Java WEB アプリケーション
高速開発プラットフォーム
ユーザガイド

version 3.0

Shanghai XinJiang Software Co.,LTD.

2005. 6.18

目次

第 0 章　このマニュアルについて

このマニュアルの読者は J2SDK、Tomcat、Apache Struts、JSP(Java Server Pages) 、XML、Javamail、log4j(Log for Java)などのソフトウエアライブラリプロダクトの使用法を習得していることを前提としています。本文の中でこれらのソフトと関係がある内容が現れる時には、その詳しい説明を行いません。各ソフトウエアのドキュメントを参照してください。

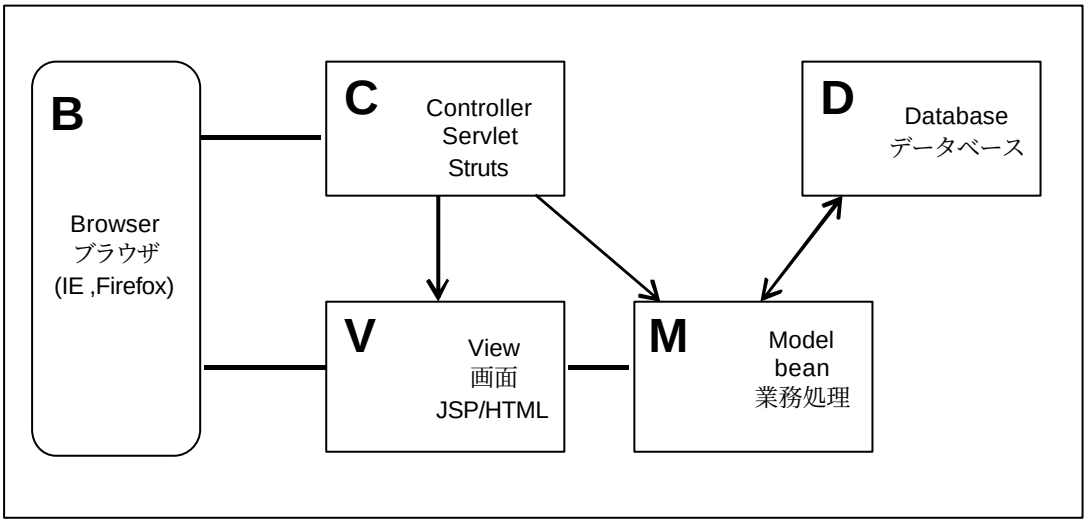
第 1 章 概要

本章では XJ@IRIS について紹介します。XJ@IRIS は Java による Web アプリケーションの高速開発プラットフォームです。

第 1節 開発理念

第 1 項 BMVCD

XJ@IRIS はWEB アプリケーションの系統的な五大要素 BMVCD を有機的に結び付けます。BMVCD とは Browser（ブラウザー）、 Model（業務処理）、 View（JSP 画面）、 Controller（コントローラ）、 Database（データベース）の省略語です。
BMVCD の関係は次の図の通りです：

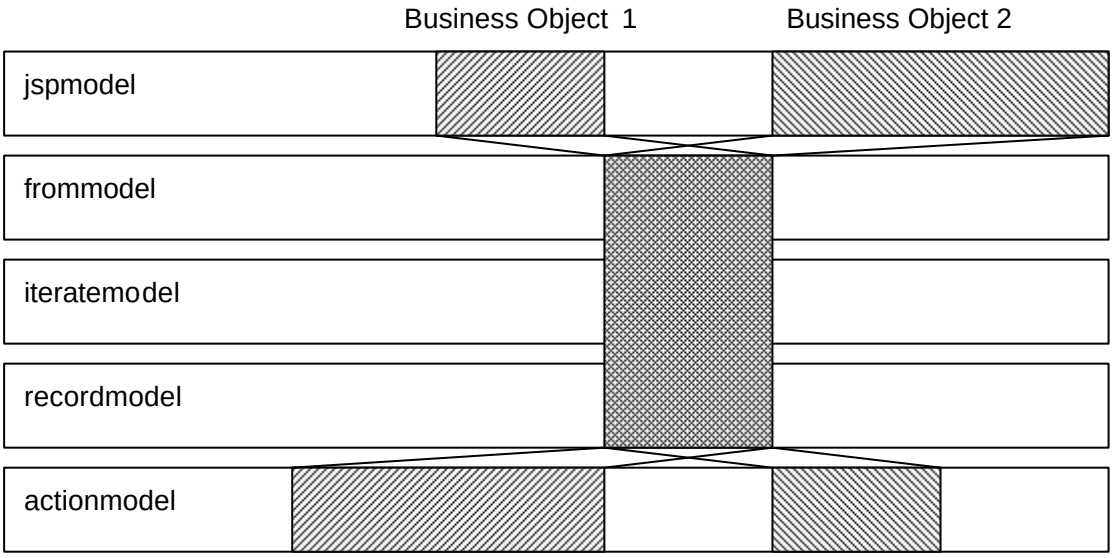


第 2 項 MDA（モデル駆動アーキテクチャ）

XJ@IRIS は先進的な開発理念である Model Driven Architecture(MDA)を基本としたアプリケーション構築プラットフォームです。プログラムソースの自動生成により、開発者の単純な繰り返し性コーディングの時間を省き、さらに WEB アプリケーションに統一のコーディングスタイルにあわせることができます。これは将来のメンテナンス性に優れ、同時にコードの間違いが起こる確率を大幅に減らします。XJ@IRIS は近年広く使われている Apache Struts Framework の上にアプリケーションを生成します。

第 3 項 ビジネスオブジェクト

ビジネスオブジェクトとは、プロジェクト中の 1 つの汎用的な機能に必要なモデルをまとめたものです。ビジネスオブジェクトはプロジェクトからエクスポートして別のプロジェクトにインポートすることができます。プロジェクトを機能ごとに分割しておくことにより、機能の再利用を促進します。プロジェクト内で定義する各種モデル要素について、プロジェクト内のビジネスオブジェクトに指定することができます。ビジネスオブジェクトは各種モデルを縦にまとめるものと考えることができます。



第 4 項 デファクトスタンダード

XJ@IRIS の生成するアプリケーションは Java を使った Web アプリケーションとしてデファクトスタンダードとして定着している以下の技術を応用しています。

Struts Framework (Apache Project)
Java Server Pages (JSP)
Java Servlet
Java Database Connectivity (JDBC)

また XJ@IRIS 本体及びランタイムライブラリ irislib ともに Pure Java で開発されています。

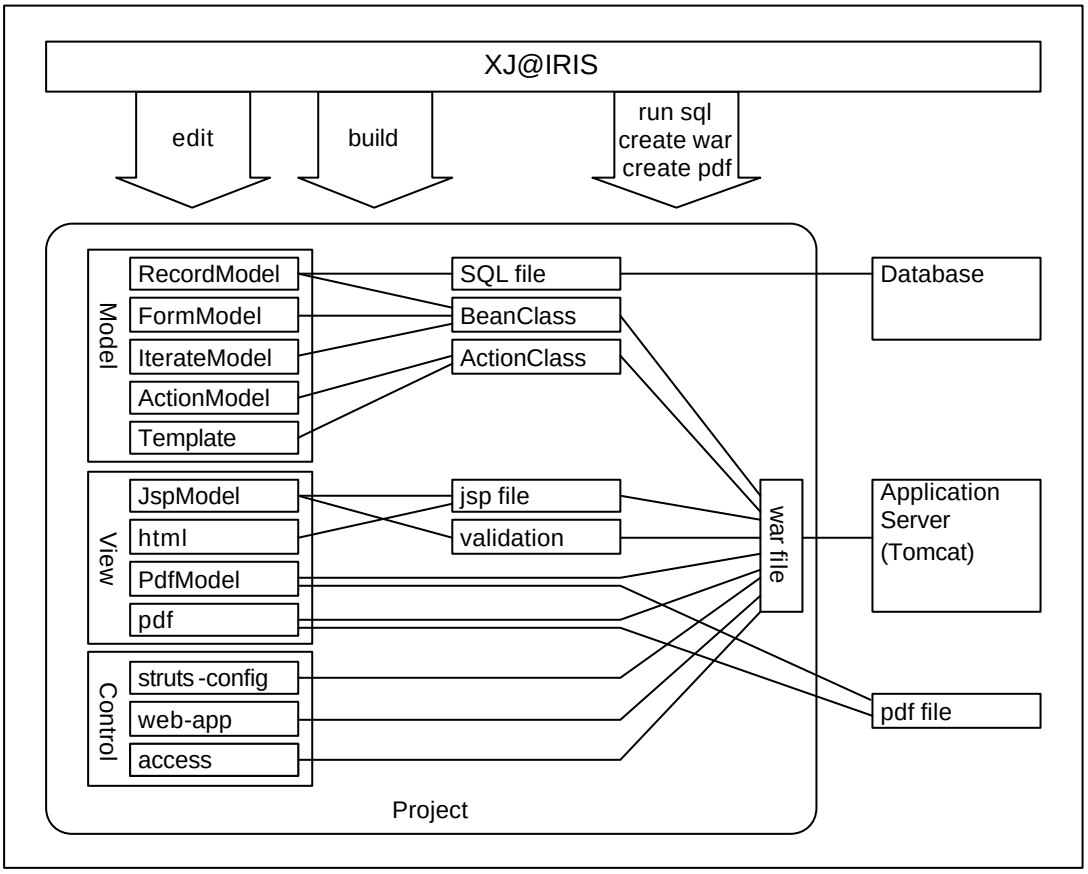
第 2節 XJ@IRIS での開発概要

第 1 項 開発手順

XJ@IRIS の機能とその手順の概要について説明します。

XJ@IRIS では各種モデルファイルを定義し、そこからJava ソースコード、JSP ファイルなどを自動生成します。ユーザは基本的にモデルファイルを編集することで目的のアプリケーションを開発します。その後 XJ@IRIS からビルドを実行するとWeb アプリケーションの動作に必要なファイルがすべて自動生成されます。また、モデルファイルもユーザがすべて定義する必要はなく各種のウイザード機能によって典型的なモデルを生成することができます。ユーザはウイザードにより生成されたモデルを変更しながら目的の動作に近づけることで短時間で開発を進めることができます。

各種モデルファイルについでは後で説明します。



第 2 項 GUI（グラフィカルユーザインターフェース）

XJ@IRIS モデル編集のための GUI を用意しています。この GUI 画面では各種モデルファイルの編集を支援するための機能が含まれています。入力形式の表示や入力値の候補からの選択入力、また参照値から定義位置にジャンプする機能などがあります。

第 3 項 強力なウィザード

ウィザードの機能を使って、一般的な WEB アプリケーションプログラムについて簡単な定義入力からアプリケーションを自動生成することが可能です。ウィザードは生成に必要なパラメータを入力することで、いくつかの画面とサーバー処理（アクション）を組み合わせた一連のモデルを生成します。ユーザはこれを部品として利用し、または改造することで独自のアプリケーションを開発します。

以下のウィザードがあります。

	名称	主要な機能
1	table wizard テーブルウィザード	複数のテーブル、および他のテーブルと外部リンクするビューを作成しその一覧表示、追加、修正、削除画面を持った Web アプリケーションを自動生成します。
2	view wizard ビューウィザード	ビューを作成しその一覧画面をもつアプリケーションを自動生成します。
3	search wizard 絞込みウィザード	テーブルの中でユーザ入力の条件によって表示件数を絞り込むような画面をもつアプリケーションを自動生成します。
4	login wizard ログインウィザード	ログイン画面及びログアウト画面とその処理を生成します。
5	category wizard 分類ウィザード	一覧とその詳細の関係を持つ 2つのテーブルまたはビューを連携して表示する画面を自動生成します。
6	pdf wizard 帳票ウィザード	テーブルあるいはビューからPDF 帳票の出力する機能を持った web アプリケーションを自動生成します。

第 4 項 データベース仮想化（豊富なデータベースサポート）

XJ@IRIS はデータベースに依存しないアプリケーション開発をサポートするためのライブラリを用意しています。この API を使ってデータベースアクセスを行うことで開発後、アプリケーションを別のデータベースで運用することが可能になります。

XJ@IRIS は以下の現在最も良く使われているデータベースをサポートしています。

データベースの名称	バージョン
Oracle	8.0/9

DB2	8.1 PE/EE
SQLServer	2000
PostgreSQL	7.3

また、アプリケーション開発を容易にするための高速な小型データベースをサポートしています。

データベースの名称	バージョン
xjPing	1.0

第 3節 XJ@IRIS の特徴と機能

第 1 項 SQL生成・実行

レコードモデルからSQL を生成、実行します。各データベース向けにテーブル・ビューの生成、削除 DDL SQL を生成・実行します。

第 2 項 一致性検査

XJ@IRIS は Web アプリケーションの War ファイル生成の前に、プロジェクト中のすべてのモデルファイルの中の要素に対して一致性検査を行います。この機能を通して、ユーザは実行時にのみ発生するエラーを早期に発見することができます。

第 3 項 Web アプリケーションのインストール（デプロイ）

War ファイルを生成したりTomcat の起動停止や War ファイルの所定のディレクトリへのコピーを行います。また War ファイルを生成せずに更新されたファイルのみをコピーするクイックインストールもできます。

第 4 項 DB スキーマ取得

既存のテーブルからスキーマ情報を取得し、レコードモデルやテーブルウィザードの設定として取り込むことができます。既にデータベースにテーブルが存在している場合にはさらに XJ@IRIS でスキーマ情報を入力する手間を省きます。

第 5 項 サイトナビ

Web アプリケーションの画面フローを2次元で表示、編集します。モデルファイルから JSP やアクションのリンク関係を解析しサイトマップとして表示します。この画面上で画面遷移を変更し

たり、各種モデルファイルを開くことができます。

第 6 項 アクセス制御

プロジェクト全体へのブラウザからのアクセス権限を一括して管理します。Web アプリケーションにブラウザからアクセスする際にログインしていないユーザにはページを表示しないなどの制限を行います。プロジェクトの一部のみを制限したり、ユーザに権限（ロール）を付けることもできます。

第 7 項 PDF 生成

データベースから取得したデータを元にPDFファイルを生成します。XJ@IRIS から PDF ファイルに出力する方法と、Web アプリケーションサーバに出力させる方法があります。

第 8 項 HTML 合成

データベースから取得したデータを元にPDFファイルを生成します。XJ@IRIS から PDF ファイルに出力する方法と、Web アプリケーションサーバに出力させる方法があります。

第 9 項 Ant ビルド

Apache Ant を使ったビルドができます。通常は XJ@IRIS の GUI から Java のコンパイルや War 生成を行いますが、XJ@IRIS がない場合でもコンパイル可能にするため、プロジェクトには Apache Ant 用のビルド定義ファイルも出力します。

第 10 項 他のツールとの連携

XJ@IRIS は他の HTML や Java 編集ツールなどとの連携を考慮しています。XJ@IRIS によって自動生成されたスケルトンの HTML ファイルに対して FrontPage や DreamWeaver を使って修飾を行うことができます。編集後は XJ@IRIS を使って JSP ファイルを生成します。 また Adobe Acrobat でスケルトン PDF ファイルのフォームを作成たり、JBuilder や Eclipse などの Java 開発環境を使って XJ@IRIS が自動生成した業務処理 Java ファイルに対して編集を行い、業務処理を完成させることができます。また、XJ@IRIS のプロジェクトを CVS(Concurrent Versions System)を使ってバージョン管理することができます。

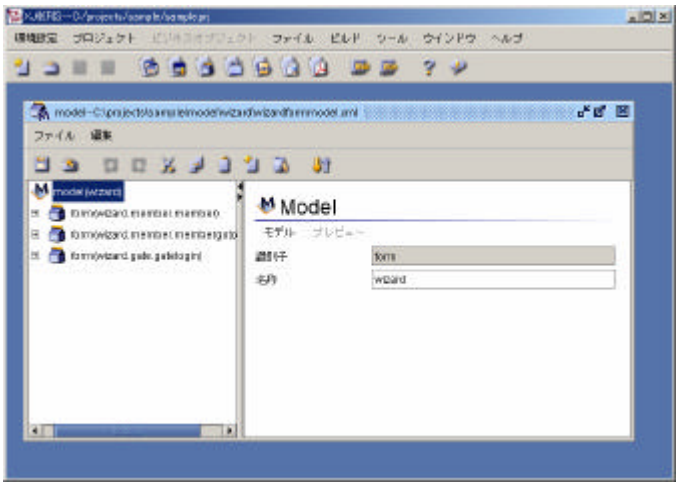
第 2 章 XJ@IRIS 基本操作

本章では XJ@IRIS の基本的な操作を説明します。

第 1 節 起動と終了

Windows スタートメニューのショートカットまたは XJ@IRIS インストールディレクトリの下の startup.cmd を使ってプログラムを開始します。終了するときはメインウインドウの閉じるボタンまたはメニューから閉じるを選択したください。

第 2 節 メイン画面



画面の一番上のタイトルバーには現在開いているプロジェクトのパスが表示されます。その下にメニューとツールバーがあります。その下にはファイル編集などの子フレームを複数開くことができます。

第 3 節 メニュー

メニューのグループ構成は以下の様になっています。

環境設定	このコンピュータにインストールされた XJ@IRIS に関する設定に関するグループです。ここで設定された内容はすべてのプロジェクトに使用されます。
プロジェクト	プロジェクトのオープン、クローズとウィザードに関するグループです。
ビジネスオブジェクト	ビジネスオブジェクトに関するグループです。
ファイル	プロジェクト中のモデルファイルやその他の編集すべきファイルのオープン、クローズなどに関するグループです。

ビルド	各種モデルファイルのビルドとWar ファイル生成に関するグループです。
ツール	各種ツールに関するグループです。 SQL 関連、War ファイルのインストール関連、DB スキーマ取得、サイトナビ、ログ表示が含まれます。
ウインドウ	子フレームの操作に関するグループです。
ヘルプ	XJ@IRIS のヘルプとバージョン情報に関するグループです。

第 4 節 ツールバー

ツールバーではメニュー項目の中でよく使われるものをすばやく起動することができます。各アイコンの上にマウスカーソルを合わせると、説明のツールチップが表示されます。

第 5 節 環境設定

第 1 項 必要な動作環境

XJ@IRIS をインストールする場合には以下の条件の環境設定が必要になります。

1. システム要件：

CPU：Pentium 3 1.0 GHz あるいはそれ以上の性能のプロセッサー
メモリ：256MB 以上
ハードディスクの空き空間：1GB 以上
OS：Windows 98/2000/XP

2. その他のソフトの環境：

XJ@IRIS を使用する前に、次の関連しているソフトウエア、ライブラリ環境のインストールを確認してください。

Java SDK	J2SDK1.4.2 推薦
Apache Tomcat	Apache Tomcat 4.1 Apache Tomcat 5.0

以下のライブラリは XJ@IRIS の slib ディレクトリにインストールされています。一部はシステム環境設定で変更することができます。

Struts	Struts 1.2
Javamail	Javamail 1.2
Log4j	Log4j 1.2.8
JavaHelp	JavaHelp 2.0.01
iText	iText 1.0.1

3.　データベースサーバと JDBC ドライバ

必要に応じてデータベースサーバとその JDBC ドライバを用意します。

XJ@IRIS のサポートしているデータベースサーバは以下のとおりです。

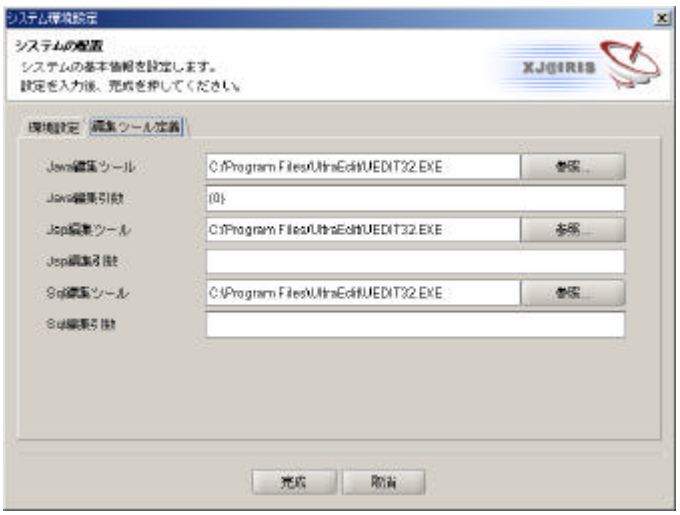
データベース サーバ	Oracle, DB2, SQLServer, PostgreSQL, または XJ@IRIS 添付の xjPing

第 2 項 システム環境設定

メニューからシステム環境設定を選択し内容を確認または変更します。

環境設定タブで Java パス、Tomcat パス、Tomcat 起動モード、Tomcat 起動命令、Tomcat 停止命令、 Tomcat 待ち時間を設定します。

Tomcat 待ち時間は Tomcat を起動/終了命令を発行した後に、 実際に起動/終了が完了するまでの待ち時間です。War ファイルのインストールでファイル削除などが失敗する場合は、長めに設定してください。

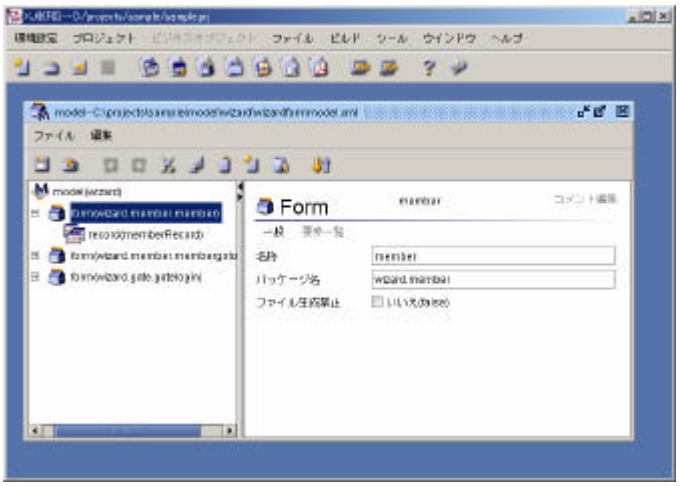


編集ツール定義タブで各種ソースファイルの編集を実行したときに使う編集ツールを設定しま

す。編集引数には編集するファイル名を{0}として設定します。編集引数に何も指定しない場合はファイル名のみが渡されます。編集ツールを指定していない場合は Windows 付属の Notepad.exe が起動されます。

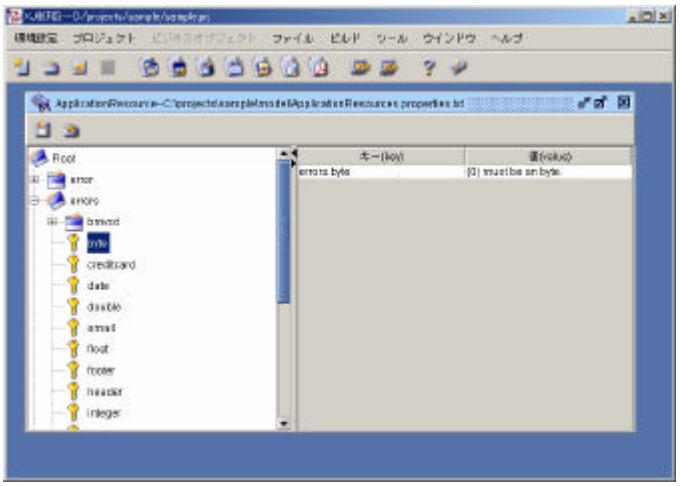
第 6 節 編集画面

メニューからファイル->オープンを選択しファイルを選択します。開くことのできるファイルはモデルファイルや struts-config などの XML ファイルとApplicationResource ファイルです。ファイルを選択するとメインウインドウ内に子フレームが開きます。
XML ファイルを開いた場合の画面は以下のようなものです。



子フレームのタイトルバーの下にメニューとツールバーがあります。その下の左側がツリーペイン、右側が編集ペインです。ツリーペインから1つのノードを選択すると編集ペインにそのノードが表示され編集することができます。メニューとツールバーからはこの ファイルに関する保存、クローズと編集操作の実行ができます。

ApplicationResouce.txt を選択した場合の画面は以下のようなものです。

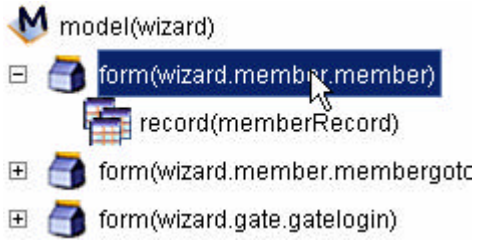


子フレームのタイトルバーの下にツールバーがあります。その下の左側がツリーペイン、右側が編集ペインです。ツリーペインから1つの フォルダまたはキーを選択すると編集ペインにキーと値が表示されます。ここで値を編集することができます。メニューからはこの ファイルの保存とクローズが実行できます。

