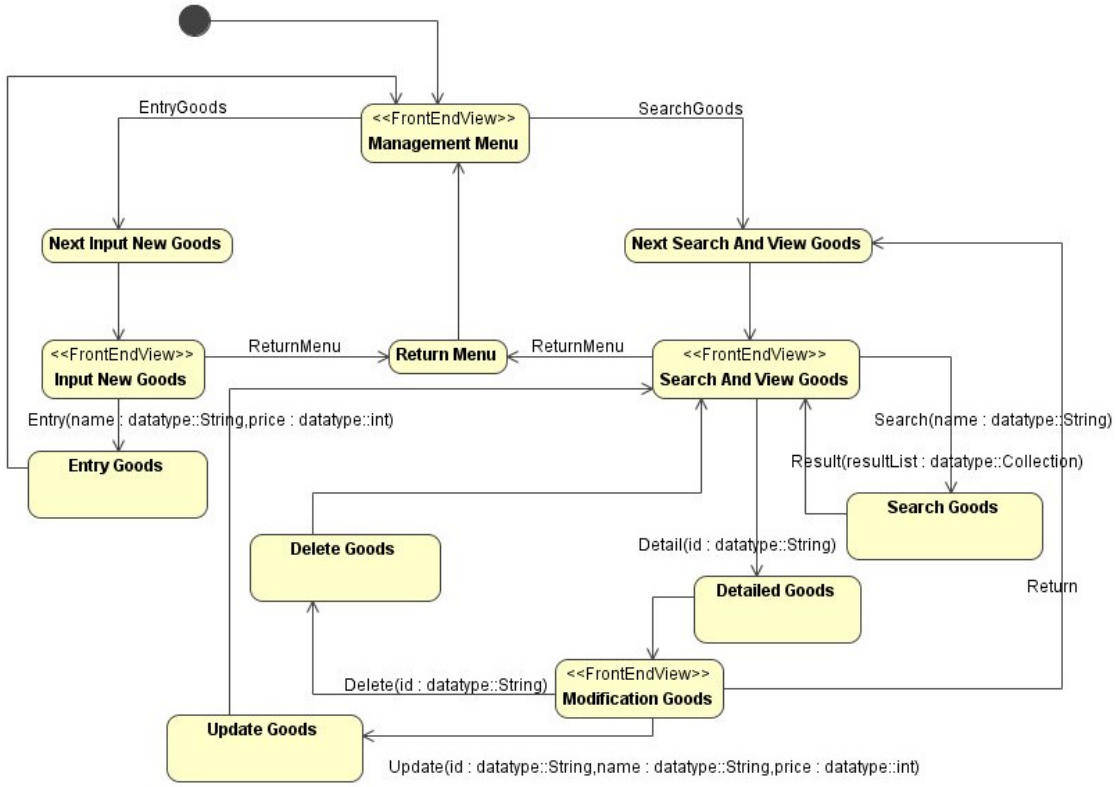
イベント名	パラメータ	型	タグ付き値
Entry	name	String	@andromda.struts.view.field.required =**true**
	price	int	@andromda.struts.view.field.required =**true**
Search	name	String	なし
Result	resultList	Collection	@andromda.struts.view.table.column =**id,name,price**
Detail	id	String	@andromda.struts.view.field.tablelink =**resultList.id**
			@andromda.struts.view.field.type=**link**
Update	id	String	@andromda.struts.view.field.readonly =**true**
	name	String	@andromda.struts.view.field.required =**true**
	price	int	@andromda.struts.view.field.required =**true**
Delete	id	String	@andromda.struts.view.field.type=**hidden**