第 7 節 ノード操作

第 1 項 ノードの選択

ツリーペインのノードから1つをクリックすることでノードを選択します。右側の編集ペインには選択したノードに応じた編集画面が表示されます。



また、この位置でマウスの右クリックをするとポップアップメニューが表示されます。

第 2 項 ノードの切り取り

左側のツリーペインからひとつのノードを選択し、メニュー、ツールバー、またはポップアップメニューから切り取りを実行します。

現在のノードが削除可能ならば、切り取りを実行してよいかどうかのメッセージボックスが表示されます。はいを選択することで切り取りが実行されます。実行後は切り取ったノードはクリップボードに保存されます。現在のノードを切り取ることができないときはその旨のメッセージボックスが表示されます。

第 3 項 ノードのコピー

ツリーペインからノードを選択し、メニュー、ツールバーまたはポップアップメニューからコピーを実行します。選択されたノードがクリップボードに保存されます。

第 4 項 ノードの貼り付け

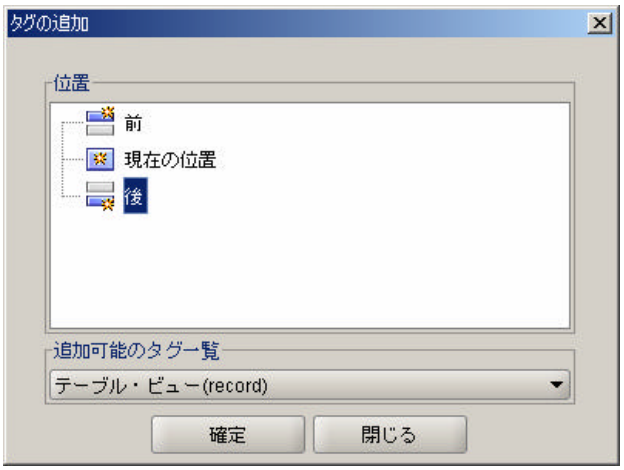
ツリーペインからノードを選択し、メニュー、ツールバーまたはポップアップメニューの貼り付けを実行します。

現在のクリップボードに保存されたノードがある場合は、貼り付けの位置を指定するダイアログボックスが表示されますので、貼り付け位置を指定して確定ボタンを押します。クリップボード中に保存されたノードがないときは、その旨のメッセージが表示されます。

第 5 項 ノードの追加

ツリーペインからノードを選択し、メニュー、ツールバーまたはポップアップメニューの追加を実行します。

ダイアログが表示されますので、追加するノードのタイプと現在のノードからの位置を入力します。確定ボタンを押すと指定した位置に新しいノードを追加します。



第 6 項 ノードの削除

ツリーペインからノードを選択し、メニュー、ツールバーまたはポップアップメニューの削除を実行します。

現在のノードが削除可能の場合は確認メッセージを表示し、了解ボタンを押すと削除が実行されます。 削除できない場合はその旨のメッセージを表示します。

第 7 項 ノードの編集を元に戻す

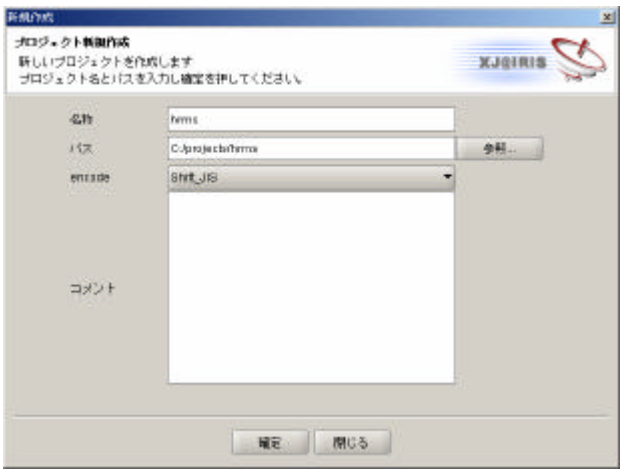
ツリーまたは編集ペインの編集を行った後、メニューまたはツールバーのアンドゥを実行します。 最後に行った編集を元に戻します。またアンドゥの取り消しで復帰することができます。

第 3 章　ウイザードの基本

本章ではウィザード機能の基本的な使い方を説明します。

第 1 節　プロジェクト作成

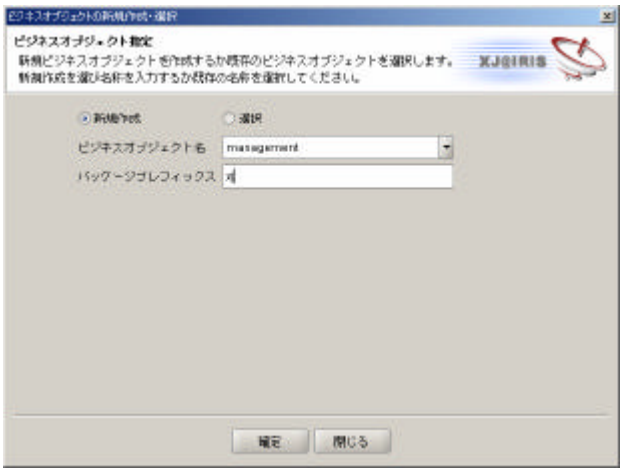
システム環境設定が正しいことを確認し、新しいプロジェクトを作成します。メニューからプロジェクト>新規作成を選択すると以下のダイアログが開きます。



名称を hrms とします。パスには適当なパスを設定し確定ボタンを押してください。これで指定したパスに hrms という名前のアプリケーションプロジェクトが作成されました。

第 2 節　テーブルウィザード (リンクなし)

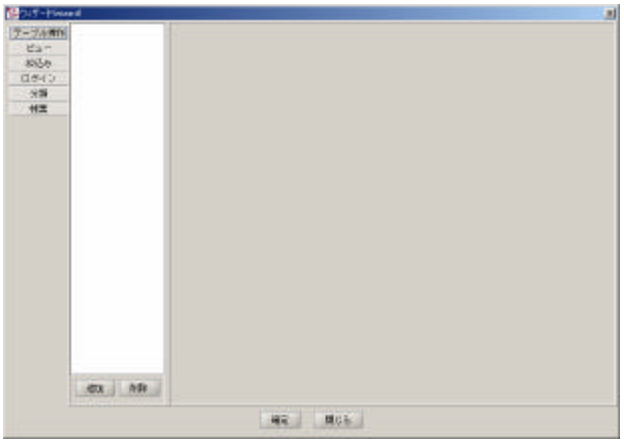
メニューからプロジェクト>ウィザードを選択します。ビジネスオブジェクトの新規作成・選択のダイアログが開きます。



ビジネスオブジェクト名に management、パッケージプレフィックスに xj と入力し確定ボタンを押します。

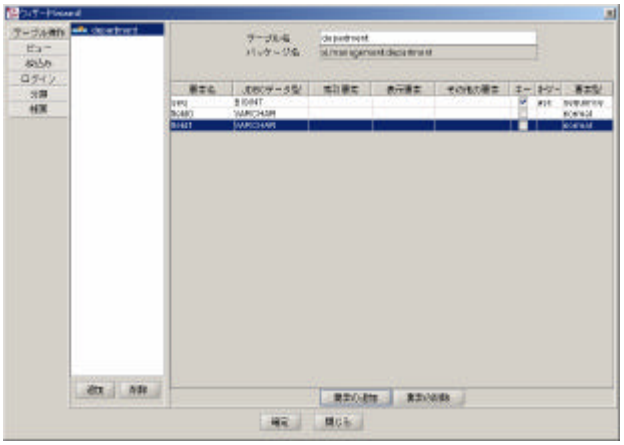
第 1 項 テーブル定義

ウィザードの定義ダイアログが開きます。

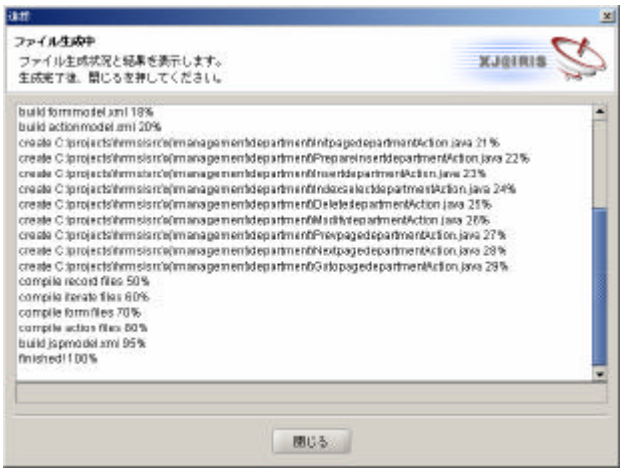


この画面の一番左にはウィザード選択のタブがあります。ダイアログが開いたときにはテーブル操作になっています。その右側にはテーブル一覧があります。初期には何も表示されていません。下方の追加、削除ボタンで操作します。その右側にはテーブル一覧から選択したテーブルの内容が表示されます。初期にはテーブルがないので表示されていません。

それでは追加ボタンを押してテーブルを追加してみましょう。



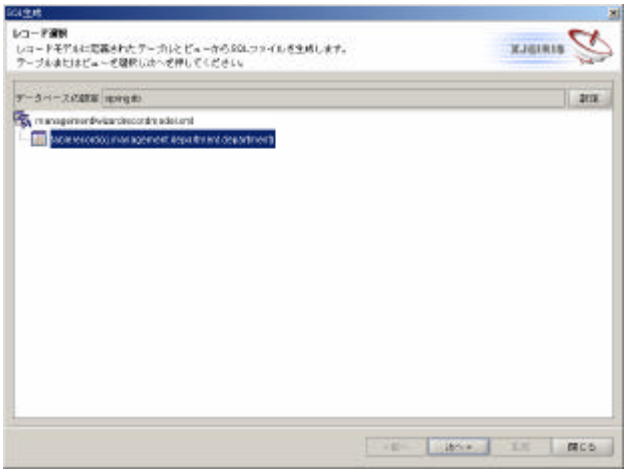
テーブル名には department と入力します。さらにテーブル内容下の要素の追加ボタンを押してこのテーブルに要素（フィールド）を追加します。2つ追加して要素名を name とphone としてください。このテーブルには seq 要素がありますがこれはこのままにしておきます。
入力が終わったら確定ボタンを押します。



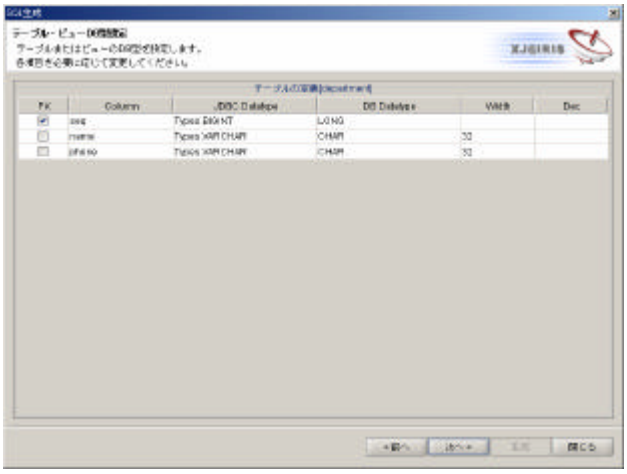
ウィザードはこの プロジェクトのアプリケーション の動作に必要な各種 ファイルの生成とコンパイルを自動的に行います。すべての生成が終わったら進捗ダイアログを閉じてください。

第 2 項 データベース

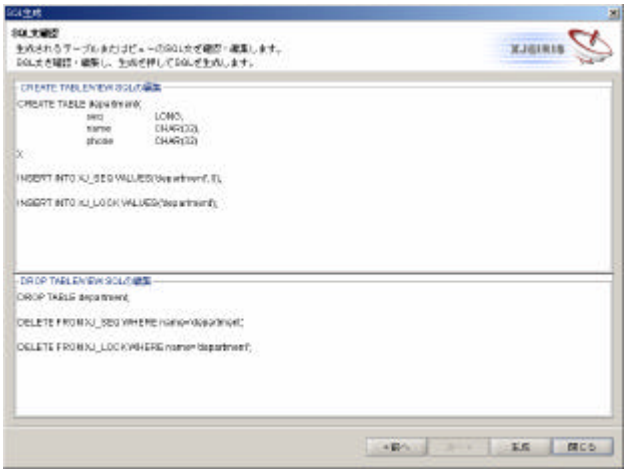
メニューからツール->SQL 生成を選択します。



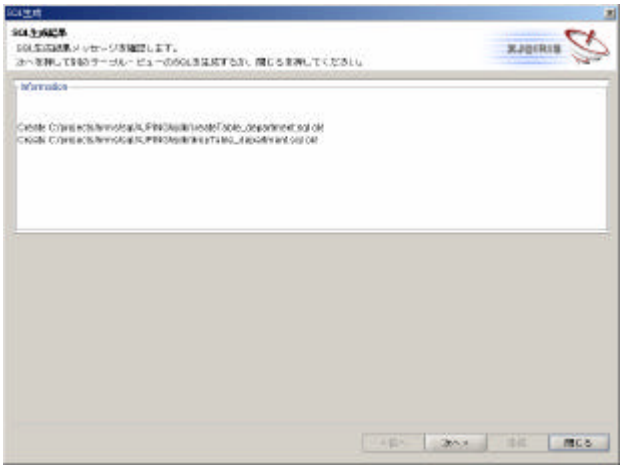
ここではデータベースとして xjPing を使用し、データベース名は xjpingdb とします。department テーブルを選択しダイアログ下方の次へのボタンを押します。



表示された画面ではテーブルのスキーマを変更可能ですが、ここでは変更の必要はないので次へを押します。



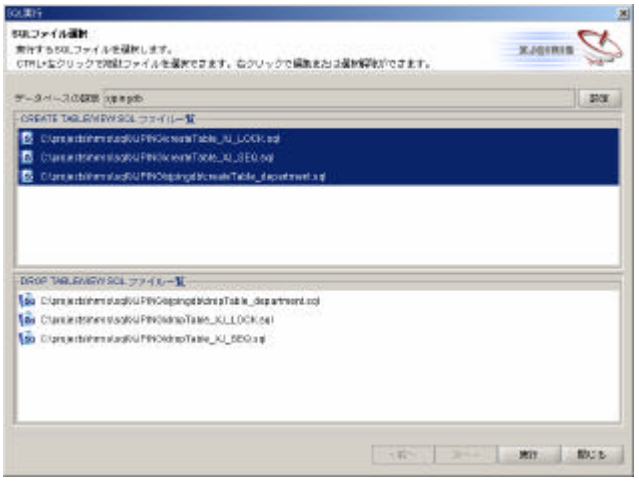
この画面では生成ボタンを押します。



これで SQL ファイルが生成されました。閉じるボタンを押してダイアログを閉じます。

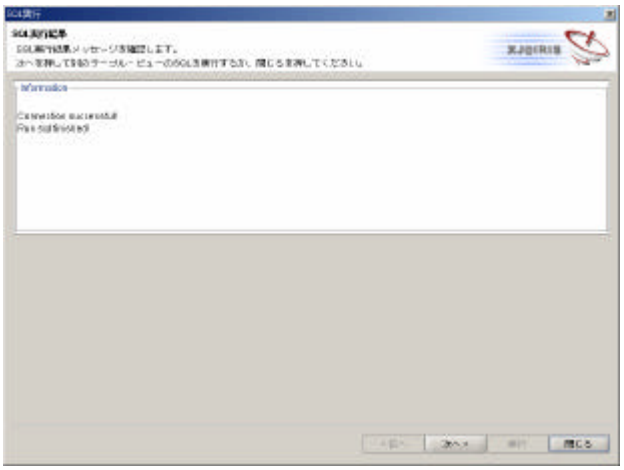
ここで生成された SQL ファイルを実行しデーターベースにテーブルを作成します。先にデータベースサーバを起動しておきます。

メニューからツール->SQL 実行を選択します。



初めてこの データベースを使う場合は XJ_SQL と XJ_LOCK テーブルが必要になります。

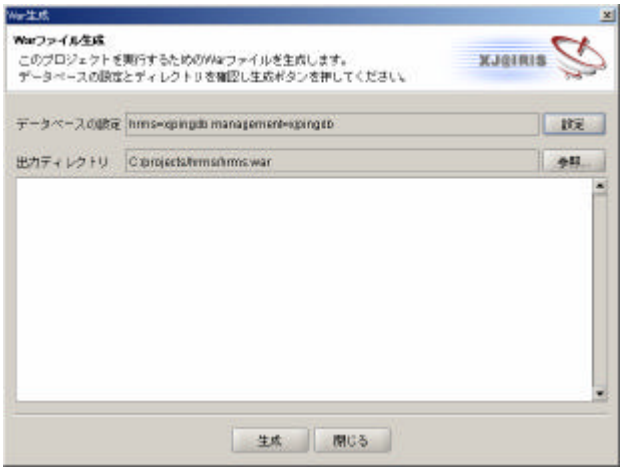
CREATE TABLE/VIEW ファイル一覧から createTable-XJ_SEQ.sql 、createTable-XJ_LOCK.sql とcreateTable-department.sql の３つのファイルを選択し実行を押します。



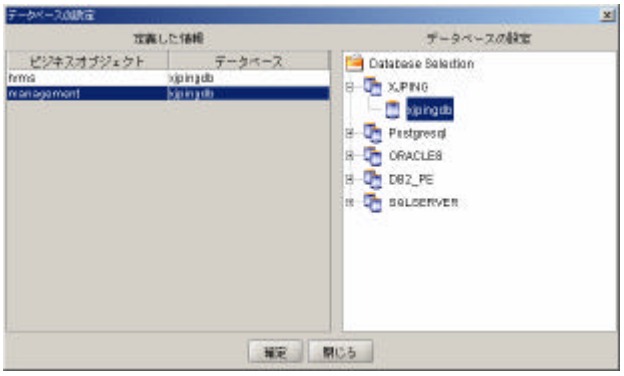
これでデータベースにテーブルが作成されました。閉じるボタンを押してダイアログを閉じます。

第３項 War ファイル

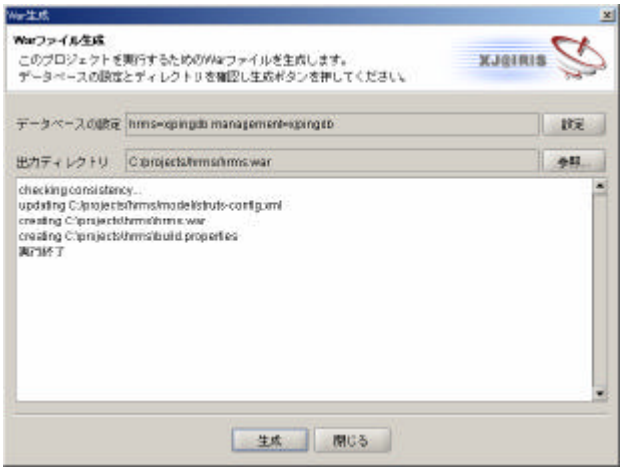
メニューからビルド>War ファイル生成を実行します。



先ず、データベースの設定を行います。設定ボタンを押してください。

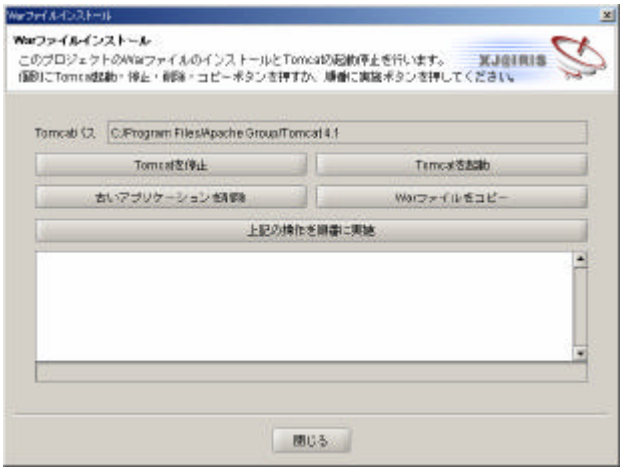


ビジネスオブジェクトmanagement のデータベースを xjpingdb に設定して確定を押します。War 生成画面に戻り生成ボタンを押すと hrms.war が生成されます。



閉じるボタンを押してダイアログを閉じてください。

次に、生成された War ファイルをインストールします。メニューからツール->War インストールを実行します。

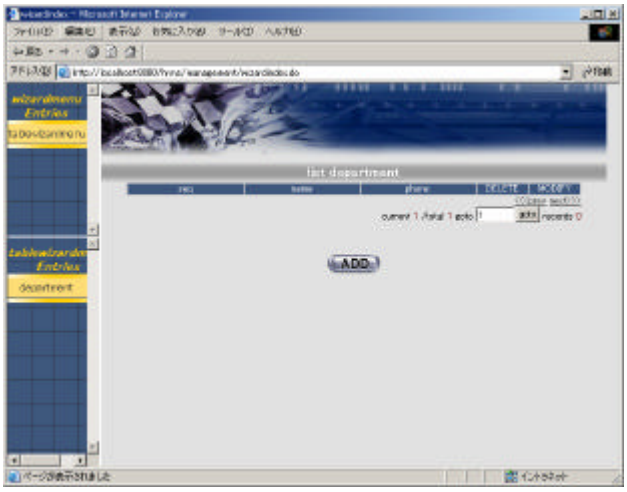


War ファイルをコピーとTomcat を起動ボタンを順に押してください。終わったら閉じるボタンを押してダイアログを閉じます。

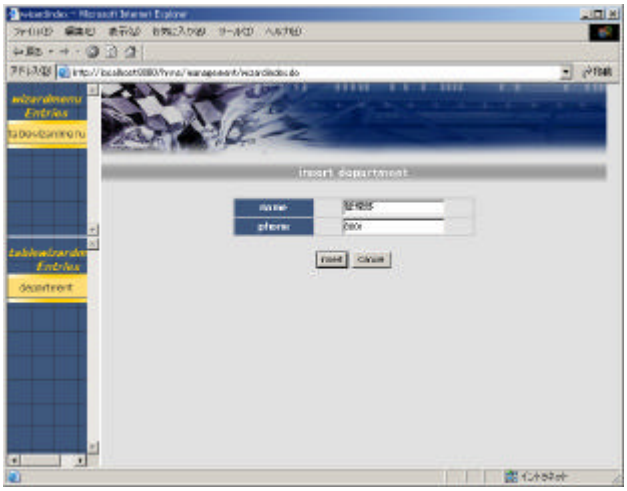
第 4 項 サンプルを実行する

ブラウザを開きのアドレス欄に
`http://localhost:8080/hrms/`
と入力します。

management のリンクをクリックし、次の画面に移動します。

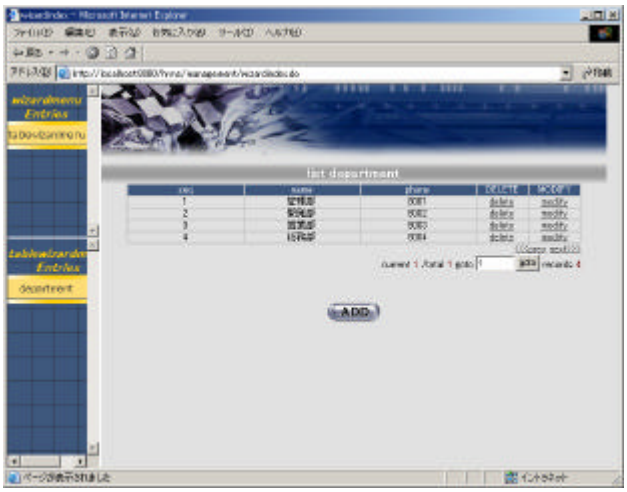


ADD のボタンを押しレコード追加の画面に移動します。

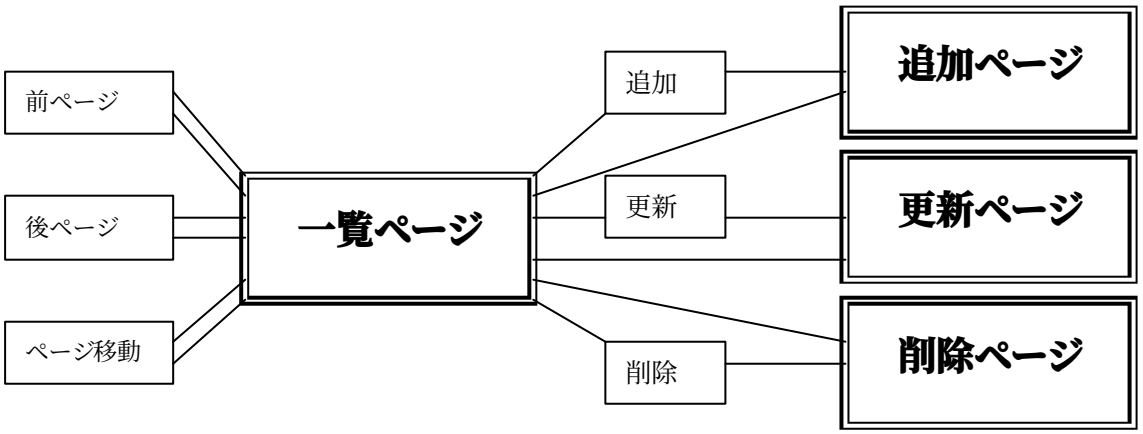


name と phone データを入力して submit をクリックします。ここでは以下の 4つのデータを順に登録します。

管理部 / 8001
開発部 / 8002
営業部 / 8003
総務部 / 8004

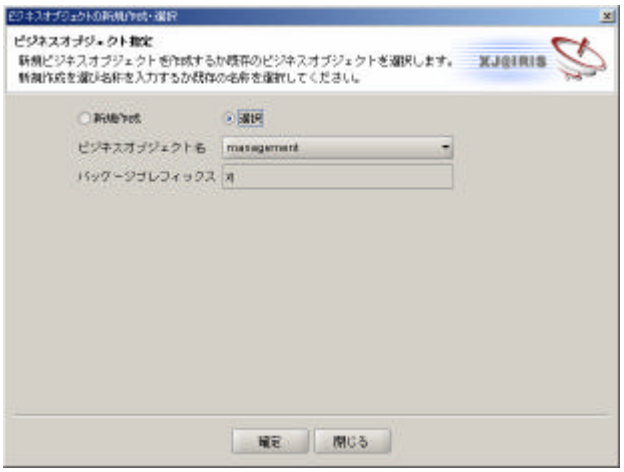


このアプリケーションの画面遷移は次のようになっています。



第３節　テーブルウィザード（リンクあり）

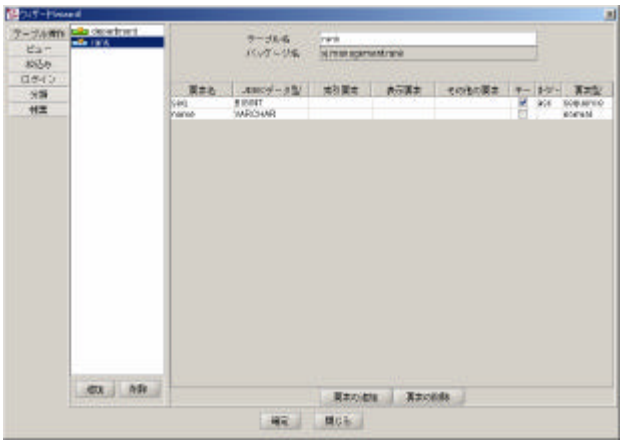
メニューからプロジェクト>ウィザードを実行します。以下のようなダイアログが開きます。



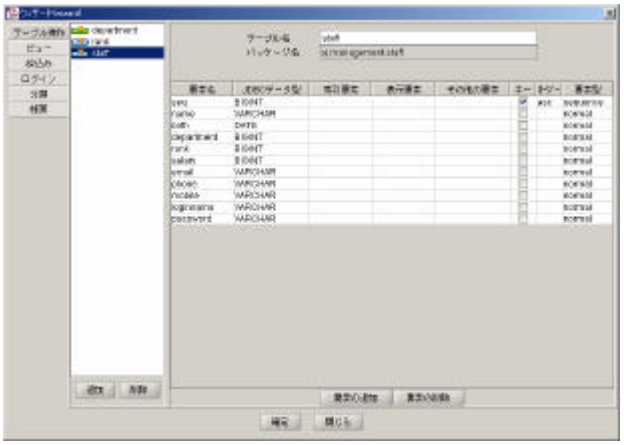
ビジネスオブジェクト名に management が選択されていることを確認して確定ボタンを押します。

第 1 項 テーブル定義

先ず、rank テーブルを追加します。rank テーブルには要素として name を追加します。



続いて、外部リンク付きの staff テーブルの設定を始めます。



staff テーブルを追加して、このテーブルに以下の フィールドを追加します。括弧つきの フィールドはその JDBC データ型にします。

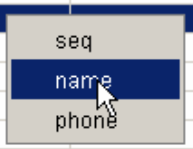
name, birth(DATE), department(BIGINT), rank(BIGINT), salary(BIGINT), email, phone,

mobile, loginname, password

その後、department の検索要素のセルをクリック後、右クリックし、ポップアップメニューから department の seq フィールドを選択します。

要素名	JDBCデータ型	索引要素	表示要素	その他の要素	キー	ｵｰﾀﾞｰ	要素型
seq	BIGINT				☑	asc	sequence
name	VARCHAR				☐		normal
birth	DATE				☐		normal
department	BIGINT				☐		normal
rank	BIGINT				☐		normal
salary	BIGINT				☐		normal
email	VARCHAR				☐		normal
phone	VARCHAR				☐		normal
mobile	VARCHAR				☐		normal
loginname	VARCHAR				☐		normal
password	VARCHAR				☐		normal

その後、隣の表示要素でも右クリックし name を選択します。



要素名	JDBCデータ型	索引要素	表示要素	その他の要素	キー	ｵｰﾀﾞｰ	要素型
seq	BIGINT				☑	asc	sequence
name	VARCHAR				☐		normal
birth	DATE				☐		normal
department	BIGINT	department.s...			☐		normal
rank	BIGINT				☐		normal
salary	BIGINT				☐		normal
email	VARCHAR				☐		normal
phone	VARCHAR				☐		normal
mobile	VARCHAR				☐		normal
loginname	VARCHAR				☐		normal
password	VARCHAR				☐		normal

これで staff テーブルの department テーブルへの外部リンクが完了しました。同様に staff テーブル中の rank テーブルへの外部リンクも完成させてください（索引要素は rank.seq、表示要素は name）。

その後確定ボタンを押します。これで staff テーブルのビューviewstaff が生成されます。

第 2 項 データベースと War ファイル

SQL 生成を実行し前節の department テーブルと同様に rank, staff, viewstaff の SQL ファイルを生成してください。その後 SQL 実行で createTable_rank.sql, createTable_staff.sql と createView_viewstaff.sql を実行してください。これでデータベースに rank テーブル、staff テーブルと viewstaff ビューが作成されます。

War 生成を実行し、War ファイルを再度生成してください。

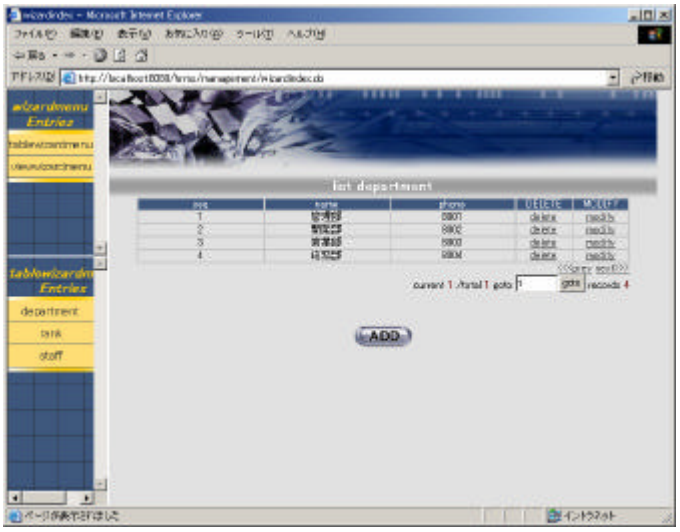
War インストールを実行します。既にTomcatが動作していますから、Tomcatの停止から行う必要があります。すべての操作を実行ボタンを押して War ファイルのインストールを実行します。

第 3 項 サンプルを実行する

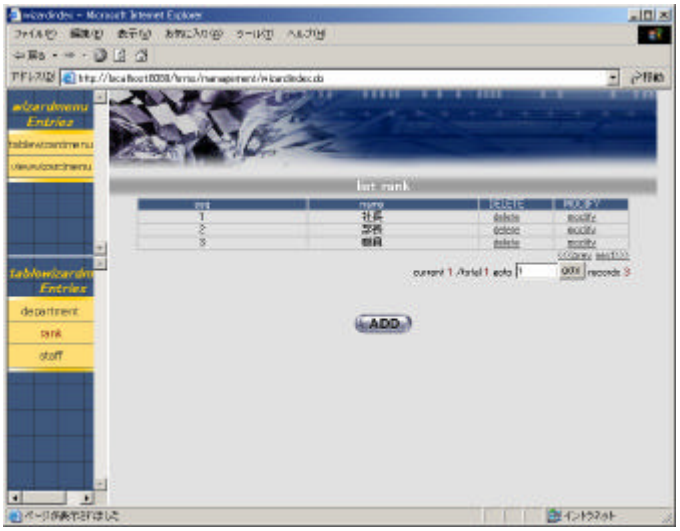
ブラウザを開きのアドレス欄に

```
http://localhost:8080/hrms/
```
と入力します。

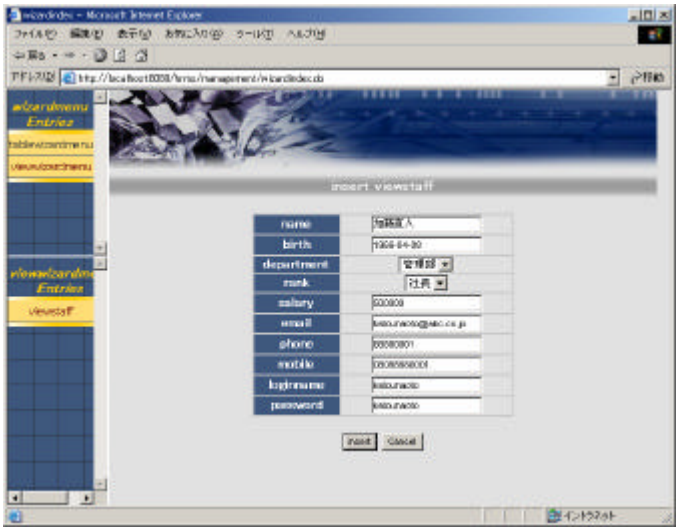
management のリンクをクリックし、次の画面に移動します。



左下で rank テーブルを選び、社長、部長、職員の 3レコードを追加します。
追加後の画面は以下のようになります。

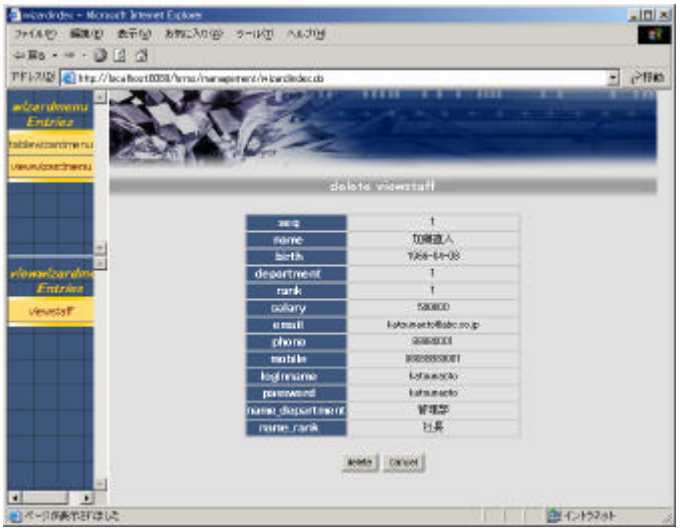


ここから、従業員情報の入力を開始します。右上の viewwizardmenu をクリックしてから左下の viewstaff をクリックして ADD ボタンをクリックすると以下の画面が表示されます。



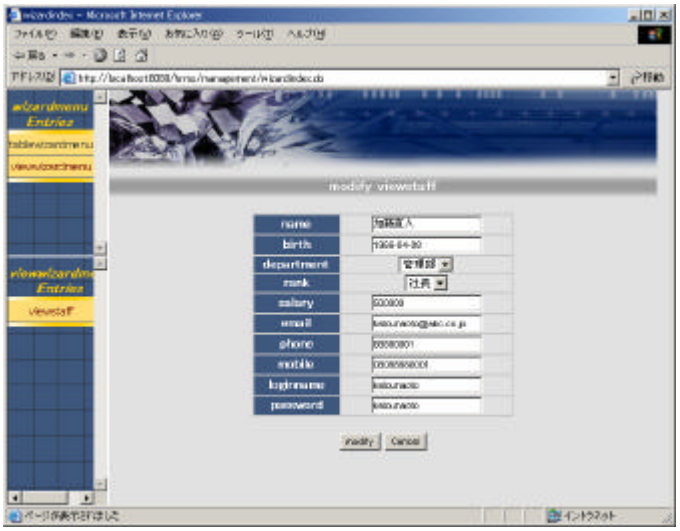
部門を入力する時に文字を入力する必要はありません。代わりに department テーブルに入力した情報が入ったコンボボックスから選択します。同じように役職の入力でもrank テーブルの情報から選択します。

従業員一覧画面のそれぞれの行の一番右側に modify、delete のリンクがあります。
delete 確認画面は以下の様になります。



delete をクリックしてレコードを削除します。

modify 確認画面は以下の様になります。



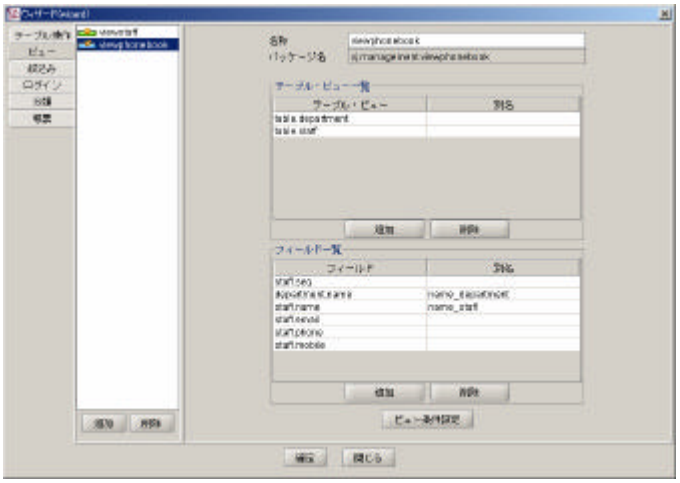
modify をクリックして情報を修正します。

第 4 節　ビューウィザード

従業員の役職と部署によっては、異なる従業員情報の一覧が望まれるかもしれません。ビューウィザードを使うとこのような画面を簡単に追加することができます。

第 1 項　ビューの定義

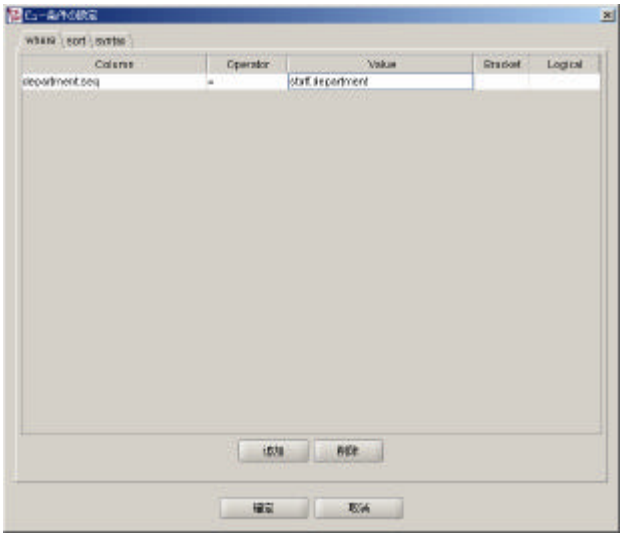
ここでは住所録ビューを作成します。ウィザードを実行しビジネスオブジェクト選択のダイアログで management を選択し確定ボタンを押してください。



ダイアログ左側のウィザード選択タブからビューを選び、下方の追加ボタンでビューウィザードを追加します。名称には viewphonebook と入力します。テーブル・ビュー一覧に department と staff を追加します。テーブル・ビューの下の追加ボタンで行を追加し追加した行のテーブル・ビューの欄を左クリック後、右クリックするとポップアップメニューからテーブルまたはビューを選択入力できます。同様の方法でフィールド一覧に画面の様に department テーブルの name、

staff テーブルの seq、name、email、phone、mobile を追加します。name という名前が 2つ存在しますのでこの 2 つには別名の欄にそれぞれ name_department と name_staff と入力します。

ビュー条件設定ボタンを押して条件設定画面を表示します。



追加ボタンを押して条件を以下の様に設定します。

department.seq = staff.department

ここでも右クリックでポップアップメニューからの選択入力ができます。

確定ボタンを押して保存し前の画面に戻ります。

これでビューウィザードの定義は完了です。確定ボタンを押して関連するファイルを生成します。

第 2 項 データベースと War ファイル

SQL 生成を実行し前節と同様に viewphonebook の SQL ファイルを生成してください。その後 SQL 実行で createView_viewphonebook.sql を実行してください。これでデータベースに viewphonebook が作成されます。

War 生成を実行し、War ファイルを再度生成してください。

War インストールを実行し、すべての操作を実行ボタンを押して war ファイルのインストールを実行します。

第 3 項 サンプルを実行する

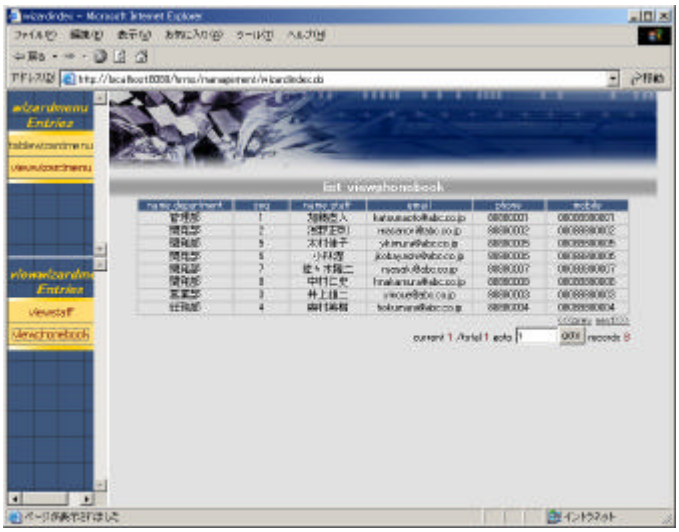
ブラウザを開きのアドレス欄に

```
http://localhost:8080/hrms/
```

と入力します。

management のリンクをクリックし、次の画面に移動します。

左上の viewwizardmenu をクリック後、左下で viewphonebook をクリックすると以下のような画面になります。

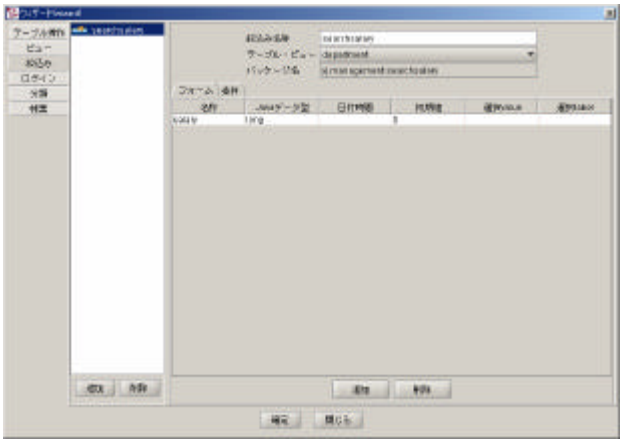


第 5 節 絞込みウィザード

従業員の情報を閲覧するとき、いくつかの条件に合う従業員の情報だけを一覧にしたいことがあります。絞込みウィザードを使うとこのような画面を簡単に作成することができます。

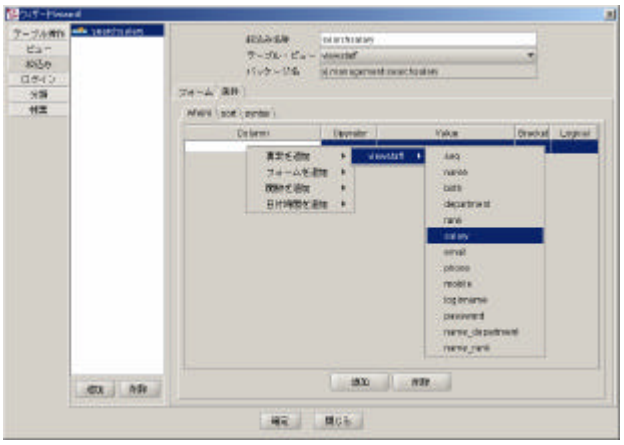
第 1 項 絞込みの定義

ここでは給与額での従業員の絞込み画面を作成します。ウィザードを実行しビジネスオブジェクト選択のダイアログで management を選択し確定ボタンを押してください。

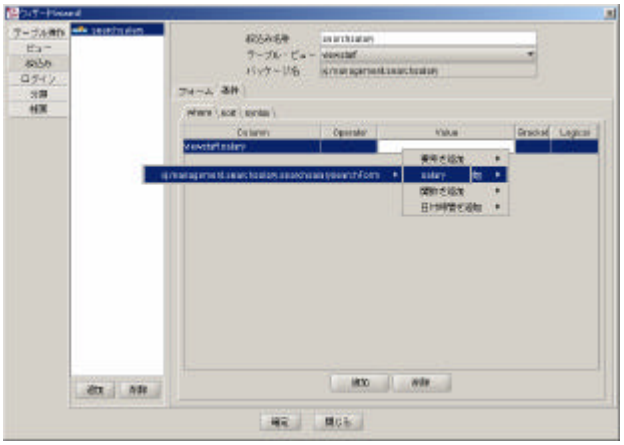


ダイアログ左側のウィザード選択タブから絞込みを選び、下方の追加ボタンでビューウィザードを追加します。名称には searchsalary と入力します。テーブルは viewstaff を選択してください。フォームタブの下の追加ボタンでフィールドを追加します。ここではフィールドの名称として salary を入力します。Java データタイプは long とします。
選択 value、選択 label に何も入力しないとフォームはテキストボックスになりユーザは自由に値を入力します。選択 value、選択 label に外部リンクしたテーブルの値を入力するとフォームはセレクトボックスになり、ユーザはその中から値を選びます。ここでは空白とします。

次に検索条件の設定をします。条件タブを選んで ください。



where タブで column フィールドとして staffview.salary、Operator として>=を設定します。



value フィールドとして PersonnelMnagement.staff.SearchsalaryForm.salary を設定します。

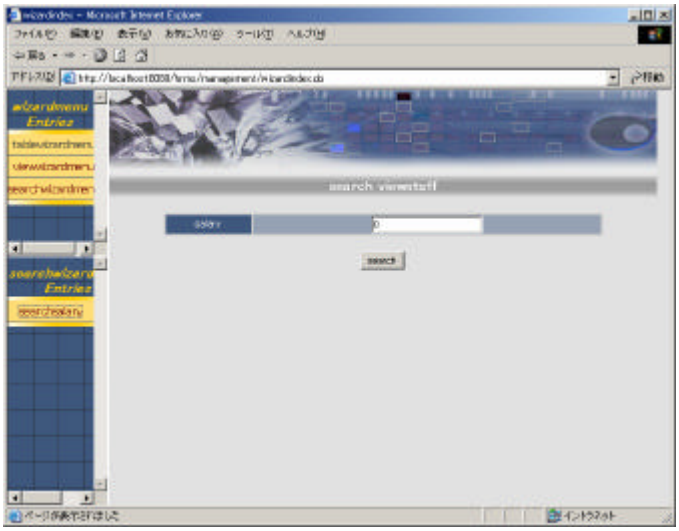
これで絞込みウィザードの設定は完了です。確定ボタンを押して関連するファイルを生成してください。

第 2 項 War ファイル

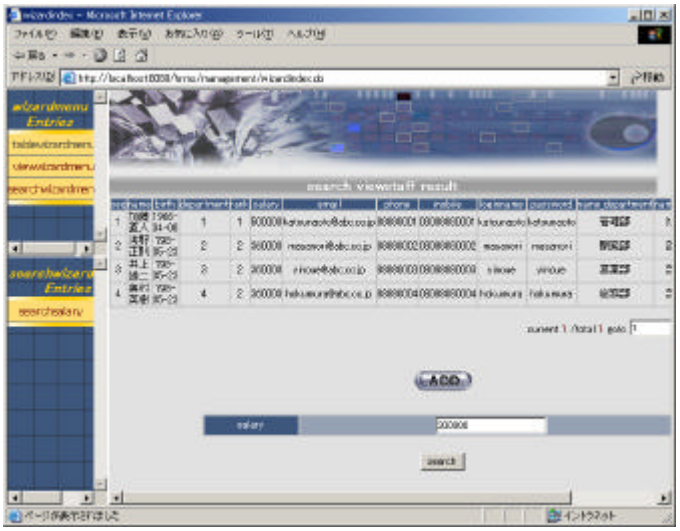
今回は SQL の生成と実行は不要です。前節と同様に War ファイルの生成とインストールを行ってください。