パラメータの設定は、「トリガー」の仕様ダイアログの「パラメータ」タブに追加する。

タグ付き値の設定は、パラメータの仕様ダイアログの「タグ付き値」タブで設定する。



すべてのイベント設定が完了すると下記のイメージのようになる。

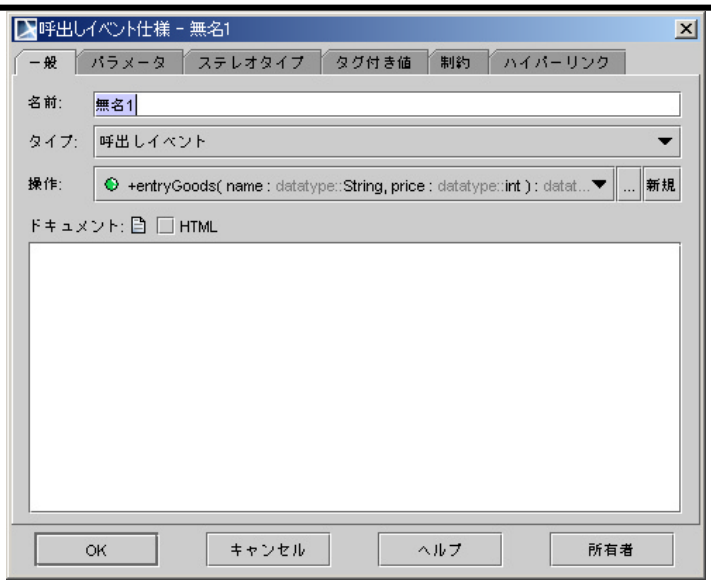


- 操作(GoodsManagementController の操作)の設定

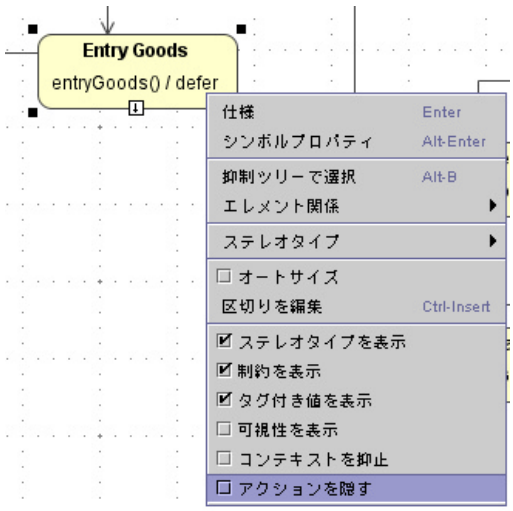
  下記のアクション状態に対して GoodsManagementController の操作を設定する。

アクション	操作
Entry Goods	GoodsManagementController.**entryGoods**()
Search Goods	GoodsManagementController.**searchGoods**()
Detailed Goods	GoodsManagementController.**getGoods**()
Update Goods	GoodsManagementController.**updateGoods**()
Delete Goods	GoodsManagementController.**deleteGoods**()

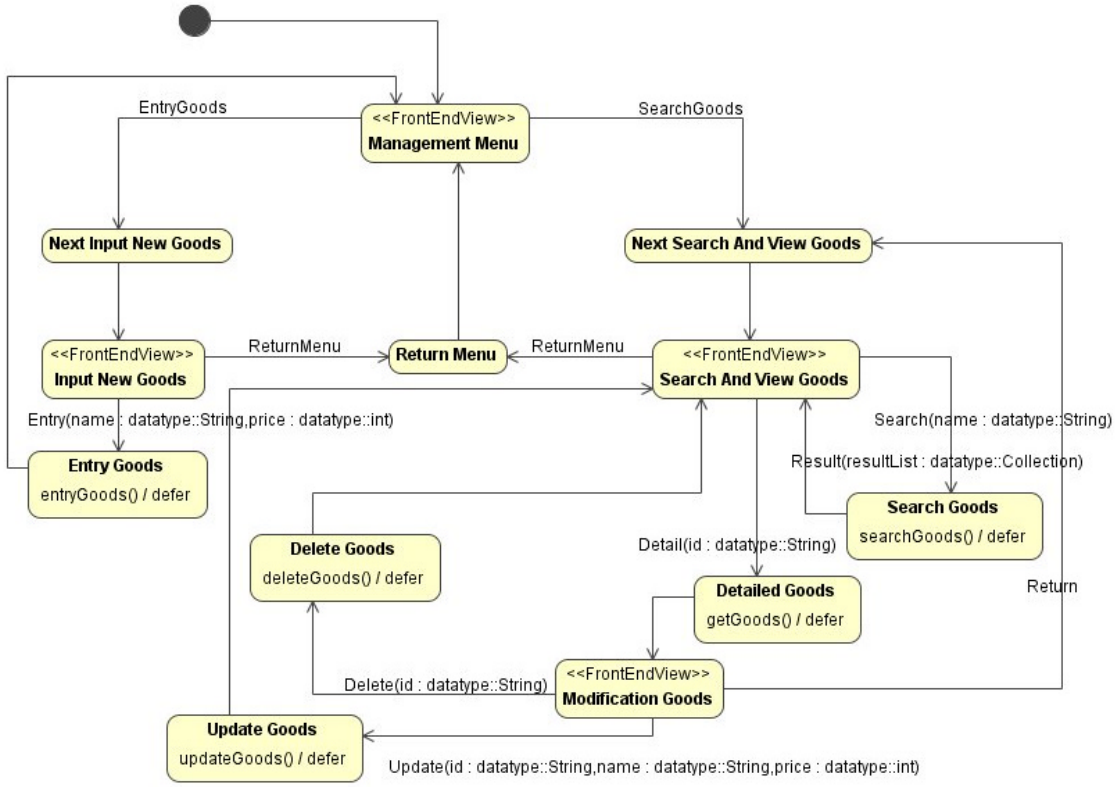
  操作の設定方法は仕様ダイアログを開き、「仕様」タブで「追加」ボタンを押下する。表示されたダイアログの「タイプ」を「呼出しイベント」、「操作」を上記の対応する操作に設定する。