第 3 項 サンプルを実行する

ブラウザを開きのアドレス欄に `http://localhost:8080/hrms/`と入力します。
wizardindex 画面で左上の searchwizardmenu をクリックした後 searchsalary をクリックすると以下の画面が表示されます。



300000 を入力した後に、search ボタンをクリックすると、検索結果の一覧画面が表示されます。

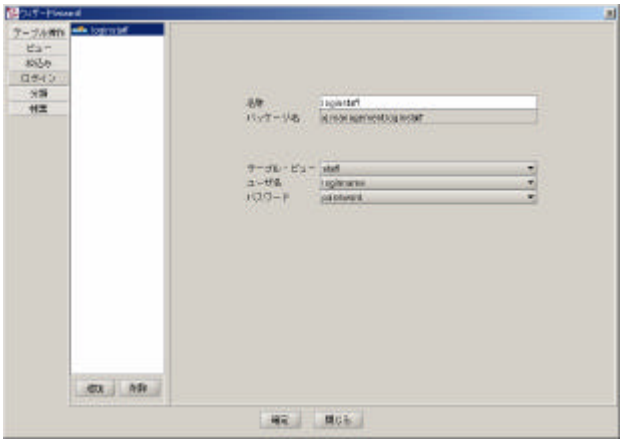


第 6 節　ログインウィザード

ログインウィザードを使ってアプリケーションに簡単にログイン画面を追加することができます。

第 1 項　ログインの定義

ここでは staff テーブルを使ってログイン画面を作成します。ウィザードを実行しビジネスオブジェクト選択のダイアログで management を選択し確定ボタンを押してください。



ダイアログ左側のウィザード選択タブからログインを選び、下方の追加ボタンでログインウィザードを追加します。名称には loginstaff と入力します。テーブル・ビューは staff を選択してください。ユーザ名、パスワードはこのテーブルのフィールドの中から選択します。ここではそれぞれ loginname と password を選択します。ログイン画面でのログイン名、パスワードに対応するフィールドになります。

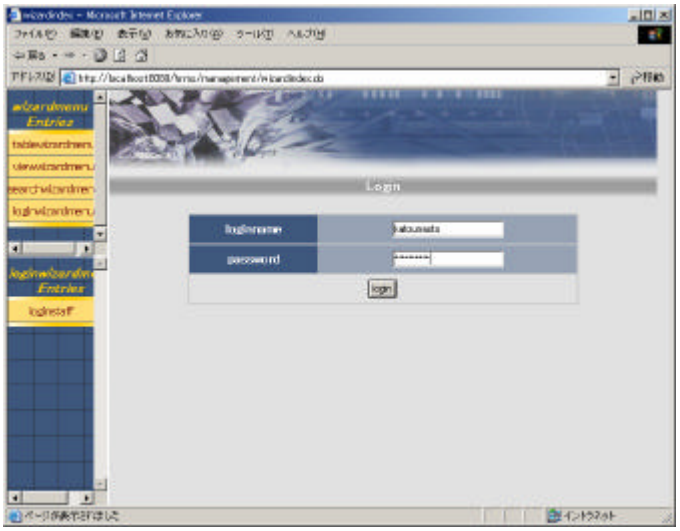
これでログインウィザードの設定は完了です。確定ボタンを押して関連するファイルを生成してください。

第 2 項 War ファイル

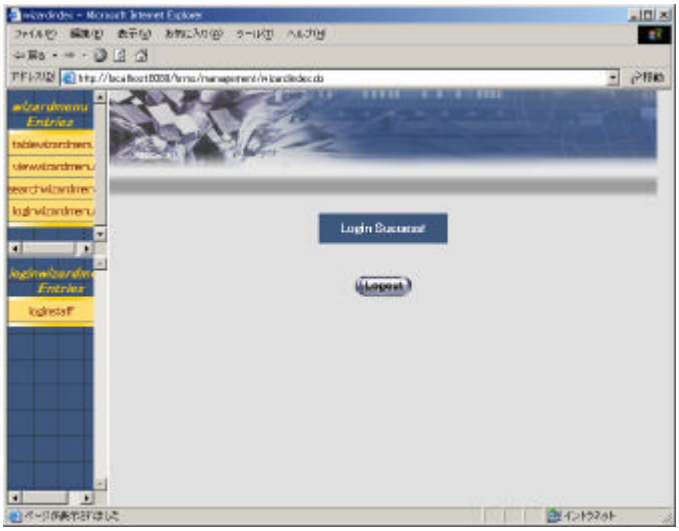
今回は SQL の生成と実行は不要です。前節と同様に War ファイルの生成とインストールを行ってください。

第 3 項 サンプルを実行する

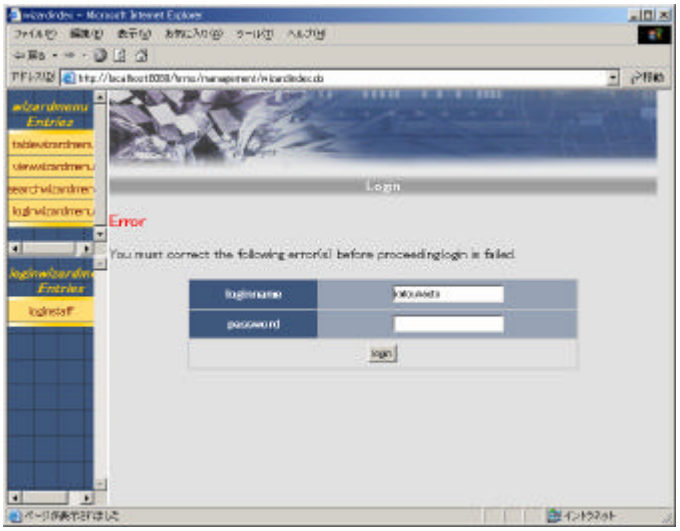
ブラウザを開きのアドレス欄に `http://localhost:8080/hrms/` と入力します。wizardindex 画面で左上の loginwizardmenu をクリックした後 loginstaff をクリックすると以下の画面が表示されます。



ユーザ名とパスワードを正しく入力してloginのボタンをクリックすると次のような画面が表示されます。



ユーザまたはパスワードが正しくない場合は次のような画面になります。



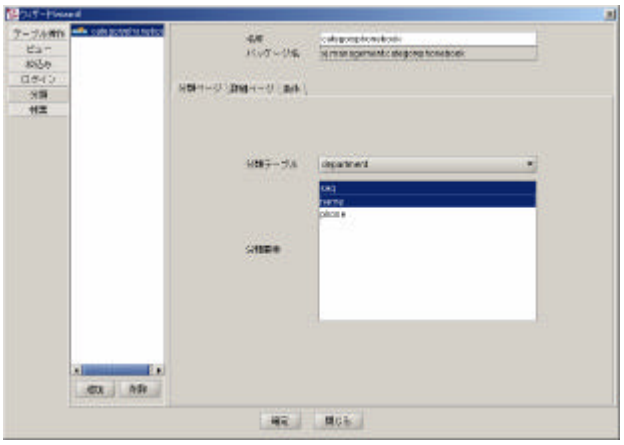
第 7 節　分類ウィザード

従業員の住所録を閲覧するとき、部門の一覧画面からそれぞれの従業員の一覧を表示するような画面を作ります。分類ウィザードを使うとこのような画面を簡単に作成することができます。

第 1 項　分類の定義

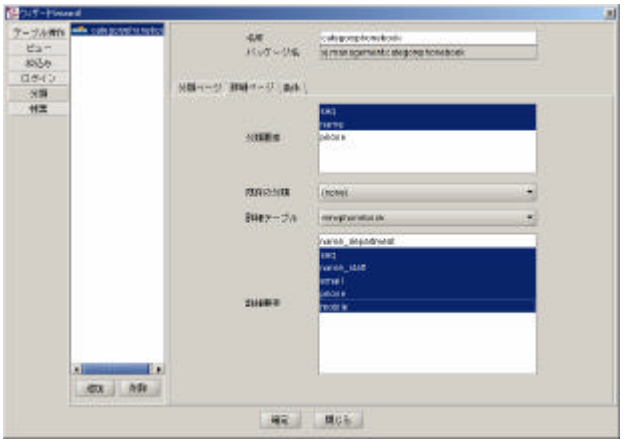
ここでは department テーブルと viewphonebook ビューを使って分類画面を作成します。ウィザードを実行しビジネスオブジェクト選択のダイアログで management を選択し確定ボタンを押し

てください。

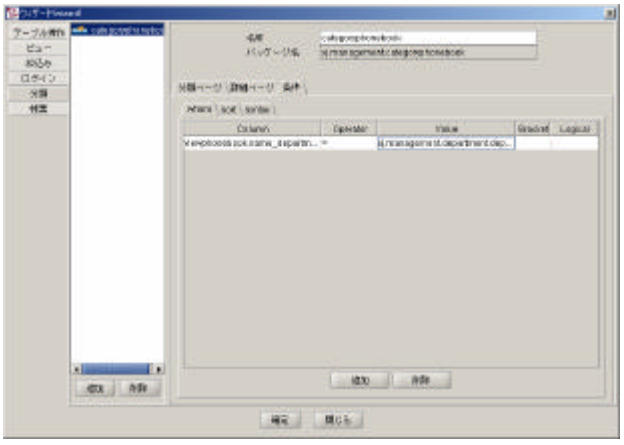


ここでは名称として categoryphonebook を入力します。

分類ページタブで分類テーブルとそのフィールドを選択します。分類テーブルは department を選択し、分類フィールドとして seq と name を選択します。分類フィールドで選択したフィールドが分類ページに表示されます。



詳細ページのタブを選択します。分類要素には seq と name を選択します。ここで選択した要素は詳細画面の上部に表示されます。詳細テーブルには viewphonebook を選択します。詳細要素には seq、name_staff、email、phone と mobile を選択します。



条件タブを選択します。分類テーブルと詳細テーブルの関係の条件を設定します。ここでは viewphonebook.name_department と department.name が等しいものとします。さらにソート

順を設定することもできます。

viewphonebook.name_department =
xj.management.department.departmentForm.departmentRecord.name

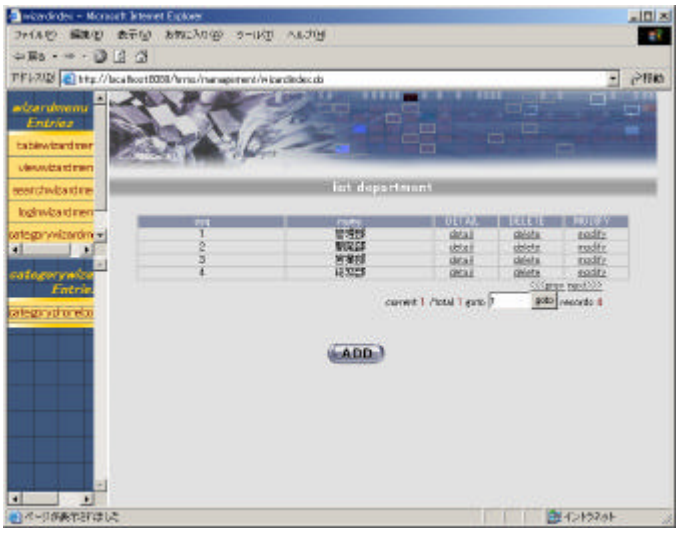
これで分類ウィザードの設定は完了です。確定ボタンを押して関連するファイルを生成してください。

第 2 項 War ファイル

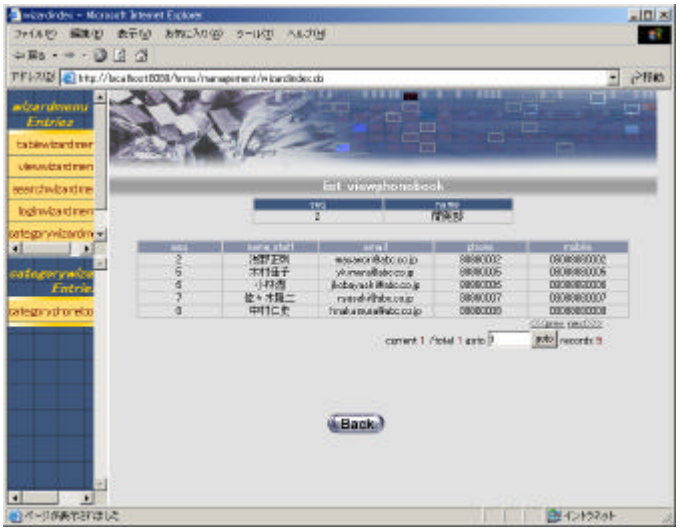
今回は SQL の生成と実行は不要です。前節と同様に War ファイルの生成とインストールを行ってください。

第 3 項 サンプルを実行する

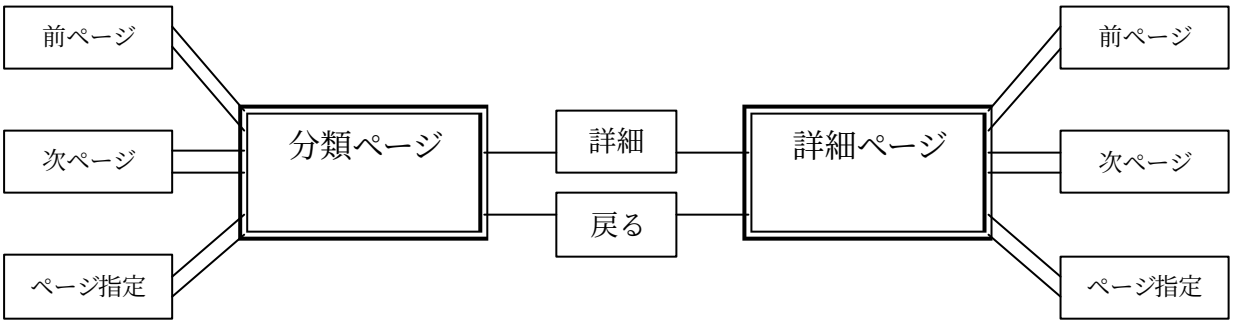
ブラウザを開きのアドレス欄に `http://localhost:8080/hrms/` と入力します。wizardindex 画面で左上の categorywizardmenu をクリックした後 categoryphonebook をクリックすると以下の画面が表示されます。



部署情報の一覧表が表示されます。各行には detail のリンクがあり、これをクリックすると次のように各部署の従業員の住所のページが表示されます。



画面遷移は次のようになっています。



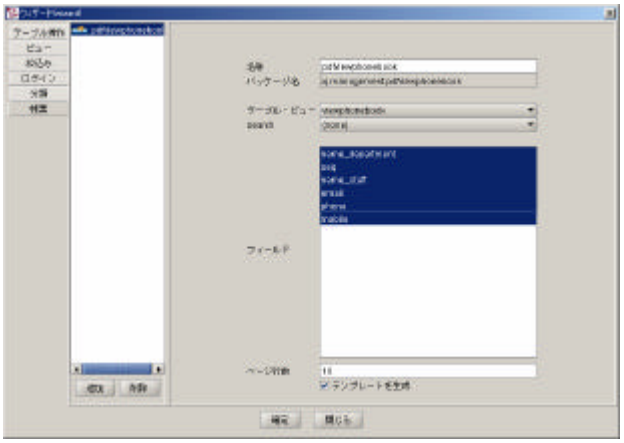
この分類及び詳細テーブルがテーブルウィザードで生成されたテーブルまたはビューであるときは、さらにそのテーブル・ビューの追加、変更および削除の操作もできます。

第 8 節 帳票ウィザード

従業員の住所録を表示する時、同時にこの帳票を出力するとこがあります。帳票ウィザードを使って印刷に適した PDF を出力する機能を簡単に追加することができます。

第 1 項 帳票の定義

ここではviewphonebookビューを使って帳票出力を作成します。ウィザードを実行しビジネスオブジェクト選択のダイアログで management を選択し確定ボタンを押してください。



ここでは名称として pdfviewphonebook、テーブル・ビューに viewphonebook、フィールドに name_department、seq、name_staff、email、phone とmobile を選択します。ページ行数には帳票の 1ページに何行のレコードまでを印字するかを指定します。デフォルト値は 10 です。テンプレート生成をチェックするとデフォルトのテンプレート PDF ファイルを生成します。ユーザがテンプレートを変更した後、再び帳票ウィザードを起動してこのテンプレートを上書きしたくない場合はチェックを外してください。

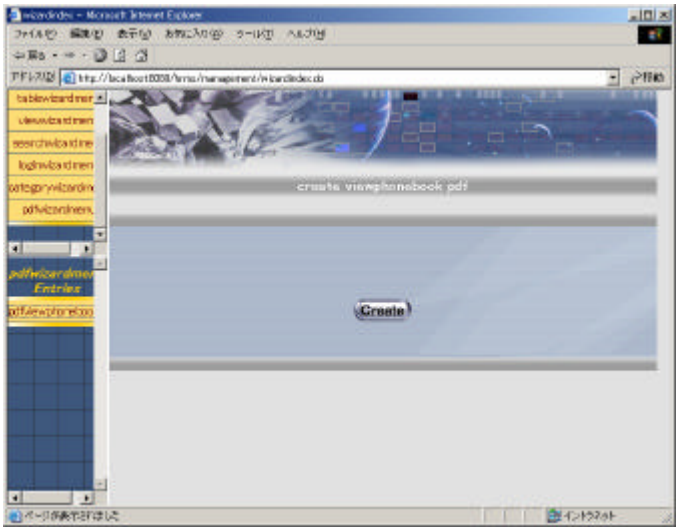
これで帳票ウィザードの設定は完了です。確定ボタンを押して関連するファイルを生成してください。

第 2 項 War ファイル

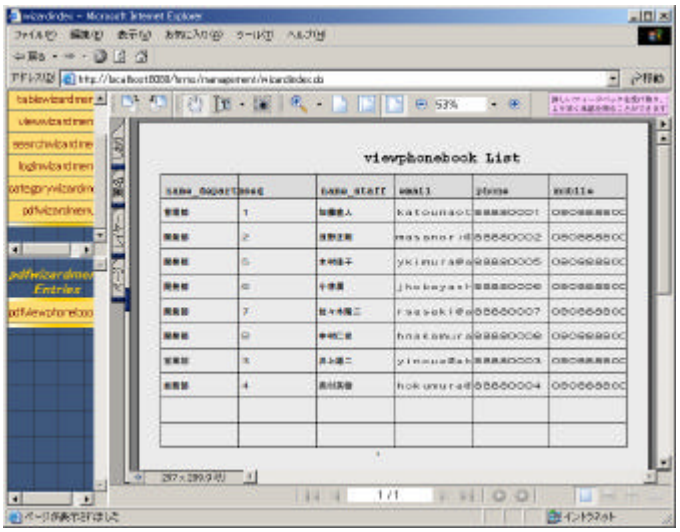
今回は SQL の生成と実行は不要です。前節と同様にWar ファイルの生成とインストールを行ってください。

第 3 項 サンプルを実行する

ブラウザを開きのアドレス欄に `http://localhost:8080/hrms/` と入力します。wizardindex画面で左上の pdfwizardmenu をクリックした後 pdfviewphonebook をクリックすると以下の画面が表示されます。



Create のリンクをクリックすると従業員一覧の PDF が表示されます。



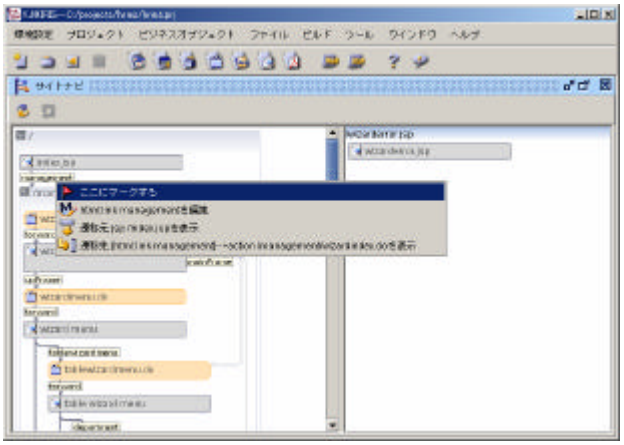
第 9 節 サイト再構築

各種のウィザードで生成された画面（JSP）とアクション(action)の関連を再構成します。次の例ではログイン画面を使ってこの例を説明します。

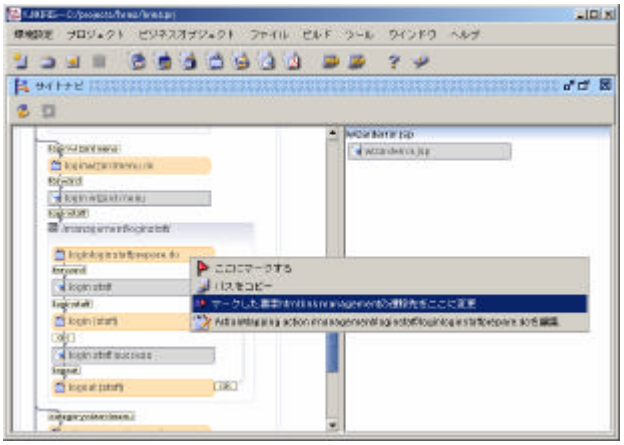
第 1 項 アプリケーションの入り口変更

ウィザードで生成したこの WEB アプリケーションのトップページは index.jsp です。この画面の中の唯一のリンクは loginindex.jsp に遷移するものです。ここではサイトナビを使ってこのリンクの遷移先を loginloginstaff.jsp に変更します。

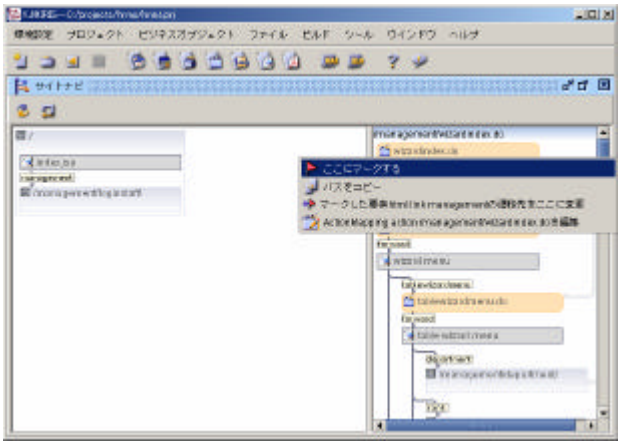
メニューからツール->サイトナビを実行します。



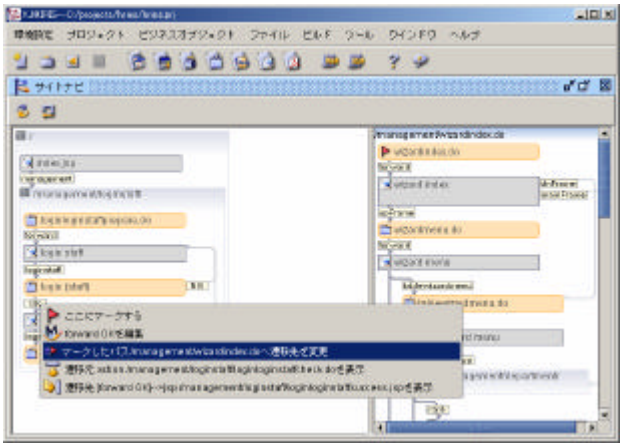
この中で index.jsp の下のリンク management で右クリックしポップアップメニューからここにマークするを選びます。



wizardmenu.jsp の下にある loginwizardmenu.jsp の中の loginloginstaffprepare.do で右クリックしマークした要素の遷移先をここに変更を選びます。



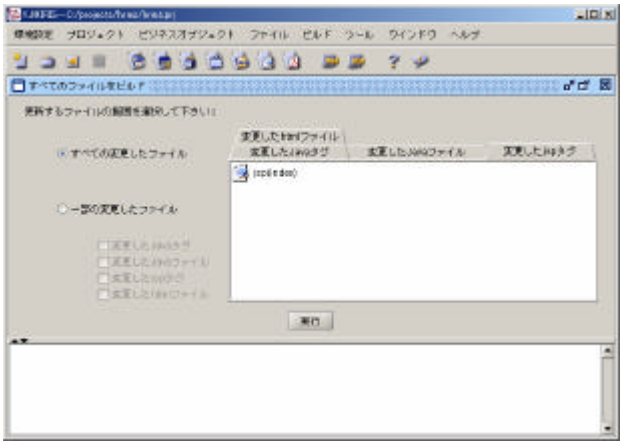
同様に wizardindex.do を選びポップアップメニューからここにマークするを選び、



loginloginstaffcheck.do の下のフォワード OK でポップアップメニューからマークしたパスへ遷移元を変更を選択します。

これでサイトナビでの操作は完了です。フレームを閉じてください。

メニューからビルド>すべてのファイルをビルドを実行します。



実行ボタンを押します。変更されたファイルがビルドされます。

これでサイト再構築の操作は完了です。

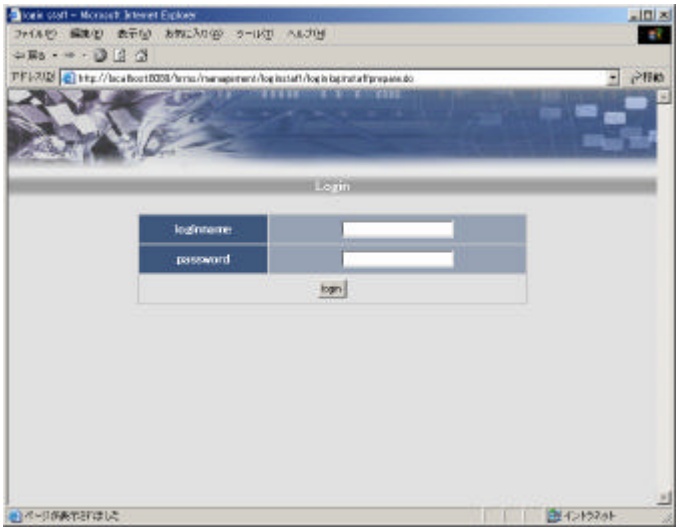
第 2 項 War ファイル

今回はSQL の生成と実行は不要です。前節と同様にWar ファイルの生成とインストールを行ってください。

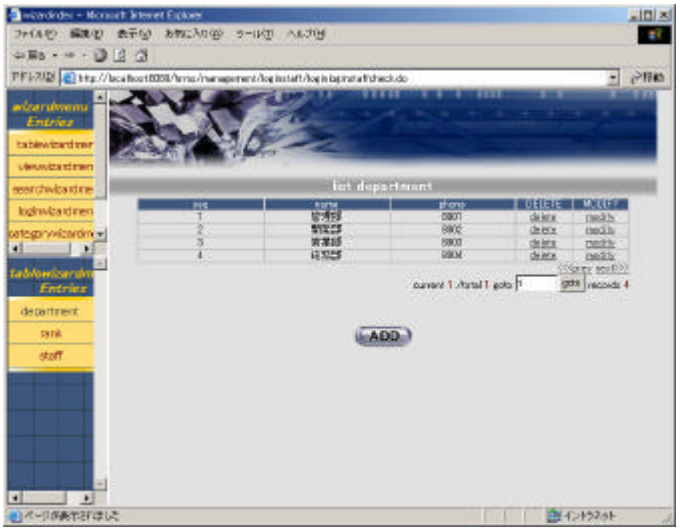
第 3 項 サンプルを実行する

ブラウザを開きのアドレス欄に `http://localhost:8080/hrms/`と入力します。

management のリンクをクリックし、次の画面に移動します。



ユーザ名とパスワードを正しく入力すると次のような画面が表示されます。



第 4 章　モデルファイル詳細

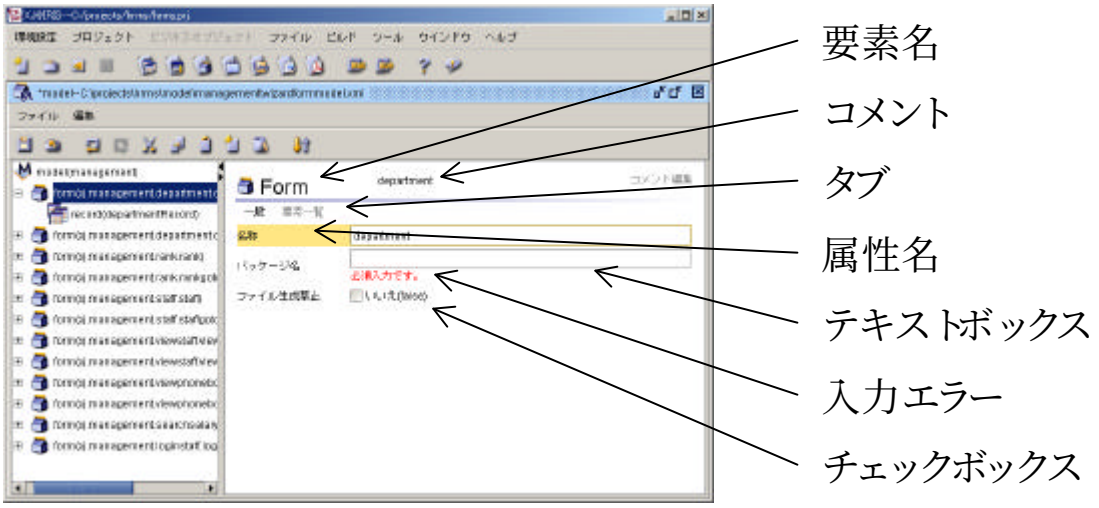
この章ではモデルファイルの編集方法と設定すべき内容について説明します。

第 1 節　XML ファイル編集の基本

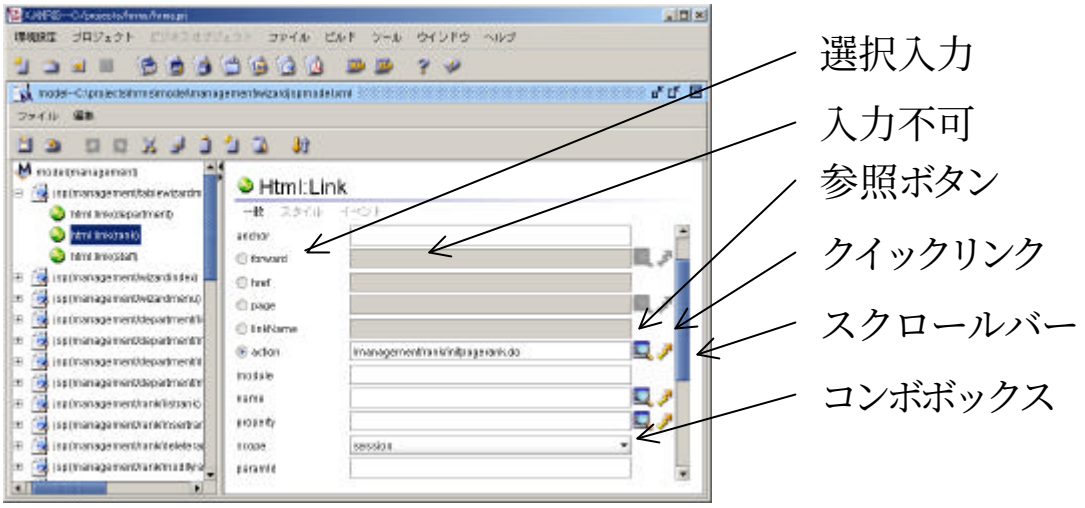
この節ではファイルエディタの編集ペインのうちすべてのモデルファイル、XML　ファイルに共通する使い方について説明します。

第 1 項　画面構成

いくつかの要素の画面を例にして説明します。



要素名	要素名を表示します。XML　ファイルのタグ名に相当します。変更できません。
コメント	いくつかの要素ではコメントをつけることができます。名称属性とは違い文字に制約はありませんので、日本語でわかりやすい説明をつけることができます。内容はこのモデルから生成される Java や JSP ソースファイルに反映されます。これは XML のコメントではありません。
タブ	文字の位置でマウスクリックすることでこれより下の部分の画面が変わります。濃い色の文字が現在選択されているタブです。
属性名	属性の名前です。名前の右側のフィールドで値を設定します。入力フィールドにフォーカスがあるときには回りが黄色になります。
テキストボックス	キーボードから文字列を入力するフィールドです。
入力エラー	このフィールドに対するエラーがあるときに表示されます。メッセージに従い修正してください。
チェックボックス	はい(true)またはいいえ(false)を設定するフィールドです。



選択入力	ラジオボタンがある属性については、グループのうちどれか 1 つを選択して入力します。
入力不可	ラジオボタンで選択されていないなどの理由で入力できない属性のフィールドは無効になります。
参照ボタン	テキストフィールドの入力を支援するためのダイアログを表示します。この属性の候補をプロジェクト内のモデルファイルの定義から表示します。ユーザーはこの中から選択入力できます。
クイックリンク	テキストフィールドに既に入力された値について、この値が定義されている場所があればその位置を表示します。
スクロールバー	属性の数が多く画面に表示しきれないときはスクロールバーが表示されます。
コンボボックス	ユーザはリストの中から選択入力するフィールドです。

第 2 項 model 要素

レコード フォーム、配列モデル、JSP、アクション、PDF モデルと定型文ファイルファイルのルート要素です。ルート要素はファイル生成時に自動的に生成されます。識別子でこのモデルの種類が決まります。識別子は変更できません。名称がありますがあまり意味はありません。通常はファイル名と同じ名称にしておきます。プレビュータブを選択すると、このモデルを保存した場合の XML ファイル全体を見ることができます。

第 2 節 レコードモデル

レコードモデルはデータベースのテーブルとビューの定義に使用します。また、このテーブル、ビューと Java ビーンとのデータ交換を行います。XJ@IRIS はレコードモデルから対応する XxxRecord.java ファイルとテーブル生成・削除の SQL ファイルを生成します。

第 1 項 tablerecord、viewrecord 要素

データベースのテーブルおよびビューに対応するモデル要素です。
この要素からパッケージ名のディレクトリに “名称+Record.java ”の名前の Java ソースファイルが生成されます。すべてのレコードモデル中のすべての tablerecord、viewrecord 要素の名称は 同じ名前になってはいけません。

tablerecord 要素ではこのテーブルの中のすべての comp 要素を定義した後にキーを設定します。キーは update や delete の条件として使われます。キーは複数指定できます。
ファイル生成禁止をチェックすると、Java ファイルの生成を禁止できます。

構成要素タブを選択すると、この table 要素の中の全ての子要素を閲覧することができます。
コメントを入力すると Java ファイル生成時、対応する Java ファイル中のコメントになります。

第 2 項 comp 要素

データベーステーブルの要素（フィールド）に対応するモデル要素です。

名称はテーブルの要素名になり、ビルドで生成される Java ファイルでクラスのフィールド名にもなります。型には JDBC データ型と Java データ型を指定します。それぞれの変換のために get/set メソッドを指定します。これらのフィールドは上にあるフィールドから順に指定します。要素型は normal または sequence を選択します。JDBC データタイプが BIGINT のときにのみ sequence を選択できます。sequence を指定するとデータベースの insert 時にこのフィールドに自動的に一意の値が設定されます。

第 3 項 ref-tablerecord、ref-viewrecord、ref-comp 要素

ref-tablerecord とref-viewrecord は viewrecord 要素の中で別の tablerecord や viewrecord を参照するために使います。さらにその中の comp 要素を参照するために ref-comp 要素を使います。名称には参照先の名称を指定します。もし親の viewrecord 要素内に同じ名前の要素があった場合は別名を指定します。別名があった場合は別名が、ない場合は名称がビルドで生成される Java ファイルでクラスのフィールド名になります。

第 4 項 block、ref-block 要素

要素構造(block)はいくつかの comp の集まりです。block 要素は要素構造の定義、ref-block 要素は要素構造の参照です。tablerecord 要素の子に ref-block 要素を置くことで、**block 要素で定義した要素構造を参照することができます**。また block 要素の子に ref-block を置くこともできます。

レコードモデルがその他の定義のファイルから参照される時、 block 要素は対応する ref-block 要素の comp 要素に展開されます。

第 3 節 配列モデル

配列(iterate)モデルは Struts の logic:iterate 要素に対応する java ファイルを生成するために使用します。XJ@IRIS は配列(iterate)モデルから対応する XxxIterate.java ファイルを生成します。

第 1 項 iterate 要素

1 つの配列ビーンを定義します。

この要素から、パッケージ名のディレクトリに “名称+Iterate.java ”の名前の Java ソースファイルが生成されます。

ファイル生成禁止をチェックすると、Java ファイルの生成を禁止できます。

構成要素タブを選択すると、この iterate 要素の中の全ての子要素を閲覧することができます。

1 つの iterate 要素の子要素に重複する名前があってはいけません。コメントを入力すると Java ファイル生成時、対応する java ファイル中のコメントになります。

第 2 項 record 要素

record 要素はレコードモデルで定義された tablerecord または viewrecord を参照する変数を宣言します。

第 3 項 comp 要素

1 つの変数を宣言します。指定した Java データ型でビーンのフィールドになります。

第 4 節 フォームモデル

フォームモデルは Struts の html:form に対応するユーザがブラウザで入力したデータを扱う Java ビーンの生成のために使用します。XJ@IRIS はフォームモデルから XxxForm.java ファイルを生成します。

第 1 項 form 要素

1 つの form ビーンを定義します。

この要素から、パッケージ名のディレクトリに “名称+Form.java ”の名前の Java ソースファイルが生成されます。ファイル生成禁止をチェックすると、Java ファイルの生成を禁止できます。構成要素タブを選択すると、この form 要素の中の全ての子要素を閲覧することができます。1

つの form 要素の子要素に重複する名前があってはいけません。コメントを入力すると Java ファイル生成時、対応する Java ファイル中のコメントになります。

第 2 項 record、iterate 要素

record 要素はレコードモデルで定義された tablerecord または viewrecord を参照する変数を宣言します。

iterate 要素は配列モデルで定義された iterate を参照する変数を宣言します。

それぞれ参照先のモデルで定義された型でビーンのフィールドになります。

第 3 項 comp 要素

1 つの変数を宣言します。指定した Java データ型でビーンのフィールドになります。

第 5 節 JSP モデル

JSP モデルでは Web アプリケーションの JSP（ページ）の中で動的に変わる部分やフォームなどユーザが入力する部分などを定義します。JSP モデルではページの見かけ、飾り付けについては定義しません。これは HTML ファイルを編集します。

この節では JSP モデル中の主要な要素についてのみ説明し、その他の要素については用途についてのみ紹介します。ここで説明されていない要素や属性の詳しい使い方については対応する Struts のドキュメントを参照してください。

第 1 項 model 要素

jsp 状態タブを選択すると、このモデルファイル中の JSP と対応する HTML ファイルのタグを検査し過不足があった場合に一覧表示します。

第 2 項 jsp 要素

1つの JSP（ページ）に対応するモデル要素です。
この要素に対応する HTML ファイルはパスのディレクトリに “名称+.html ”のファイル名になります。JSP ファイルは拡張子を jsp にしたものになります。

ファイル生成禁止をチェックすると、JSP ファイルの生成を禁止できます。

HTML ファイル合成ボタンを押すと、この JSP に対応する HTML ファイルを編集できます。編集を行う前に、現在の JSP モデルを保存して下さい。

第 3 項 form 要素

Struts の html:form タグに対応します。ブラウザには form タグとして出力されます。ユーザが入力するフィールドをまとめる役割があります。通常はフォームタグの子にhtml:text 要素などの入力要素とhtml:submit などのボタン要素を置きます。

名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。
name にはこの フォームに対応するフォームビーンの保存名を指定します。type にはフォームビーンの型を指定します。action にはこの フォームの子のボタン要素が押されたときに実行されるアクションのパスを指定します。

html:javascript を生成が設定された場合は、対応するform の前に 1つの html:javascript 要素を追加することができます。 クライアントでの入力チェックを行う場合に便利です。

第 4 項 iterate 要素

Struts の logic:iterate タグに対応します。この要素の子は複数回繰り返してページに出力されます。

名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。
iterate 要素は form 要素の子になることができます。form 要素の子になると入力要素を繰り返すことができます。form要素の子の場合にはid フィールドはproperty フィールドと同じでなければなりません。

第 5 項 html:link 要素

Struts の html:link タグに対応します。ブラウザには HTML の a タグ（アンカー）として出力されます。名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。

このリンクをクリックされたときの遷移先を次のどれか一つの方法で指定します。ただし linkName 属性を指定した場合は遷移先ではなく出力される a タグの name 属性になります。

forward	struts-config で定義された global-forward を参照します。
href	指定した値がそのまま a タグに出力されます。
page	JSP ページを指定します。
linkName	name 属性として出力されます。
action	アクションのパスを指定します。

第 6 項 その他のリンク系要素

html:link と同じようにパスを指定する要素には以下のようなものがあります。

html:frame	HTML の frame タグとして出力されます。
html:img	HTML の img タグとして出力されます。
html:link	HTML の a タグとして出力されます。

html:rewrite	指定されたパスのみが出力されます。
logic:redirect	HttpServletResponse.sendRedirect() が実行されます。
logic:forward	global-forward の名前をを参照して遷移します。

第 7 項 html:text

Struts の html:text タグに対応します。ブラウザには HTML の input タグ（type=text）として出力されます。名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。

name 属性にはこの入力フィールドで使用するフォームビーンの保存名を指定します。省略した場合は親の form 要素で指定された値になります。property 属性にはそのフォームビーンの中のフィールド名を指定します。

validation の constant 定義、validation の field 定義タブはこの入力フィールドの validation を指定します。詳しくは validation の説明を参照してください。

第 8 項 その他の入力系要素

html:text と同じように入力フィールドを指定する要素とその関連要素には以下のようなものがあります。

html:checkbox	HTML の input タグ type=checkbox として出力されます。
html:multibox	HTML の input タグ type=checkbox として出力されます。
html:radio	HTML の input タグ type=radio として出力されます。
html:text	HTML の input タグ type=text として出力されます。
html:textarea	HTML の textarea タグとして出力されます。
html:hidden	HTML の input タグ type=hidden として出力されます。
html:option	1つの HTML の option タグとして出力されます。
html:options	複数の HTML の option タグとして出力されます。
html:optionsCollection	複数の HTML の option タグとして出力されます。 html:options をより使いやすくしたものです。
html:password	HTML の input タグ type=password として出力されます。
html:select	HTML の select タグとして出力されます。
html:file	HTML の input タグ type=file として出力されます。
テキスト	そのまま文字列として出力されます。html:option の子要素として使用します。

第 9 項 html:submit

Struts の html:submit タグに対応します。ブラウザには HTML の input タグ（type=submit）として出力されます。名称には適当な名称を設定してください。アプリケーションの動作には関係

ありません。

property 属性を指定すると、フォームを送信するときにその変数に value 属性で指定した値が代入されて送信されます。

第 10 項 その他のボタン系要素

html:submit と同じようにフォームのボタンとして機能する関連要素には以下のようなものがあります。

html:cancel	HTML の input タグ type=submit として出力されます。Struts の Action.isCancelled()が使えるようにパラメータが設定されます。
html:reset	HTML の input タグ type=reset として出力されます。
html:button	HTML の input タグ type=button として出力されます。
html:submit	HTML の input タグ type=submit として出力されます。
html:image	HTML の input タグ type=image として出力されます。

第 11 項 bean:write

Struts の bean:write タグに対応します。保存したビーンから文字列を出力します。名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。

name 属性には使用するビーンの保存名を指定します。省略した場合でこの要素の親に form 要素があると form 要素で指定されたビーンになり、iterate 要素があるとその id を name とすることができます。property 属性にはそのビーンの中のフィールド名を指定します。

第 12 項 html:errors

Struts の html:errors タグに対応します。この位置にアクション ActionErrors として設定されたエラーメッセージを出力します。名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。

第 13 項 その他の文字列出力系要素

html:errors と同じように文字列を出力する要素には以下のようなものがあります。

html:errors	アクションで ActionErrors として設定されたメッセージが出力されます。
html:messages	アクションで ActionMessages として設定されたメッセージが出

	力されます。
bean:message	アプリケーションリソースや保存したビーンから作成した文字列が出力されます。
bean:write	保存したビーンから作成した文字列が出力されます。

第 14 項 logic:equal

Struts の logic:equal タグに対応します。この要素の条件を満たした場合のみこの要素の子要素や文字列を出力します。名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。

cookie、header、parameter、name 属性から 1 つを入力します。name に保存したビーン名を指定した場合にはそのフィールドを property 属性に指定することができます。動作時にはその値と value 属性で指定した定数を比較し、等しい場合にのみ子要素を出力します。

第 15 項 その他の条件評価系要素

logic:equal と同じように条件を判定し、子要素の出力を制御する要素には以下のようなものがあります。

logic:empty	指定された変数が空であるか判定します。
logic:notEmpty	指定された変数が空であるか判定します。
logic:present	指定された変数が存在するか判定します。
logic:notPresent	指定された変数が存在するか判定します。
logic:equal	指定された変数が"="定数であるか判定します。
logic:notEqual	指定された変数が"!="定数であるか判定します。
logic:greaterEqual	指定された変数が">="定数であるか判定します。
logic:greaterThan	指定された変数が">"定数であるか判定します。
logic:lessEqual	指定された変数が"<="定数であるか判定します。
logic:lessThan	指定された変数が"<"定数であるか判定します。
logic:match	指定された変数に定数が含まれるか判定します。
logic:notMatch	指定された変数に定数が含まれるか判定します。
logic: messagesPresent	指定されたメッセージが存在するか判定します。
logic: messagesNotPresent	指定されたメッセージが存在するか判定します。

第 6 節 アクションモデル

アクションモデルでは Web アプリケーションの動作を定義します。このファイルからアクションクラスが生成されます。

第 1 項 model 要素

アクション一覧タブを選択すると、すべてのアクション名称とタイプを 閲覧することができます。

第 2 項 action 要素

1 つのアクションクラスを定義します。

この要素から、パッケージ名のディレクトリに “名称+Action.java ”の名前の Java ソースファイルが生成されます。定型文選択で定型文ファイルを指定します。Java ファイル生成の時にはこの定型文の head、init、tail 部分追加されます。ファイル生成禁止をチェックすると、Java ファイルの生成を禁止できます。定型文生成以後 Java ファイルを変更した場合には、生成禁止にしておくと間違った上書きを防ぐことができます。コメントを入力すると Java ファイル生成時、対応する Java ファイル中のコメントになります。プレビュータブを選択してこの action 要素から生成される Java ファイルを閲覧することができます。

第 3 項 unit 要素

アクション内の 1 つの一つの機能を表すモデル要素です。
型によってアクションテンプレートから自動生成することができます。引数自動生成ボタンを押すと、この unit 要素に必要なデフォルトの param 子要素が自動生成されます。引数一覧タブを選択すると、 この unit 要素下のすべての param 要素を閲覧することができます。コメントを入力すると、対応する Java ソース部分のコメントになります。

定型文タブを選択すると、指定した型のアクションテンプレート中の ソースファイルを閲覧することができます。プレビュータブを選択すると、指定した param 子要素を反映した Java ソースファイルを閲覧することが出来ます。

第 4 項 param 要素

unit 要素の型で選択したアクションテンプレートへの引数を指定します。param 要素は 2 つの場所で使うことができます。1 つは unit 要素の子要素として使います。このときは、親の unit に対する引数になります。もう1 つは model 要素の子要素として使います。この引数は 1 つのアクションモデルファイル内のすべての unit 要素の引数となることができます。もし unit 要素の子要素の param とグローバルの param で同じ引数名が定義されていた場合は unit の子要素が

優先されます。定型文タブを選択すると、親の unit 要素の型に対応するアクションテンプレートファイルを閲覧することができます。

パラメータの命名規則と選択規則を示します。
パラメータはの構造は X-Yyy または X-Yyy-Zzz となっています。
X で引数の変換方式を決定し、Yyy で選択する一覧を指定します。

文字	可能な選択	意味	変換後の文字列
X	C	Package 付きのフルクラス名	Package.+U(name)+Yyy
	B	Bean メソッド名の get/set 以後の部分	name+ Yyy
	V	comp(変数) の Bean メッソド名の get/set 以後の部分	name
	S	Session 中の属性名	Package_+ Yyy
	N	変換を行わない	name
	P	定数	name
	Q	共通引数	

U(name)は name の先頭を大文字にしたものを表します。

文字	可能な選択	意味
Yyy	Table	recordmodel.xml 中で定義したデータベースのテーブル/ビュー名の中から選択します。
	Record	recordmodel.xml の record の名称の一覧から選択、または、record の comp の名称の一覧から選択します。
	Iterate	iteratemodel.xml の iterate の名称の一覧から選択、または、iterate の comp の名称の一覧から選択します。
	Form	formmodel.xml の form の名称の一覧から選択、または、form の comp の名称の一覧から選択します。
	Action	actionmodel.xml 中で定義したアクション名の中から選択します。

第 7 節 PDF モデル

PDF モデルでは PDF 帳票出力に必要な情報の取得方法と PDF テンプレートファイルを指定します。大きく分けて、情報の取得方法を指定する pdfdatasource と、帳票の定義である pdffill があります。