設定は完了したが、そのままではモデル図に表示されないので、対象のアクションの右クリックメニューで「アクションを隠す」のチェックをはずす。これで、モデル図に表示されるようになる。



すべて設定すると、下記のイメージのようになる。



下記にアクティビティ図作成で使用したステレオタイプとタグ付き値の説明を記述する。

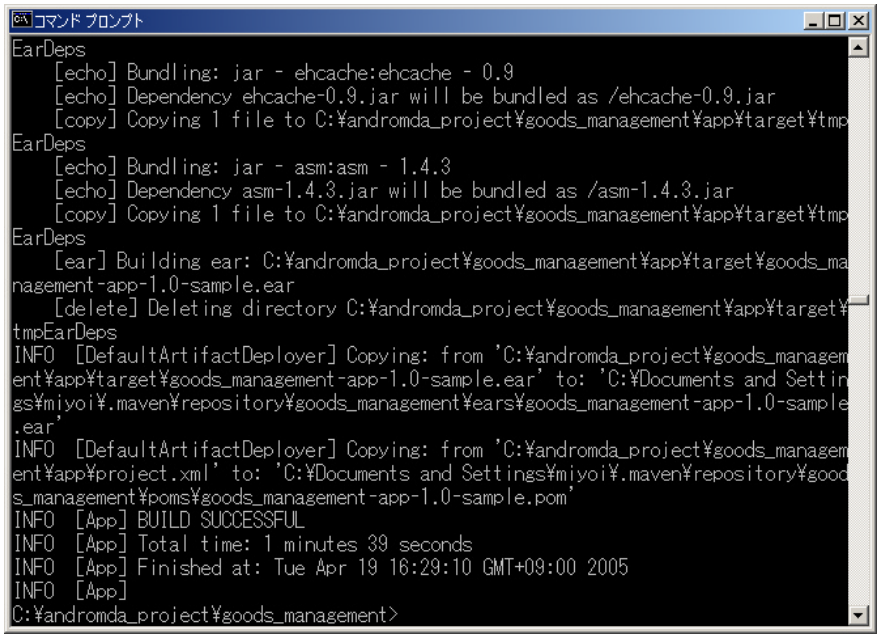
ステレオタイプ	内容
FrontEndView	JSP を表す

タグ付き値	内容
@andromda.struts.view.field.required	必須項目かどうかを設定する
@andromda.struts.view.table.column	テーブルに表示する項目を設定する。カンマで区切る。
@andromda.struts.view.field.tablelink	リンクを作成するテーブルのカラムを設定する。設定方法は、 テーブルパラメータ.カラム (例：resultList.id)
@andromda.struts.view.field.type	項目の種別を設定する
@andromda.struts.view.field.readonly	読み取り専用項目かどうかを設定する

３．２．３　ビルド&デプロイ

この章では maven を使用してプロジェクトの追加実装・ビルドを行い。JBoss へデプロイを行う。コマンドプロンプトを開き、「**GoodsManagement プロジェクト**」まで移動する。下記のコマンドを実行し、ビルドを行う。

```
>maven
```



「BUILD SUCCESSFUL」と表示されればビルドは成功である。

ビルドは完了したが、DB へのアクセスなどの実装追加が必要である。
追加実装が必要なファイルは以下の２つのファイルである。

○ GoodsManagementServiceBeanImpl.java
　パスは「**GoodsManagement プロジェクト**\core\src\java\
　goodsmanagement\sample\server\GoodsManagementServiceBeanImpl.java」

a) インポートの実装

下記の import 文を実装する。

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import javax.ejb.EJBException;
import net.sf.hibernate.HibernateException;
```

b) handleSearchGoods メソッドの実装

下記のロジックをメソッドに上書きする。