第 1 項 pdfdatasource 要素

情報の取得方法を指定します。いくつかの table 要素、unit 要素のまとまりをあらわします。ここで指定した名称によって pdffill から参照します。

第 2 項 table、column 要素

DB テーブルからの 情報取得を定義します。
テーブル ・ビュー参照ボタンか ら、レコードモデルで定義されたテーブル またはビューを選択します。テーブル・ビュー指定後にコラムを自動選択ボタンを押すと column 子要素が自動生成されます。DB からの データ取得の条件を指定します。

table データソースの値は 1つの 2次元配列であり、その第 1行目は PDF ファイルでは column の名称になります。

第 3 項 unit 要素

unit 要素は 1 つの文字列の値を定義します。

unit 要素の値の文字列の中に次の特殊なキーワードを含むことができます。これらは PDF 生成時に次の通り展開されます。

"%PAGENUM%"――現在のページ番号。
"%DATE%"――現在の日付、フォーマットは “yyyy-MM-DD ”です。
"%YEAR%"――年、フォーマットは “yyyy ”です。
"%MONTH%"――月、フォーマットは “MM ”です。
"%DAY%"――日、フォーマットは “DD ”です。

例えば unit 要素の値として “page %PAGENUM% ”と指定し, このデータソースを pdffill ノード中で指定した acrofield には、第 5 ページの出力は"page 5"となります。

第 4 項 pdffill 要素

pdffill 要素は 1種類の PDF ファイルに相当します。

この要素には対応する PDF テンプレートが必ず一つ必要です。この PDF テンプレートファイルの acrofield に値を埋め込むことによって PDF ファイルを出力します。

属性の説明：

名称 :pdffill 要素の名称。対応する pdf ファイル名は"名称.pdf ”になります

パス :XJ@IRIS から PDF ファイルを生成する際にプロジェクトディレクトリの pdf の下にこのパスをつけてファイルを出力します。

テンプレート:PDF テンプレートファイルをプロジェクトディレクトリの pdfTemplates からの相対パスで指定します。もし、pdfTempletes 以下以外にファイルがある場合はインポートを使ってファイルをコピーする必要があります。

複数ページ :複数ページを許可します。データソースの件数が PDF テンプレートの行数より多い場合は出力ファイルが複数ページになります。

最大ページ数 :最大のページ数を指定します。0 の時は無制限を意味します。

複数ページとなるデータ :pdffill 要素下の 1つの tablefill 名称を選択します。複数ページはこのデータを基準とします。

acrofield を自動生成ボタンを押すと、PDF テンプレートファイルで使われているフィールドを読み込んで acrofield 子要素を自動的に生成します。

第 5 項 acrofield 要素

PDF テンプレートファイル中のフィールドに対応します。tablefill に含まれるフィールドは表示されません。

名称：PDF テンプレートの textfield 名称です。

datasource：この acrofield に出力する unit データソースを指定します。キーフィールドは“pdfdatasource 名称.unit 名称”のように 2 部分から構成されます。

型：text とpicture の 2 種類から選択します。text の場合、この acrofield は純粋なテキストで、unit データソースの値を出力します。picture の場合、この acrofield は絵・写真で、unit データソースの値は絵・写真のパス（ファイルの絶対的なパスあるいは url）を指定します。acrofield の大きさと位置が絵・写真の大きさと位置になります。

第 6 項 tablefill 要素

tablefill 要素は PDF ファイル中の表に対応します。table データソースの情報を PDF テンプレートの textfield に出力することを指定します。

名称には適当な名称を設定してください。アプリケーションの動作には関係ありません。datasource の参照ボタンを押して table データソースを選択します。データソースは“pdfdatasource 名称.table 名称”のように 2つの部分から構成されます。
右枠のタブはこのテーブル中の column に対応します。左側の acrofield 一覧からそれぞれの column に追加します。自動マッチを押すとacrofield の名称が"column 名称+#+数字"の規則に合う場合に自動的に配置します。

tablefill 要素で column の変更を行った後は acrofield を自動更新ボタン押して pdffill 要素の中の acrofield を整理してください。

第 8 節 定型文ファイル

XJ@IRIS は定型文ファイルによってソースファイルなどを生成します。

定型文ファイルはたとえばアクションテンプレートや、メール定型文に使用することができます。

プロジェクトを新規作成したときに、actiontemplete.xml という名前のデフォルトのアクションテンプレートファイルが生成されます。

第 1 項 unit 要素

定型文の生成単位です。この中に固定の文字列（テキスト要素）と param 要素を置きます。引数・属性タブを選択するとこの要素で使われている param と attri 要素の一覧を見ることがで

きます。プレビュータブを選択すると文字列を含めたこの要素全体を表示します。共通引数を宣言するとその引数はこの unit 要素内のいくつかの param 要素でデフォルト値として使うことができます。共通引数を複数宣言する場合はカンマ(,)で区切ってください。

第 2 項 param 要素

この要素の部分に与えられた引数の値を展開します。共通引数を選択した場合はその値がデフォルト値として使われます。参照引数は引数を設定する場合にユーザが参照ボタンにより入力できるようにするために設定します。デフォルト値は値と共通引数が定義されていない場合に採用されます。

第 3 項 attri 要素

この要素の部分に与えられた引数に名前と値を展開します。
例えば：名称を name，値を value として置換すると、次のような文字列になります。

```
name = "value"
```

第 9 節 Web-app 定義ファイル

Servlet のデプロイメント記述子（deployment descriptor）を定義します。
XJ@IRIS は Servlet で定義された要素のうちで通常使用される要素のみ編集または参照できます。特殊な用途でその他の要素をしなければならない場合は別の編集ツールを使用してください。ここでは XJ@IRIS で編集可能な要素のうち重要なもののみについて説明します。詳しくは Servlet または Tomcat のドキュメントを参照してください。

第 1 項 web-app 要素

このファイルのルート要素です。プレビュータブでこのモデルを保存した場合の XML ファイル全体を見ることができます。

第 2 項 welcome-file-list、welcome-file 要素

ブラウザからファイル名を指定せずディレクトリだけのリクエストがきた場合に表示するデフォルトのファイル名を指定します。welcome-file-list 要素の下には複数の welcome-file を指定することができます。

第 10 節 Struts 定義ファイル

Struts の設定ファイル（struts-config）です。

XJ@IRIS は Struts で定義された要素のうちで通常使用される要素のみ編集または参照できます。特殊な用途でその他の要素をしなければならない場合は別の編集ツールを使用してください。ここでは XJ@IRIS で編集可能な要素のうち重要なもののみについて説明します。ここで説明していない要素、属性については Struts のドキュメントを参照してください。

第 1 項 struts-config 要素

このファイルのルート要素です。プレビュータブでこのモデルを保存した場合の XML ファイル全体を見ることができます。

第 2 項 data-sources、form-beans 要素

これらの要素は War 生成を実行するとモデルファイルから自動的に更新されます。ここでは閲覧のみ可能です。

第 3 項 global-exceptions、exception 要素

アクションクラスで throw され個別の action でハンドルされない例外の動作を指定します。

第 4 項 global-forwards、forward 要素

アクションクラス内で参照され、個別の action で定義されていない forward の遷移先を指定します。

第 5 項 action-mappings、action 要素

action 要素が action-mappings の子要素になります。action 要素ではアクションの URL パスと動作するアクションクラス、引数となるフォームビーンを関連づけます。
path にはこのアクションの URL パスを指定します。拡張子を付ける必要はありません。実際には web-app の設定により.do を補った URL が指定されたときに呼び出されます。
name には送信されたフォーム入力を代入するフォームビーンを指定します。フォーム送信後のアクションでない場合は指定する必要はありません。
input にはこのアクションの前の JSP 画面のパスを指定します。フォームで validation を指定しエラーがあったときにはこのパスへ戻ります。forward を指定した場合にはアクションクラスを実行せず指定したパスに移動します。type には実行するアクションクラスを指定します。

第 6 項 forward 要素

アクションクラス中で呼び出された forward 名に対応するパスを指定します。

name には forward 名を指定してください。path には遷移する URL パスを指定します。

第 7 項 exception 要素

アクションクラス中で throw された例外のハンドラを指定します。
handler には例外ハンドラクラス名を指定します。
path には遷移する URL パスを指定します。
type には catch する例外のクラス名を指定します。

第 11 節 アクセス定義ファイル

アクセス定義ファイルはログインしていないユーザセッションでのアプリケーションへのアクセス防ぐなどの目的で使用します。このファイルは web-app で指定されている AccessFilter で使用されます。

第 1 項 access 要素

このファイルのルート要素です。このモデルを保存した場合の XML ファイル全体を見ることができます。

第 2 項 zone-map 要素

ここでユーザリクエストURL と対応するゾーンを関連付けます。zone-map 要素はアクセスファイルに 1 つだけ存在します。

ゾーンとは同じ権限でアクセスできる JSP やアクションの集まりです。ここでリクエストURL からゾーンを決定して zone 要素でそのゾーンに対する権限のチェックをします。ゾーンマップでは表の 1 行を選択した状態でボタンを押して行の追加や削除、順序の入れ替えを行います。URL パターンにはユーザリクエストのパターンを入力します。パターンにはアスタリスク(*) とクエスチョン(?)のワイルドカードを含めることができます。ゾーンマップは上から順に検索されます。パターンにマッチした場合にはその行のゾーン名がリクエストURL のゾーンになります。どの行にもマッチしなかった場合は最下行の default のゾーンになります。default の行は URL パターンの入力、移動、削除はできません。

第 3 項 zone 要素

zone 要素では各ゾーンに対する権限チェックの方法を指定します。

名称にはこの zone 要素の名称を入力してください。zone-map 要素によりリクエストURL からzone の名称が決定され対応する zone が参照されます。zone-map と同様に、表のうち 1 行を選択して行の操作を行います。name にはセッションビーンの名前を入力します。参照ボタンにより選択入力ができます。property にはビーンのフィールドを指定します。ドット(.)で区切って

さらに子のフィールドを指定することもできます。比較には比較演算子を選択します。定数には文字列を入力します。左辺の変数の型に合った文字列を入力してください。たとえば型がboolean の場合は"true" または"false"と入力します。forward にはこの行の式が成り立った場合に遷移する global-forwards の名称を入力します。forward に何も入力しない場合はグレーでthrough と表示されリクエストURL がそのまま実行されることを意味します。式は表の上から順番に評価され、成り立った行の forward に決定されます。どの行の式も成り立たなかった場合には最下行の default の forward に決定されます。default の行には式は入力できません。forward のみ変更できます。default の行は移動、削除できません。

第 12 節 アプリケーションリソース

アプリケーションのリソースバンドルを編集します。

このファイルは XML ファイルではないため他のファイルと編集方法が異なります。各メッセージのキーはドット(.)で区切った文字列です。画面では区切りごとにツリー構造で表示しています。左側のツリーペインから1 つのノードを選択し、右側の編集ペインの表で値を変更します。

第 1 項 キーの追加

ツリーペインからノードを選択しマウスの右クリックでポップアップメニューを開きます。追加を選択すると現在選択位置の子として新しいキーが作成されます。キーにドット(.)を含めるともで来ます。

第 2 項 キーの変更

キーのノードを選択しマウスの右クリックでポップアップメニューを開きます。変更を選択するとキーを変更することができます。

第 3 項 キーの削除

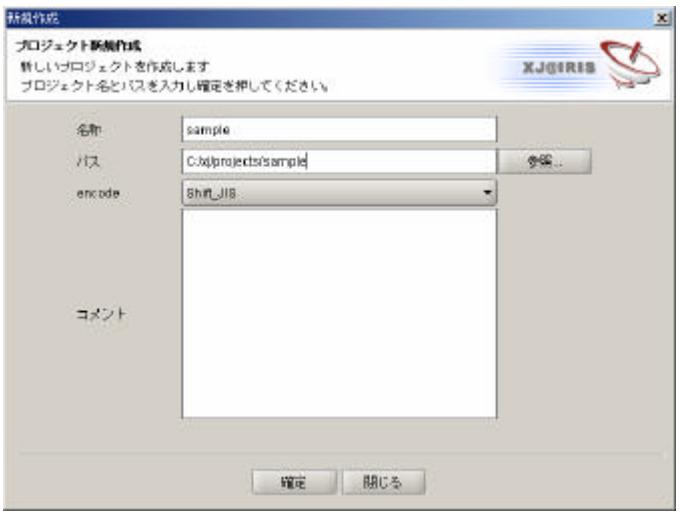
子のないキーのノードを選択しマウスの右クリックでポップアップメニューを開きます。削除を選択するとキーを削除することができます。

第 5 章　基本機能詳細

第 1 節　プロジェクト

第 1 項　プロジェクト新規作成

メニューからプロジェクト新規作成を選択します。



名称とパスを入力し、確定ボタンを押してください。プロジェクトが作成されその プロジェクトがオープン状態になります。メインウインドウのタイトルにプロジェクトのパスが表示されます。

第 2 項　プロジェクトオープン

既存のプロジェクトを開く方法はいくつかあります。メニューからプロジェクト>オープンを選択してダイアログからプロジェクトファイル(*.prj)を選択するか、メニューから最近使ったプロジェクトでプロジェクトパスを指定します。

第 3 項　プロジェクトクローズ

メニューからプロジェクト>クローズを選択します。開いていたファイルの情報がプロジェクトに保存されます。

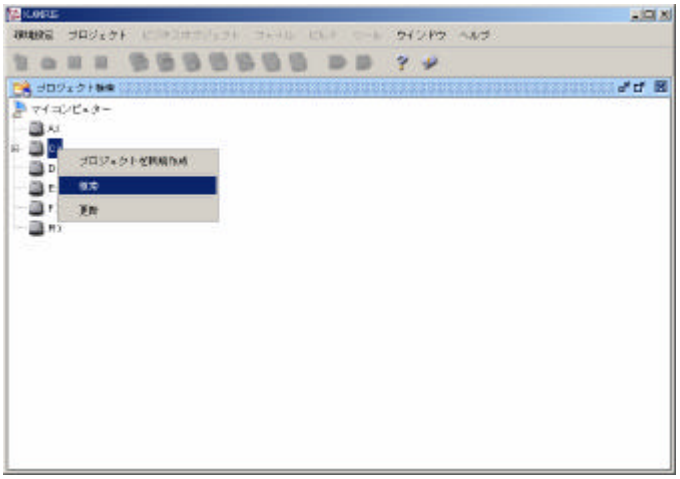
第 4 項　プロジェクト属性

メニューからプロジェクト属性を選択します。ダイアログで現在のプロジェクトの コメントを入力

します。

第 5 項 プロジェクト検索

メニューからプロジェクト検索を選択すると以下のようなフレームが開きます。



このコンピュータの ドライブが表示されます。ドライブのアイコンをダブルクリックするとその下のディレクトリが表示されます。ディレクトリで右 クリックをするとポップアップメニューが表示され以下の操作ができます。

プロジェクトの新規作成 :この位置に新しいプロジェクトを作成します。

検索 :このディレクトリ以下のプロジェクトファイル(*.prj)を検索します。ファイル数の多いディレクトリや ドライブで検索を行うと時間がかかります。

更新 :このディレクトリ以下の情報を最新状態にします。

プロジェクトファイル(.prj)をダブルクリックするとプロジェクトのモデルファイルが表示されます。モデルファイルをダブルクリックするとモデル内の要素が表示されます。

プロジェクトファイルを選択し右クリックするとポップアップメニューが表示され以下の操作ができます。

プロジェクトをオープン :選択プロジェクトを開きます。

開いたプロジェクトをクローズ :現在のプロジェクトをクローズします。

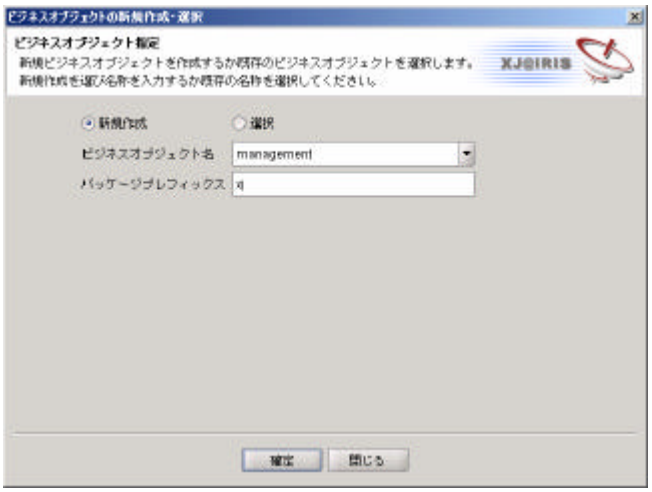
更新 :表示されたモデルファイルを最新の状態にします。

第２節　ウィザード

第１項　テーブルウィザード

テーブルウィザードで複数のテーブル、および他のテーブルと外部リンクするビューを作成しその一覧表示、追加、修正、削除画面を持った Web アプリケーションを自動生成します。

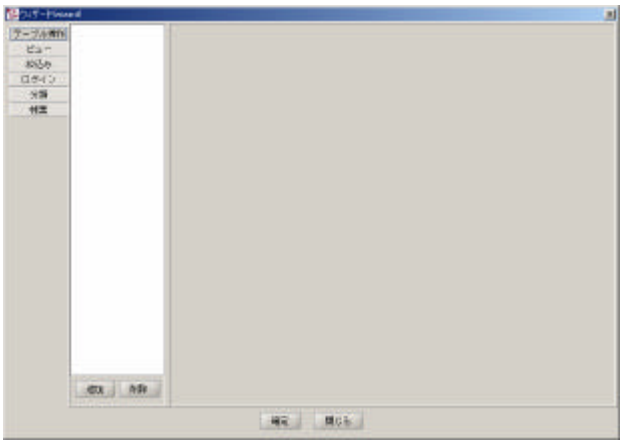
メニューからプロジェクト>ウィザードを選択します。ビジネスオブジェクトの新規作成・選択のダイアログが開きます。



一回目は新規作成ラジオボタンを選択し、ビジネスオブジェクト名とパッケージプレフィックスに適当な名称を設定し、確定ボタンを押して、次の画面に入ります。

二回目以後は既存のビジネスオブジェクトに編集したい場合は、選択ラジオボタンを選択し、既存のビジネスオブジェクト一覧で編集したいものを選らんで、確定ボタンを押して、次の画面に入ります。

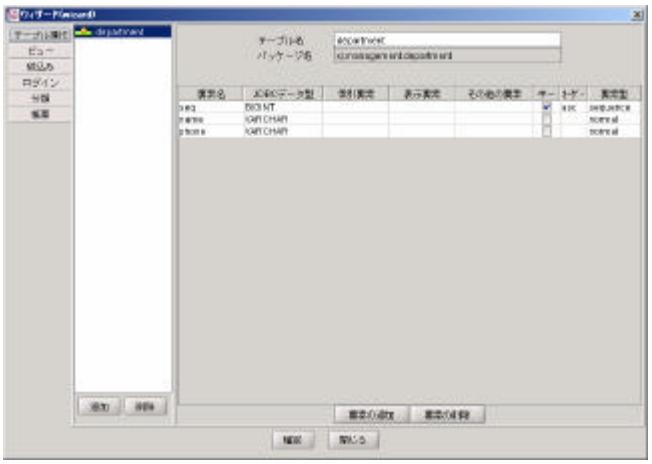
ウィザードの定義ダイアログが開きます。



この画面の一番左にはウィザード選択のタブがあります。ダイアログが開いたときにはテーブル操作になっています。その右側にはテーブル一覧があります。初期には何も表示されていません。下方の追加、削除ボタンで操作します。その右側にはテーブル一覧から選択したテーブ

ルの内容が表示されます。初期にはテーブルがないので表示されていません。

追加ボタンを**押すと**テーブルが追加されます。



テーブル名はデータベースのテーブル名です。

フィールドには要素名、JDBC データ型、索引要素、表示要素、その他の要素、キー、オーダーと要素型など属性があります。

要素名と JDBC データ型は必須属性です。

外部リンクを定義する時に、索引要素に同じ JDBC データ型の html:option の value に対応する要素を、表示要素に html:option の表示テキストに対応する要素を、その他の要素にビューに追加したい要素をそれぞれ設定します。

要素型は sequence, normal, create date, update date 四種類があります。

JDBC データ型が BIGINT の場合は、要素型に sequence を指定できます。このフィールドの値の設定は要りません、XJ@IRIS ではデータベースに挿入するときに、自動的シーケンス値を取得し、設定します。

JDBC データ型が DATE と TIMESTAMP の場合は、要素型に create date と update date を指定できます。このフィールドの値の設定は要りません、XJ@IRIS ではデータベースに挿入・更新するときに、自動的日付と時刻を取得し、設定します。

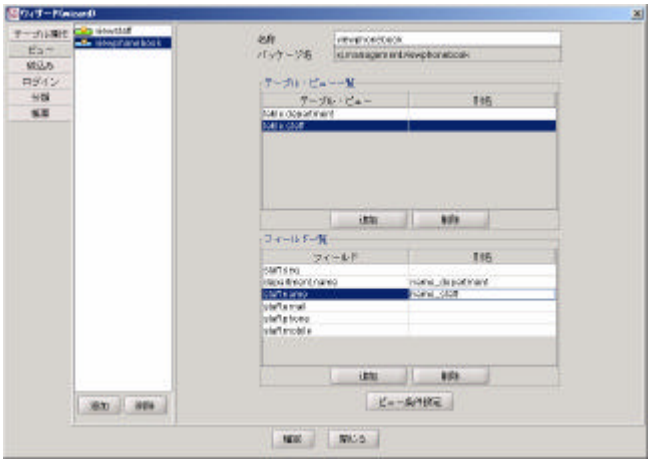
確定ボタンを押して、このテーブルに関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

第 2 項　ビューウィザード

ビューウィザードでビューを作成しその一覧画面をもつアプリケーションを自動生成します。

ウィザードを実行しビジネスオブジェクト選択のダイアログで既存のビジネスオブジェクトを選択し確定ボタンを押してください。

ダイアログ左側のウィザード選択タブからビューを選び、下方の追加ボタンでビューウィザードを追加します。



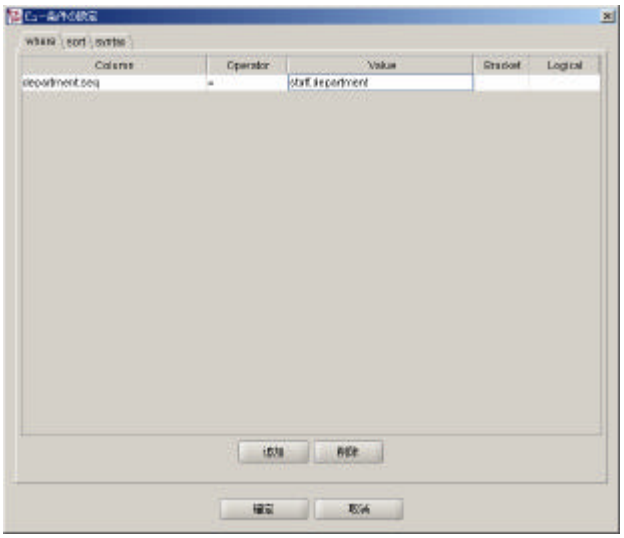
名称はデータベースのビュー名です。

テーブル・ビュー一覧にこのビューに関連するテーブル・ビューの定義ができます。追加した行のテーブル・ビューの欄を左クリック後、右クリックするとポップアップメニューからテーブルまたはビューを選択入力できます。

フィールド一覧にこのビューのフィールドを定義します。同様の方法でポップアップメニューからテーブルまたはビューのフィールドを選択入力できます。

重複の名称が存在する場合は、別名の欄にそれぞれ別名を定義して下さい。

ビュー条件設定ボタンを押して条件設定画面を表示します。



ビューの where 条件の定義を行います。ここでも右クリックでポップアップメニューからの選択入力ができます。

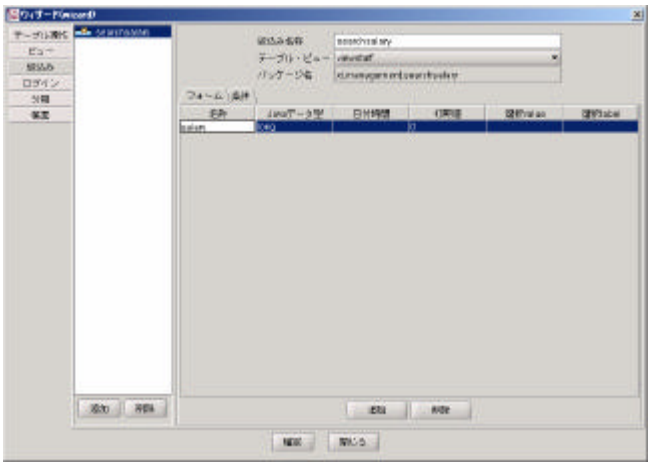
確定ボタンを押して、このビューに関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