```
try {
  Collection goodsList = null;
  if (name == null || name.length() == 0) {
    goodsList = GoodsFactory.findAll(session);
  } else {
    goodsList = GoodsFactory.findByName(session, name);
  }
  Collection resultList = new ArrayList();
  if (goodsList != null && !goodsList.isEmpty()) {
    GoodsVO vo;
    Goods goods;
    Iterator itr = goodsList.iterator();
    while (itr.hasNext()) {
      goods = (Goods)itr.next();
      vo = new GoodsVO(goods.getId(), goods.getName(), goods.getPrice());
      resultList.add(vo);
    }
  }
  return resultList;
}
catch (HibernateException e) {
  throw new EJBException(e);
}
```

c) handleGetGoods メソッドの実装

下記のロジックをメソッドに上書きする。

```
try {
  Goods goods = GoodsFactory.findByPrimaryKey(session, id);
  return new GoodsVO(goods.getId(), goods.getName(), goods.getPrice());
} catch (HibernateException e){
  throw new EJBException(e);
}
```

d) handleEntryGoods メソッドの実装

下記のロジックをメソッドに上書きする。

```
Goods newGoods = GoodsFactory.create(goods.getName(), goods.getPrice());
try {
  session.save(newGoods);
}
catch (HibernateException e) {
  throw new EJBException(e);
}
```

e) handleUpdateGoods メソッドの実装

下記のロジックをメソッドに上書きする。

```
try {
  Goods updateGoods = GoodsFactory.findByPrimaryKey(session, goods.getId());
  updateGoods.setName(goods.getName());
  updateGoods.setPrice(goods.getPrice());
  session.save(updateGoods);
}
catch (HibernateException e) {
  throw new EJBException(e);
}
```

f) handleDeleteGoods メソッドの実装

下記のロジックをメソッドに上書きする。

```
try {
  Goods goods = GoodsFactory.findByPrimaryKey(session, id);
  session.delete(goods);
} catch (HibernateException e){
  throw new EJBException(e);
}
```

これで GoodsManagementServiceBeanImpl.java の追加実装は完了する。

○ GoodsManagementControllerImpl.java

パスは「**GoodsManagement プロジェクト**\web\src\java\goodsmanagement\sample\web\ GoodsManagementControllerImpl.java」

a) インポート文の実装

下記の import 文を実装する。

```
import goodsmanagement.sample.server.GoodsVO;
```

b) searchGoods メソッドの実装

メソッド searchGoods() の実装を、デフォルトの「form.setResultList(resultListDummyList);」ではなく下記のコードで上書きする。

```
// form.setResultList(resultListDummyList);
form.setResultList(this.getGoodsManagementService().searchGoods(form.getName()));
```

c) getGoods メソッドの実装

下記のようにメソッドのロジックを修正する。

```
// form.setId("id-test");
// form.setName("name-test");
// form.setPrice((int)106934601);
GoodsVO vo = this.getGoodsManagementService().getGoods(form.getId());
form.setId(vo.getId());
form.setName(vo.getName());
form.setPrice(vo.getPrice());
```

d) entryGoods メソッドの実装

下記のようにメソッドのロジックを修正する。

```
// form.setName("name-test");
// form.setPrice((int)106934601);
GoodsVO vo = new GoodsVO();
vo.setName(form.getName());
vo.setPrice(form.getPrice());
this.getGoodsManagementService().entryGoods(vo);
```

e) updateGoods メソッドの実装

下記のようにメソッドのロジックを修正する。

```
// form.setId("id-test");
// form.setName("name-test");
// form.setPrice((int)106934601);
GoodsVO vo = new GoodsVO();
vo.setId(form.getId());
vo.setName(form.getName());
vo.setPrice(form.getPrice());
this.getGoodsManagementService().updateGoods(vo);
```

f) deleteGoods メソッドの実装

下記のようにメソッドのロジックを修正する。

```
// form.setId("id-test");
this.getGoodsManagementService().deleteGoods(form.getId());
```

これで GoodsManagementControllerImpl.java の追加実装は完了である。

再度、ビルドを行う。

```
>maven
```

ビルド終了後、JBoss へデプロイを行う。下記のコマンドを実行する。

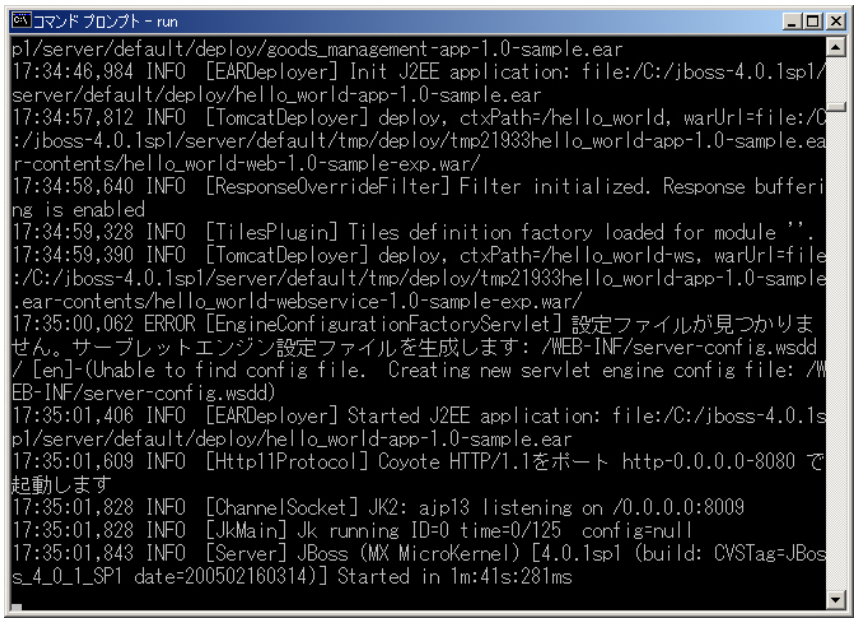
```
>maven deploy
```

このコマンドにより、ファイル「(GoodsManagement プロジェクト)\app\target\goods_management-app-1.0-sample.ear」が「%JBOSS_HOME%\server\default\deploy」にコピーされる。これでデプロイが完了する。

３．２．４　アプリケーション実行

コマンドプロンプトを開き、「%JBOSS_HOME%\bin」へ移動する。下記のコマンドを実行して JBoss を起動する。

>run

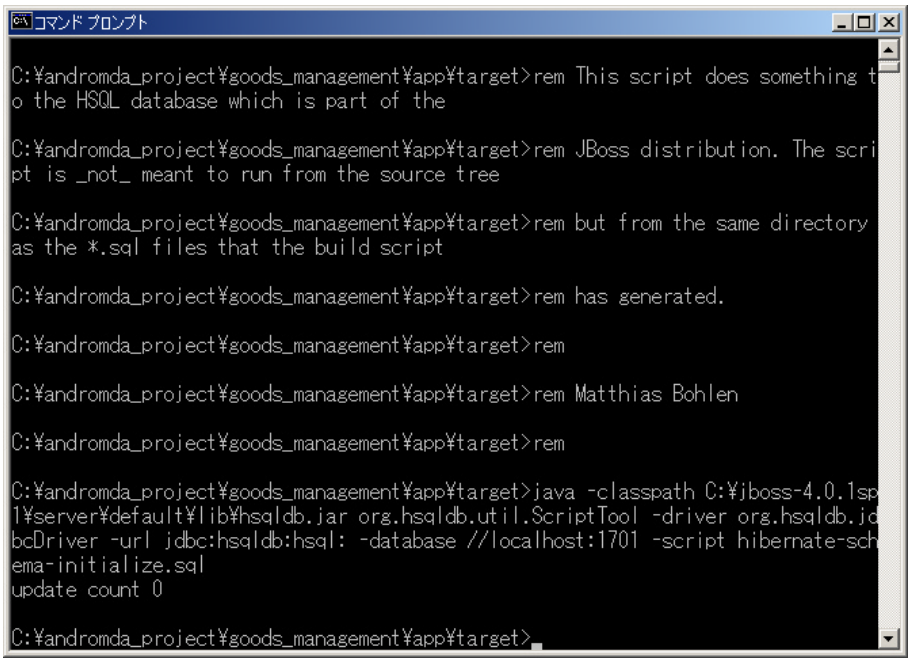


起動後、データベースへテーブルを作成する。

コマンドプロンプトを開き、「GoodsManagement プロジェクト\app\target」へ移動する。

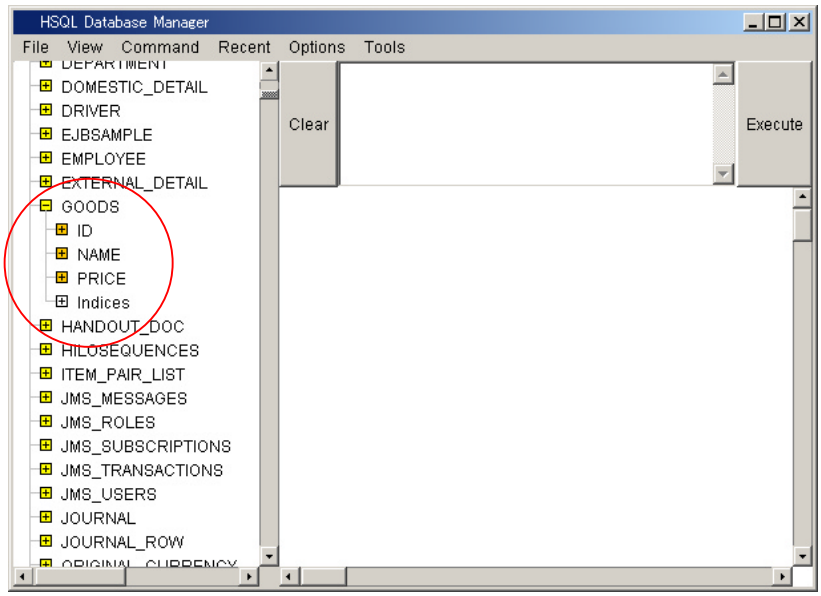
ディレクトリ上にある下記のファイルを実行する。

>initializeSchema.cmd



エラーが発生しなければ、テーブルの作成が成功している。確認するには、下記のコマンドを実行する。