第 3 項 絞込みウィザード

絞り込みウィザードでテーブルの中でユーザ入力の条件によって表示件数を絞り込むような画面をもつアプリケーションを自動生成します。

ウィザードを実行しビジネスオブジェクト選択のダイアログで既存のビジネスオブジェクトを選択し確定ボタンを押してください。

ダイアログ左側のウィザード選択タブから絞り込みを選び、下方の追加ボタンで絞り込みウィザードを追加します。



名称は絞込みウィザード名です。

テーブル・ビューは絞込みの対象です。

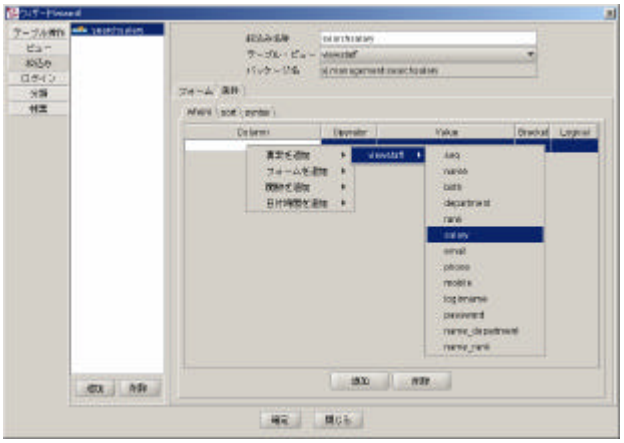
フォームタブの下の追加ボタンでフィールドを追加します。ここではフィールドの名称、java データ型、日付時間、初期値、選択 value と選択 label を定義します。

名称と java データ型は必須属性です。

Java データタイプに DateString を指定する場合は、日付時間属性でフォーマットの指定ができます。生成される java ソースの java データ型は java.lang.String です。

選択 value、選択 label に何も入力しないとフォームはテキストボックスになりユーザは自由に値を入力します。選択 value、選択 label に外部リンクしたテーブルの値を入力するとフォームはセレクトボックスになり、ユーザはその中から値を選びます。

次に検索条件の設定をします。条件タブを選んでください。



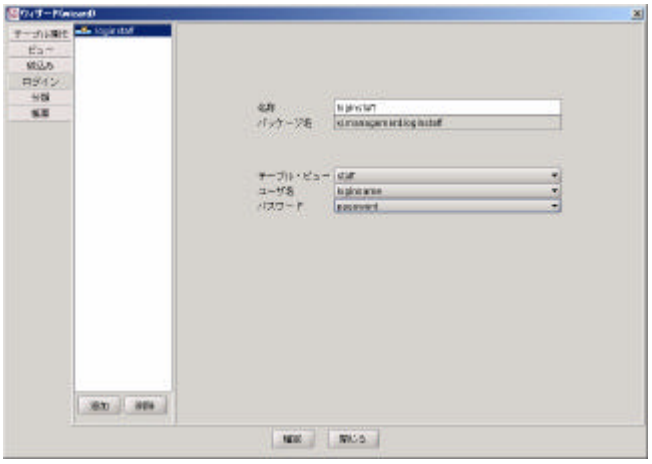
where タブで絞込みの where 条件を設定できます。

確定ボタンを押して、この絞込みに関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

第 4 項 ログインウィザード

ログインウィザードでログイン画面及びログアウト画面とその処理を生成します。

ウィザードを実行しビジネスオブジェクト選択のダイアログで既存のビジネスオブジェクトを選択し確定ボタンを押してください。
ダイアログ左側のウィザード選択タブからログインを選び、下方の追加ボタンでログインウィザードを追加します。



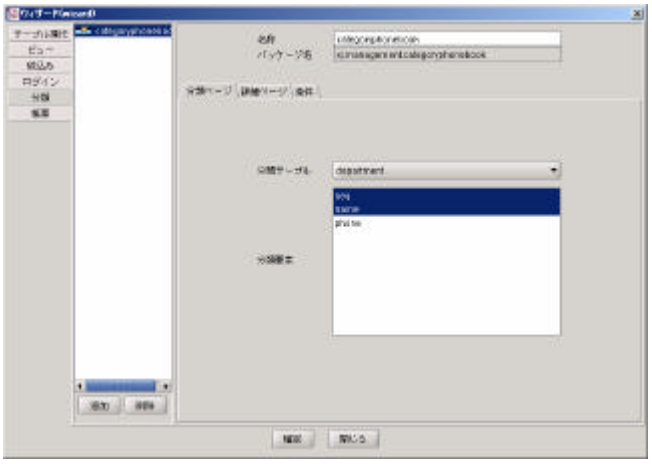
名称はログインウィザード名です。
テーブル・ビューはログイン情報が保存されているテーブルを選択します。
ユーザ名、パスワードは指定したテーブルの String フィールドの中から選択します。

確定ボタンを押して、このログインに関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

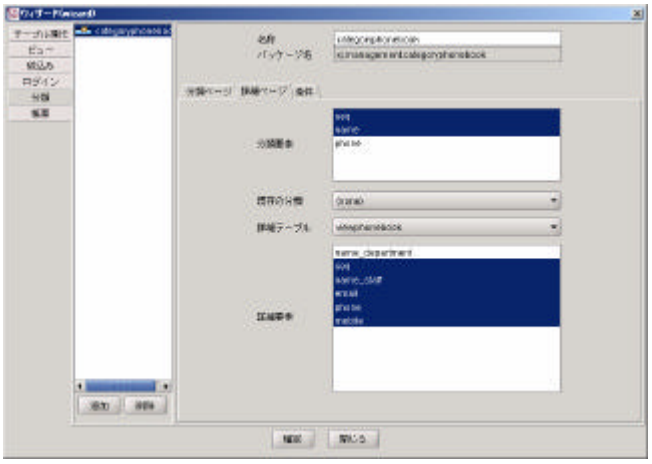
第 5 項 分類ウィザード

分類ウィザードで一覧とその詳細の関係を持つ2つのテーブルまたはビューを連携して表示する画面を自動生成します。

ウィザードを実行しビジネスオブジェクト選択のダイアログで既存のビジネスオブジェクトを選択し確定ボタンを押してください。
ダイアログ左側のウィザード選択タブから分類を選び、下方の追加ボタンで分類ウィザードを追加します。

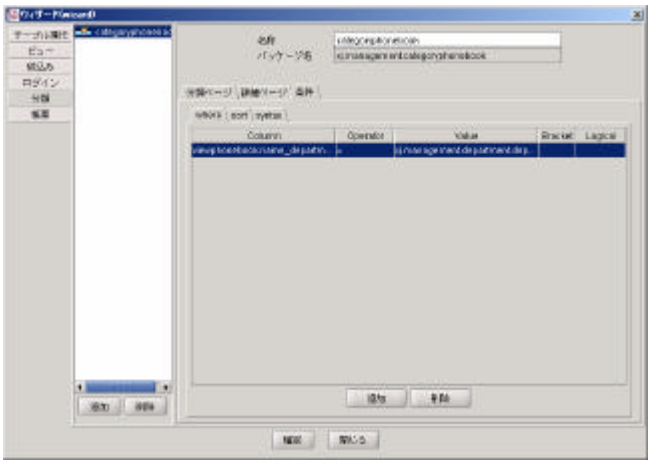


名称は分類ウィザード名です。
分類ページは二つ連携しているテーブルまたはビューの中の前のものの情報を表示するページです。ここでは、表示したい分類テーブルとそのフィールドを選択します。



詳細ページは二つ連携しているテーブル またはビューの中の後のものの情報を表示するページです。ここでは、表示したい詳細テーブルとそのフィールドを選択します。

三つ以上テーブルまたはビューを連携する場合は、A-B-C **の連携のうち、まず分類** B-C **を先に定義して、別の分類ウィザードを追加して分類ページを** A **とし、詳細テーブルの代わりに既存の分類** B-C **を選択します。**



そして、分類テーブルと詳細テーブルの連携条件を設定します。

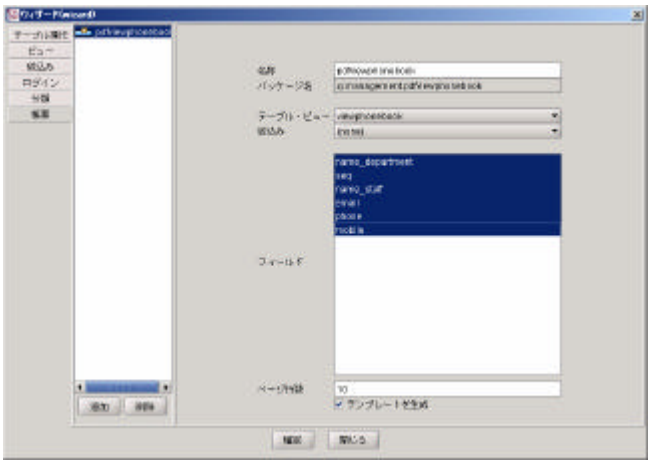
最後に、確定ボタンを押して、この分類に関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

第 6 項 帳票ウィザード

帳票ウィザードでテーブルあるいはビューからPDF 帳票の出力する機能を持った web アプリケーションを自動生成します。

ウィザードを実行しビジネスオブジェクト選択のダイアログで既存のビジネスオブジェクトを選択し確定ボタンを押してください。

ダイアログ左側のウィザード選択タブから帳票を選び、下方の追加ボタンで帳票ウィザードを追加します。



名称は帳票ウィザード名です。

テーブル・ビューとフィールドに帳票に出力したいテーブル・ビューとそのフィールドを選択します。

絞込みは検索条件です。前作った絞込みウィザードと連携ができます。

ページ行数には帳票の 1ページに何行のレコードまでを印字するかを指定します。デフォルト値は 10 です。

テンプレート生成をチェックするとデフォルトの テンプレート PDF ファイルを生成します。ユーザがテンプレートを変更した後、再び帳票ウィザードを起動してこの テンプレートを上書きしたくない場合はチェックを外してください。

確定ボタンを押して、この帳票に関する xml,java,class,html,jsp などファイルが全部自動的に生成されます。

第 3 節 ファイル

第 1 項 ファイル新規作成

新しいファイル作成しプロジェクトに追加します。

メニューからファイル->新規作成を選択しダイアログを開きます。ファイルの種類と名称を指定して確定ボタンを押すと新しいファイルが開きます。

第 2 項 ファイルオープン

ファイルエディタのフレームを開きます。
メニューからファイル->オープンを選択しダイアログを開きます。

現在のプロジェクト中のファイル一覧からファイルを選択し確定ボタンを押します。
または、メニューから最近使ったファイルを選択するとサブメニューに最近開いたファイルが表示されます。この中から直接ファイル名を選択することができます。

第 3 項 すべて保存

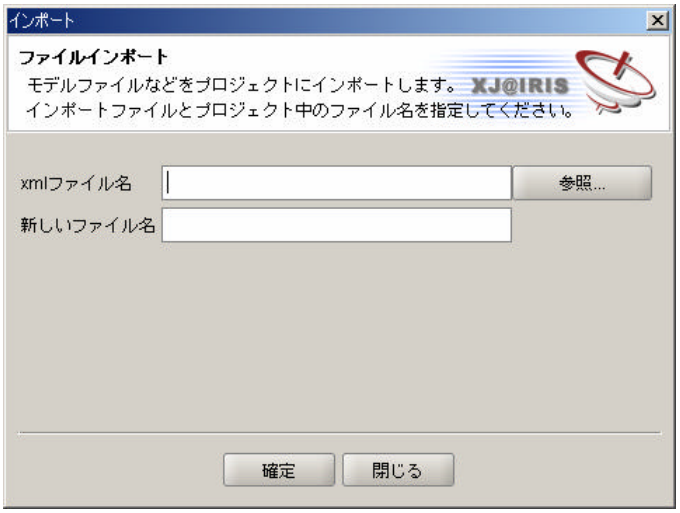
現在オープンしているファイルのうち保存していないものをすべて保存します。
メニューからすべて保存を選択します。

第 4 項 すべてクローズ

現在オープンしているファイルをすべてクローズします。
メニューからすべてクローズを選択します。

第 5 項 ファイルインポート

このコンピュータ上にあるモデルファイルをこのプロジェクトのファイルとして取り込みます。メニューからファイル->インポートを選択してダイアログを開きます。



XML ファイル名にはプロジェクト外のフルパスファイル名を入力します。新しいファイル名にはこのプロジェクト内の追加するファイル名を指定します。新しいファイル名にはスラッシュ(/)で区切ってディレクトリを指定することができます。

第 6 項 ファイル削除

プロジェクト内のファイルを削除します。
メニューからファイル->削除を選択してダイアログを開き削除するファイルを選択します。

第 7 項 エディタの移動

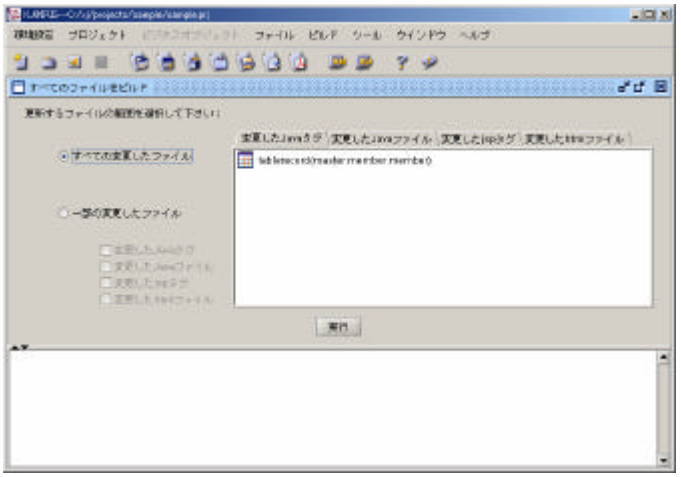
メニューまたはツールバーから前の位置、次の位置を選択するとオープンしているファイルエディタ上のノード位置を移動することができます。クイックリンクなどで移動した後に元の位置に戻るのに便利です。

第 4 節 ビルド

第 1 項 すべてのファイルをビルド

ビルドの実行が必要な更新されたモデル要素やファイルのみを一括してビルドする機能です。

メニューからすべてのファイルをビルドを選択しフレームを開きます。



このフレームでは変更されたモデルファイルの要素やファイルでビルドが必要なもののみを表示します。以下の 4 つのタブに分かれています。

- 変更した Java タグ :モデルファイル中の変更した要素を表示します。（record、form、iterate、action など Java ファイルを生成する要素です）新しい Java ファイルの生成が必要です。
- 変更した Java ファイル :変更された Java ファイルを表示します。モデルファイルから Java ファイルが生成された、あるいは別のエディタなどを使用して Java ソースファイルが書き換えられた後ビルドされていないことを示します。class の生成をする必要があります。
- 変更した jsp タグ :JSP モデルで変更された要素です。新しい HTML ファイルを生成あるいは更新する必要があります。
- 変更した html ファイル :変更された HTML ファイルを表示します。JSP モデルから HTML が生成または更新された、あるいは他のエディタなどで HTML ファイルが変更されたことを示します。JSP の生成が必要です。

ビルドは以下を選択することができます。

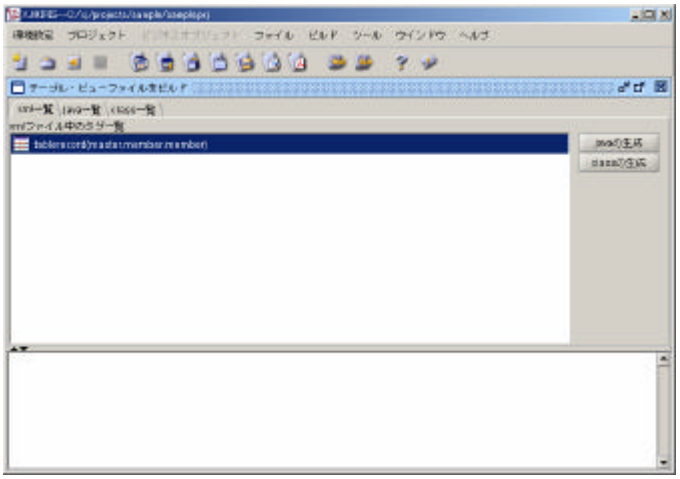
- すべての変更したファイル :上記 4つのタブのファイルをすべてビルドします。
- 一部の変更したファイル :上記 4 つのタブのファイルのうち、チェックボックスでチェックされた物のみをビルドします。

実行ボタンを押すと指定されたビルドを実行します。エラーなどは フレーム下部に表示されます。

第 2 項 テーブル・ビューファイルをビルド

メニューからテーブル・ビューファイルをビルドを選択しフレームを開きます。この フレームでは

record 要素から対応する Java ファイルを生成とコンパイル、また生成された Java の編集、削除、コンパイルされた class ファイルの削除ができます。



フレームには 3 つのタブがあります。

- XML 一覧

  このプロジェクト中のすべての record モデルファイルの record 要素が表示されます。

  java 生成ボタンを押すと選択した 1つまたは複数の要素に対応する Java ファイルが生成されます。

  class 生成ボタンを押すと選択した 1 つまたは複数の要素に対応する Java ファイルを生成しコンパイルして class ファイルを生成します。

- java 一覧

  モデルの要素に対応する Java ファイルの一覧が表示されます。

  java 削除ボタンを押すと選択した 1 つまたは複数の Java ファイルを削除します。

  java 編集ボタンを押すと選択した Java ファイルをシステム設定で指定した編集ツールで開きます。

  class 生成ボタンを押すと選択した 1 つまたは複数の Java ファイルをコンパイルしclass ファイルを生成します。

- class 一覧

  Java ファイルに対応する class ファイルの一覧を表示します。

  class 削除ボタンを押すと選択した 1 つまたは複数の class ファイルを削除します。

第 3 項 配列ファイルをビルド

メニューから配列ファイルをビルドを選択しフレームを開きます。

このフレームでは iterate 要素から対応する Java ファイルを生成とコンパイル、また生成された Java の編集、削除、コンパイルされた class ファイルの削除ができます。

画面の構成及び操作は 「テーブル・ビューファイルをビルド」と同じです。

第 4 項 フォームファイルをビルド

メニューからフォームファイルをビルドを選択しフレームを開きます。

このフレームでは form 要素から対応する Java ファイルを生成とコンパイル、また生成された

Java の編集、削除、コンパイルされた class ファイルの削除ができます。
画面の構成及び操作は 「テーブル・ビューファイルをビルド」と同じです。

第 5 項 アクションをビルド

メニューからアクションファイルをビルドを選択しフレームを開きます。
このフレームでは action 要素から対応する Java ファイルを生成とコンパイル、また生成された Java の編集、削除、コンパイルされた class ファイルの削除ができます。
画面の構成及び操作は 「テーブル・ビューファイルをビルド」と同じです。

第 6 項 その他の Java ファイルをビルド

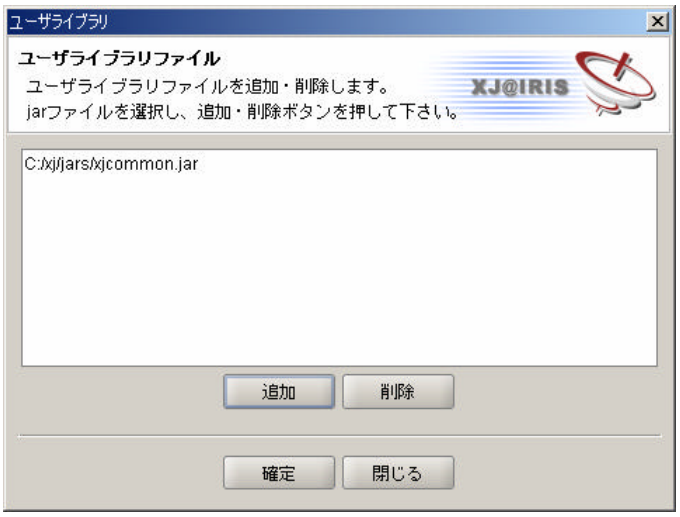
メニューからその他の Java ファイルをビルドを選択しフレームを開きます。このフレームではプロジェクトディレクトリの src ディレクトリの下にあり、どのモデルの要素にも対応しない Java ファイルに対するコンパイル、編集、削除、またコンパイルされた class ファイルの削除ができます。

フレームには 2つのタブがあります。

- java 一覧
  Java ファイルの一覧が表示されます。
  java 削除ボタンを押すと選択した 1 つまたは複数の Java ファイルを削除します。
  Java 編集ボタンを押すと選択した Java ファイルをシステム環境設定で指定した編集ツールで開きます。
  class 生成ボタンを押すと選択した 1 つまたは複数の Java ファイルをコンパイルし class ファイルを生成します。
- class 一覧
  Java ファイルに対応する class ファイルの一覧を表示します。
  class 削除ボタンを押すと選択した 1 つまたは複数の class ファイルを削除します。

第 7 項 ユーザライブラリ

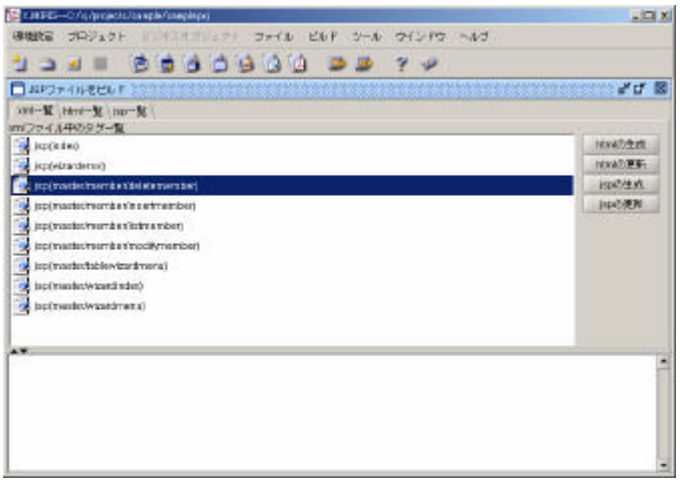
現在のプロジェクトに必要なユーザライブラリの jar ファイルを指定します。ビルドと War の生成の時にこの設定の内容が反映されます。
メニューからユーザライブラリを選択します。



追加ボタンを押して jar ファイルを追加します。ファイルを選択して削除ボタンを押すとその jar ファイルが現在のプロジェクトから参照されなくなります。

第 8 項 JSP ファイルをビルド

メニューから JSP ファイルをビルドを選択しフレームを開きます。このフレームでは JSP モデルの jsp 要素から対応する html ファイルを生成・更新と削除、また html から jsp ファイルの生成と編集、削除ができます。



フレームには 3 つのタブがあります。

- xml 一覧

  このプロジェクト中のすべての JSP モデルファイルの jsp 要素が表示されます。

  html 生成ボタンを押すと選択した 1つまたは複数の要素に対応する html ファイルが生成されます。html 生成ではモデルファイルから新しい html ファイルを作成し、もし古い同名のファイルがあった場合は上書きします。

  html 更新ボタンを押すと選択した 1つまたは複数の要素に対応する html ファイルが更新されます。html 更新では古い html ファイルがあるとき、モデルに定義された要素の属性のみを置換え html ファイル中のその他のタグや文字列は変更しません。

  jsp 生成ボタンを押すと選択した 1 つまたは複数の要素に対応する HTML ファイルを生成しさらにこのファイルをビルドして JSP ファイルを生成します。

jsp 更新ボタンを押すと選択した 1 つまたは複数の要素に対応する HTML ファイルを更新し、さらにこのファイルをビルドして JSP ファイルを生成します。

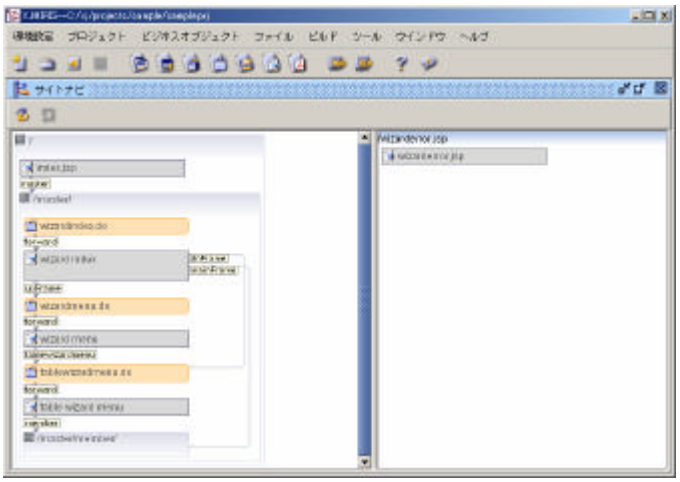
- html 一覧
  モデルの要素に対応する HTML ファイルの一覧が表示されます。
  html 削除ボタンを押すと選択した 1 つまたは複数の HTML ファイルを削除します。
  jsp 生成ボタンを押すと選択した 1 つまたは複数の HTML ファイルをビルドしJSP ファイルを生成します。
- jsp 一覧
  HTML ファイルに対応する JSP ファイルの一覧を表示します。
  jsp 削除ボタンを押すと選択した 1 つまたは複数の JSP ファイルを削除します。
  jsp 編集ボタンを押すと選択した JSP ファイルをシステム設定で指定した編集ツールで開きます。