```
>java -cp %JBOSS_HOME%\server\default\lib\hsqldb.jar org.hsqldb.util.DatabaseManager -url jdbc:hsqldb:hsql://localhost:1701
```

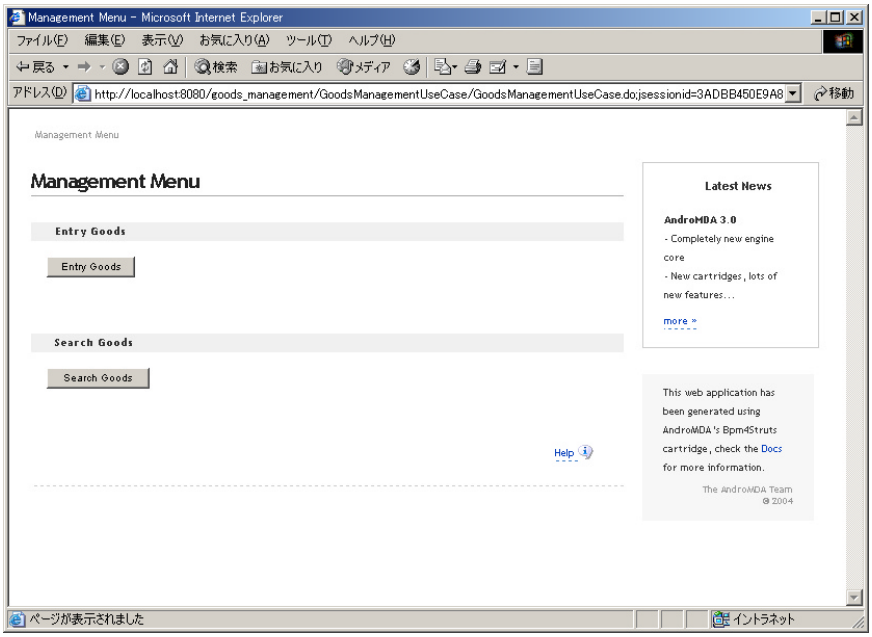


GOODS テーブルが存在することを確認する。

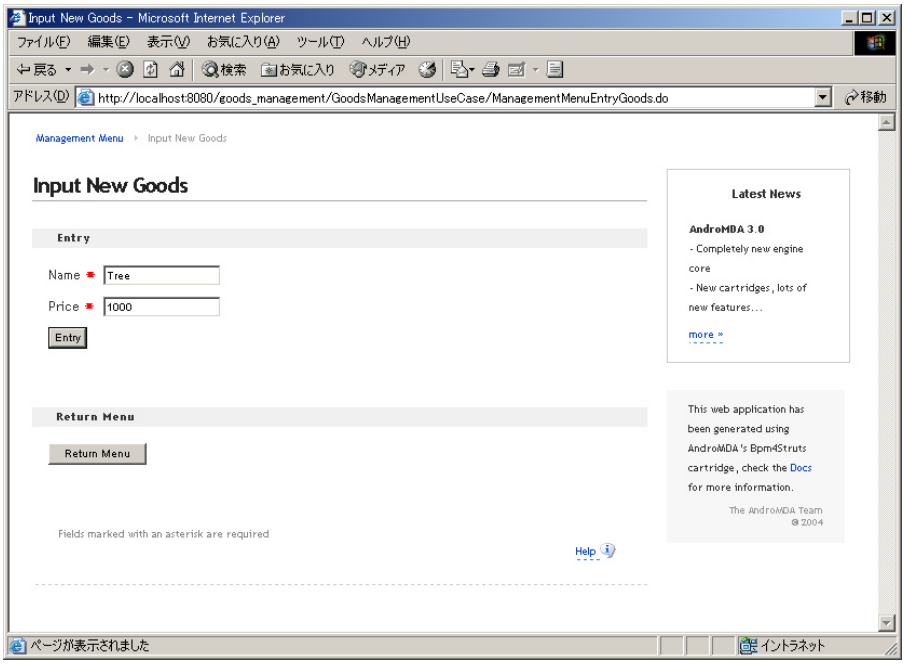
Web ブラウザを開き(例：IE)下記の URL へアクセスする。

http://localhost:8080/goods_management

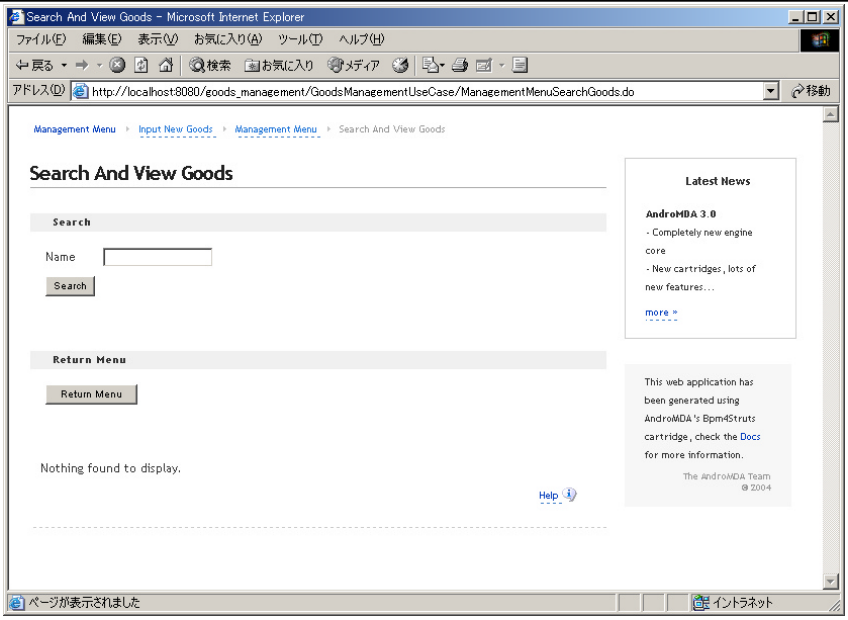
正常にアプリが実行された場合、下記のような画面が表示される。



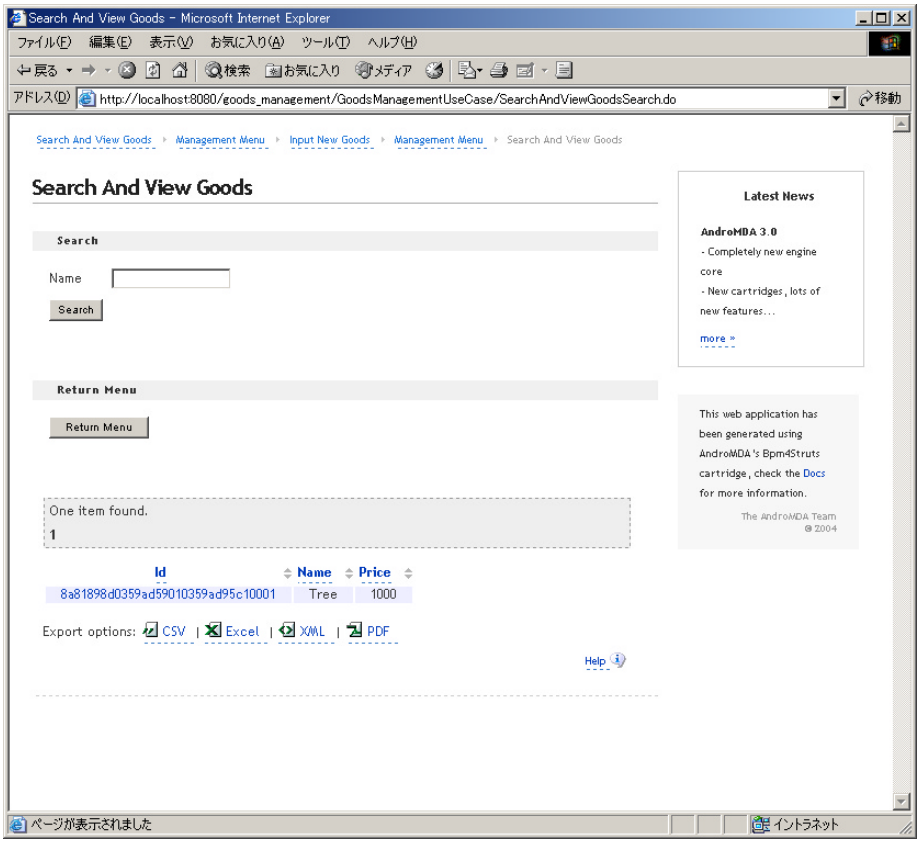
「Entry Goods」ボタンを押下すると登録画面へ、「Search Goods」ボタンを押下すると検索画面へ遷移する。



登録画面で「Name」と「Price」に値を入力し、「Entry」ボタンを押下すると DB へ登録する。
「Return Menu」ボタンを押下するとメニューへ戻る。

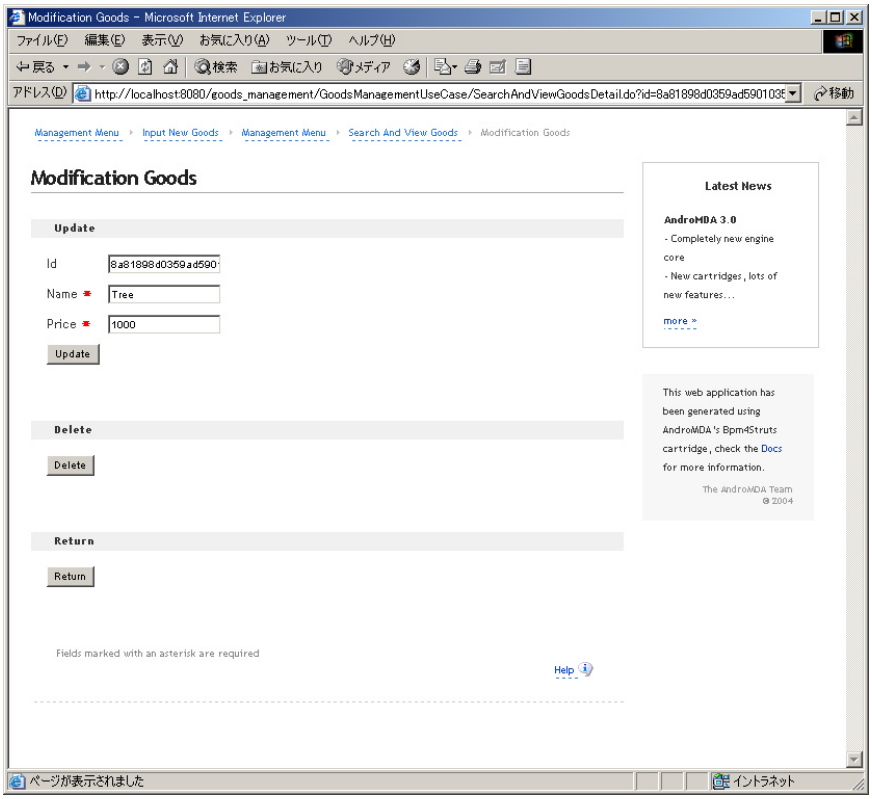


検索画面で「Search」ボタンを押下すると検索を実行する。



検索結果の Id カラムのハイパーリンクにアクセスすると、データの詳細(更新・削除)画面へ遷移する。

「Return Menu」ボタンを押下するとメニューへ戻る。



「Name」や「Price」を変更し「Update」ボタンを押下するとデータの更新を実行する。
「Delete」ボタンを押下するとデータの削除を実行する。
「Return」ボタンを押下すると検索画面へ戻る。