第 5 節 サイトナビ

サイトナビでは現在のプロジェクト中のページとアクションの関連を視覚的に表示します。また、遷移を変更することができます。

第 1 項 メインウインドウ

メニューからツールサイトナビを選択するとフレームが開きます。



このウインドウのは左右のペインに分かれています。左側は現在のプロジェクトの入り口から繋がる一連のページとアクションのツリーになっています。右側はプロジェクトの入り口から繋がっていない独立したページやアクションで、さらに複数のツリーが表示されることがあります。

最新の状態に更新ボタン :モデルファイルを変更した場合、ファイル保存後にこのボタンを押すことでサイトナビの画面を更新します。

第 2 項 グループノード

グループノードはいくつかのノードが集まった矩形領域で内部にいくつかの関連するノードが含まれています。グループノードの の部分をクリックすると内部を展開して表示します。再度クリックすると元の状態に戻すことができます。グループノードには中に含まれるノードのプロジェクトからの相対パスが表示されています。ウインドウ左側のもっとも外側のグループノードが現在のプロジェクトのルートになります。ルートは web.xml ファイル中の welcome-files に指定されたものになります。

第 3 項 JSP ノード

jsp ノードは 1つの jsp ページに対応します。このノードの上にマウスカーソルを合わせると数秒後にこのノードのツールチップが表示されます。この jsp ノードのプロジェクトからの相対パスなどの情報が含まれます。

このノードの上でマウスの右クリックをすると以下のようなポップアップメニューが表示されます。

- ここにマークする：

  このページ位置にマーク を設定します。これは別の遷移要素からこの位置へ遷移先を変更するために利用します。
- パスをコピー：

  このページのプロジェクト相対のパスをシステムのクリップボードにコピーします。
- マークした要素～の遷移先をここに変更：

  別の遷移要素が既にマークされている場合のみ表示されます。別の遷移要素の移動先をこのページノードに変更します。
- jsp ～を編集：

  対応する jsp モデルの編集画面を開きます。

第 4 項 action ノード

action ノードは struts-config.xml ファイルの action-mappings 内の action タグに相当します。

action ノードの上にマウスカーソルを移動して数秒待つとプロジェクト相対のパスや実行するアクションのクラス名などを含むツールチップが表示されます。action ノードの上で右クリックすると以下のようなポップアップメニューが表示されます。

- ここにマークする：

  この action 位置にマーク  を設定します。これは別の遷移要素からこの位置へ遷移先を変更するために利用します。
- パスをコピー：

  この aciton のプロジェクト相対のパスを文字列としてシステムのクリップボードにコピーします。
- マークした要素～の遷移先をここに変更：

  別の遷移要素が既にマークされている場合のみ表示されます。別の遷移要素の移動先をこの action ノードに変更します。
- ActionMapping ～を編集：

  対応する struts-config のタグ編集画面を開きます。
- ActionModel ～を編集：

  対応する action モデルのタグ編集画面を開きます。
- ソースファイル～を編集：

  システム設定で指定した編集ツールで対応する Java ソースファイルを開きます。

第 5 項 遷移要素

遷移要素はページ要素や action 要素の間をつなぐ矢印と名前を付けたラベルでできています。同じ遷移元や遷移先を共有する場合には矢印の一部が重なって表示されます。矢印の重なっている部分では可能な操作が限られます。ラベルは一つの矢印の部分（重なっていない部分）に表示されます。遷移要素の矢印やラベルの上でマウスの右クリックすると次のポップアップメニューが表示されます。

- ここにマークする：

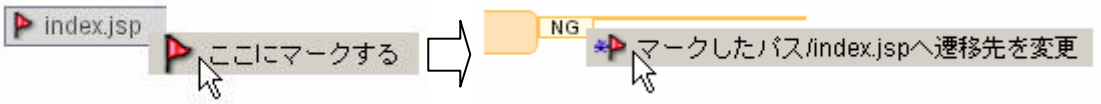
この遷移ノードにマーク  を設定します。これは別のページまたは action 要素の遷移元をこの位置へ変更するために利用します。

- ～を編集：
  対応する struts-config または JSP モデルのタグ編集画面を開きます。
- マークしたパス～へ遷移先を変更：
  別のページまたは action 要素が既にマークされている場合のみ表示されます。別のページまたは action 要素の移動元をこの遷移ノードに変更します。
- 遷移元～を表示：
  矢印の起点のノードを画面上に見えるようにスクロールします。
- 遷移先～を表示：
  矢印の終点のノードを画面上に見えるようにスクロールします。

第 6 項 遷移の変更

遷移を変更する場合、変更する遷移ノード（矢印）と新しい遷移先（ページまたは action ノード）を指定する必要があります。どちらを先に指定することもできます。先に指定する操作は「～をマークする」メニューで行います。マークを指定した後、残る指定位置で「マークした位置へ変更」を実行して操作を完了します。

先に遷移先を指定する場合：



先に遷移元を指定する場合：

これで遷移が変更され画面が更新されます。

第 6 節 SQL ファイル

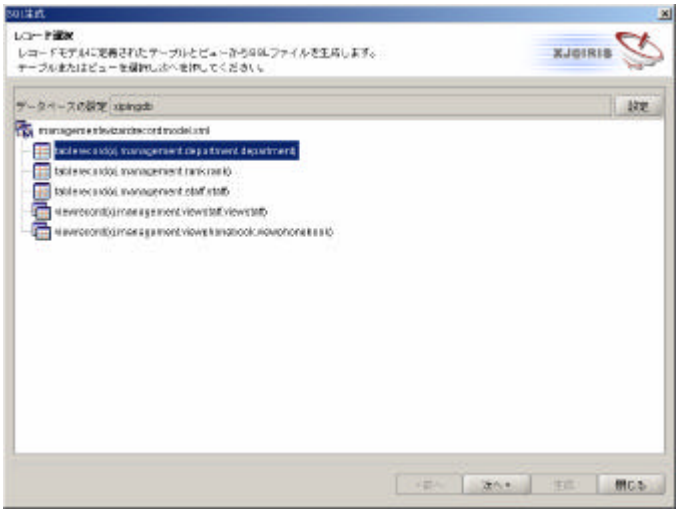
XJ@IRIS には SQL ファイルの生成と実行の機能があります。レコードモデルで定義した record 要素から選択したデータベースの文法に適合したテーブルやビューの create 及び drop 文の SQL を生成します。また生成した SQL 文を直接データベースサーバに対して実行することができます。

第 1 項 SQL ファイル生成

メニューからツール->SQL 生成を選択してダイアログを開きます。

■ レコード選択画面

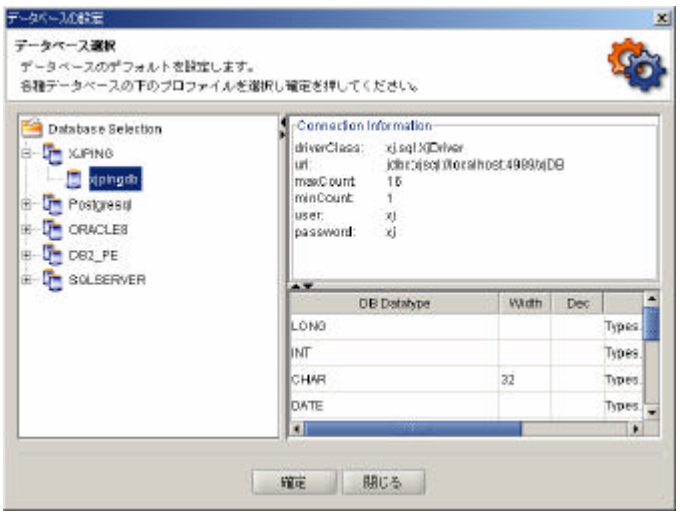
SQL 生成の最初の画面は以下のような record 選択の画面です。



データベースの設定ではデータベースを選択します。生成されるSQL の文法はこのデータベースに適合したものになります。必要ならば設定ボタンを押して変更します。

record 一覧にはこのプロジェクトの record 要素がモデルファイルごとにツリー状に表示されます。モデルファイルをクリックすると中の record 要素が表示されます。この中から 1 つの record 要素を選択し次へボタンを押します。

設定ボタンを押すと以下のような画面が表示されます。



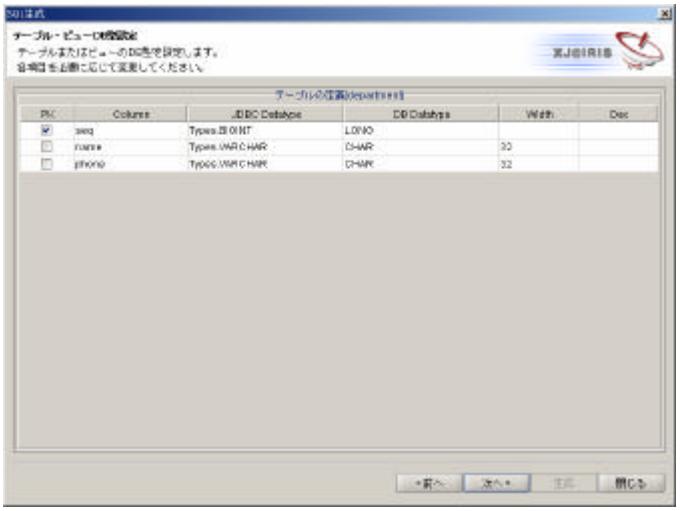
データベース環境設定で定義したデータベースプロファイルの中から1つを選択します。左側のツリーにはデータベースの種類ごとのプロファイルが表示されます。右側のペインには選択したプロファイルの情報が表示されます。

■ テーブル・ビューDB 型設定画面

この画面では選択したテーブルまたはビューのフィールドについて、データベース型などのパ

ラメータを必要に応じて調整します。

テーブルの場合の型定義画面は以下の様になります。



表中の各列の意味は以下のとおりです。

PK :プライマリキー。レコードモデルの中で定義されています。

Column :フィールド名。レコードモデルの中で定義されています。

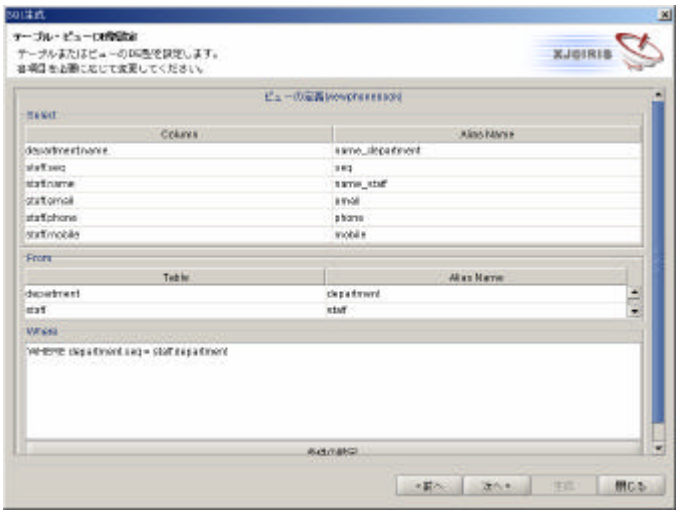
JDBC Datatype :JDBC データ型。レコードモデルの中で定義されています。

DB Datatype :データベースデータ型。JDBC データ型に対応するデータベース型の中から選択することができます。

Width :長さ。DB Datatype の長さが必要な場合に設定します。

dec :小数長さ。DB Datatype の小数長さが必要な場合に設定します。

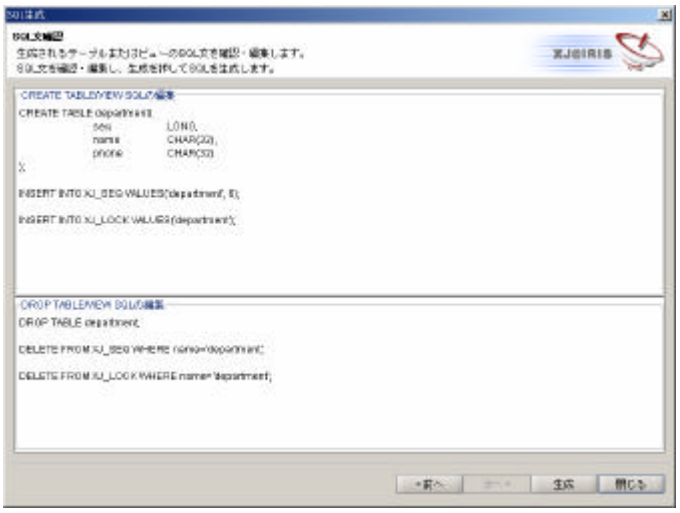
ビューの場合の型定義画面は以下の様になります。



必要に応じて条件の設定ボタンを押して条件を変更します。

■ SQL 文確認画面

生成される SQL 文を確認します。

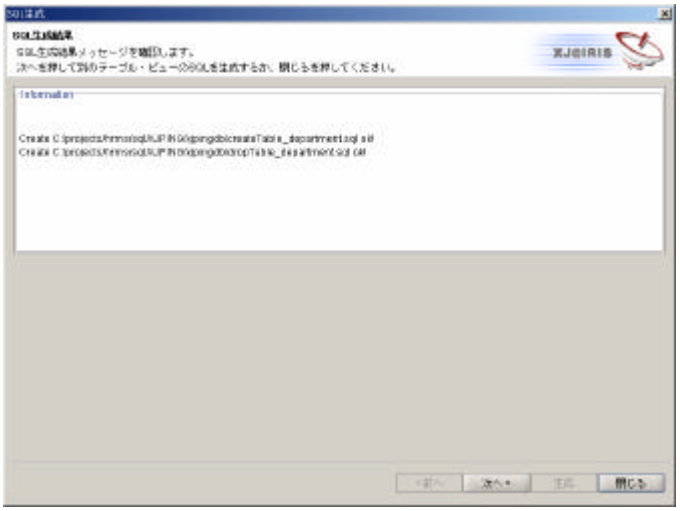


テーブルの場合は"createTable_テーブル名.sql"と"dropTable_テーブル名.sql"
ビューの場合は"createView_ビュー名.sql"と"dropView_ビュー名.sql"が生成されます。画面上部に create の SQL 文、下部に drop の SQL 文が表示されます。

XJ@IRIS のデータベースライブラリはレコードごとにユニークな ID を自動的に付与する機能をサポートしています。このためこれ以外に XJ_SEQ という名前のシーケンス表が必要になります。データベースごとに 1 つめの SQL を生成する際に、このテーブルの create、drop の SQL 文も同時に生成されます。またデータベースサーバとして xjPing を使っている場合はロックの為のテーブル XJ_LOCK も生成されます。

■ SQL 生成結果画面

生成の結果を表示する画面です。



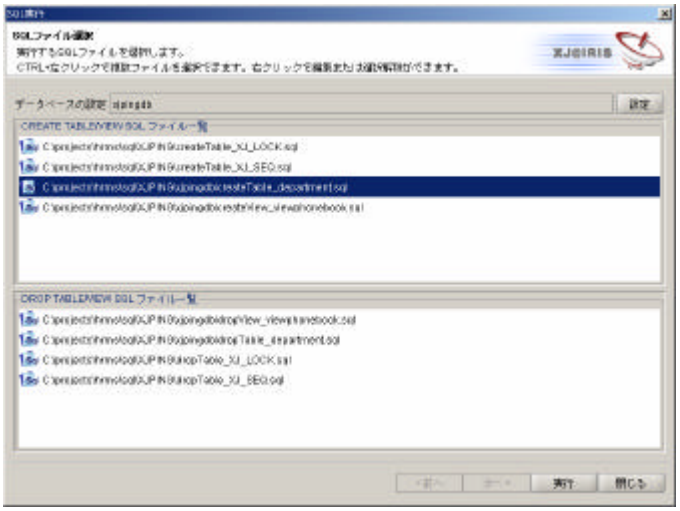
作成されたファイル名が表示されます。
作成されるディレクトリは以下の様になります。
"プロジェクトディレクトリ/sql/データベース種類/ファイル名.sql"
たとえば、
C:/projects/hrms/sql/XJPING/xjpingdb/createTable_department.sql
となります。

第 2 項 SQL ファイル実行

メニューからツール->SQL 実行を選択してダイアログを開きます。

■ SQL ファイル選択画面

SQL 実行の最初の画面は以下のような SQL 選択の画面です。



データベースの設定：

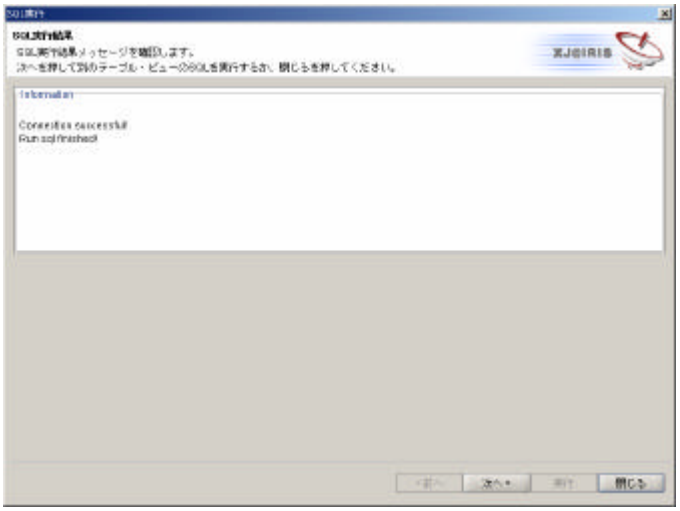
データベースプロファイルを設定します。値を変更する場合は設定ボタンを押して設定してください。詳しくは SQL 生成の説明を参照してください。

SQL ファイル一覧：

現在のプロジェクトの選択されたデータベースの SQL ファイルの一覧です。上部に生成の SQL、下部に削除の SQL が表示されます。マウスクリックで 1つまたは複数の SQL を選択し実行ボタンを押します。

■ SQL 実行画面

SQL の実行結果を表示します。



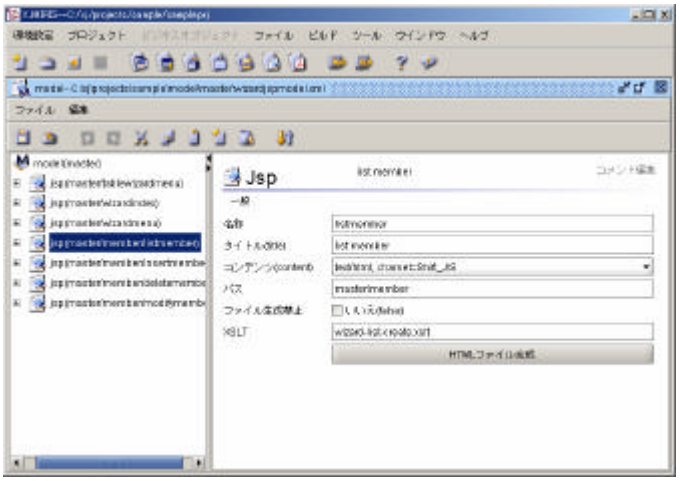
次へボタンを押して別の SQL ファイルを実行するか、閉じるを押してください。

第 7 節 HTML 合成

JSP モデルで定義した要素とその他のページの飾りや固定の文字列などを HTML ファイルに合成する機能です。

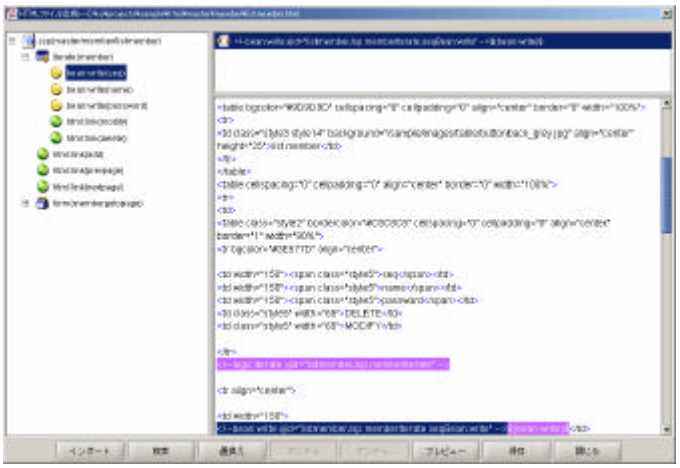
第 1 項 HTML 合成エディタ

ファイルオープンで JSP ファイルを開き jsp 要素を選択します。



編集ペインの HTML ファイル合成ボタンを押すと、選択した jsp 要素対応する HTML ファイルが HTML エディタ画面で編集できます。

HTML エディタは以下のような画面です。



画面の説明

- 左ペイン :対応する jsp 要素とその内部要素をツリー状に表示します。編集するノードを選択してそのノードに対応するタグの置換えを行います。
- 右上ペイン :左ペインで選択したノードのタグを表示します。HTML ファイル中のタグの置換えに使用します。jsp 要素を選択した場合は jsp 要素内のすべての要素を表示します。
- 右下ペイン :HTML ファイルを編集します。JSP モデルの要素に対応するタグの部分は紫色

または青色になります。青色の部分は左ペインで選択した要素に対応するタグです。
ボタンの説明

- インポート:デザイン済の HTML ファイルで HTML エディタの部分（右下ペイン）を置換えます。現在のエディタの内容は上書きされます。このボタンを押し読み込み元ファイルを指定します。
- 検索 :HTML の中から文字列を検索します。
- 置換え :HTML の中の文字列を置換えます。
- アンドゥ :一つ前の編集を取り消します。
- アンドゥ取消 :アンドゥで取り消した編集を元に戻します。
- プレビュー :システムデフォルトのブラウザで編集中の HTML を表示します。
- 保存 :HTML ファイルを保存します。
- 閉じる :HTML 合成エディタを閉じます。

第 2 項 HTML ファイル合成の操作手順

1. JSP ファイルを開きます。
2. jsp 要素とその子要素を追加し属性を設定します。または既存の jsp 要素を変更します。JSP モデルファイルを保存します。
3. JSP ファイルのビルドを実行しシンプルな HTML ファイルを生成します。もしjsp 要素が新しく追加したものでない場合はこのビルドは必要ありません。
4. jsp 要素を選択し HTML ファイル合成ボタンを押し、HTML 合成エディタを開きます。HTML のタグをJSP モデルのタグに対しHTML の中のタグの過不足を調整します。あるいは先にデザイン済の HTML ファイルをインポートし、タグの置換えをします。
5. タグの置換えが終わったらHTML を保存し、JSP のビルドで jsp 生成を実行します。

第 8 節 入力値検査 validation

Strtus フレームワークの提供するフォーム入力の validation 機能では validation.xml ファイルを定義する必要があります。XJ@IRIS ではこの ファイルを直接定義するのではなくJSP モデルで必要な情報を定義しvalidation.xml を自動生成します。

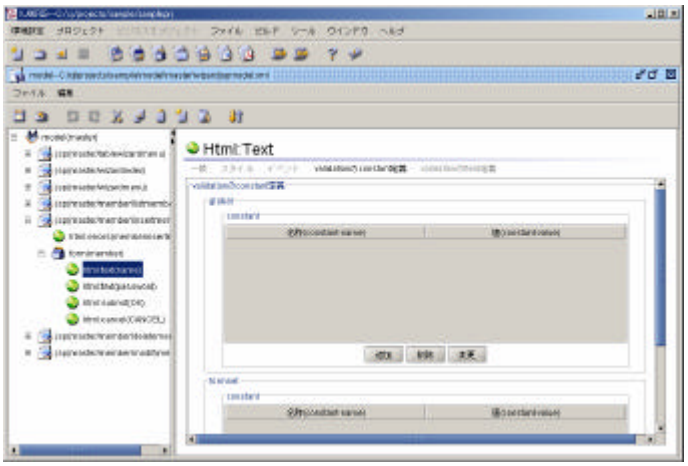
validation を定義できる JSP モデルの要素は html:text、html:textarea、html:password です。

validation.xml については Strtus のドキュメントを参照してください。

JSP モデルを編集で html:text、html:textarea あるいは html:password 要素を選択し validation の constant 定義または validation の field 定義タブを選択します。

第 1 項 validation の constant 定義

validation の constant 定義の画面は以下の様になります。



■　global constant の定義

global の枠の中の constant 領域は validation.xml の global 要素の下の constant 要素に相当します。

該当する部分の XML ファイルの例：

```
<global>
  <constant>
    <constant-name>name</constant-name>
    <constant-value>value</constant-value>
  </constant>
</global>
```

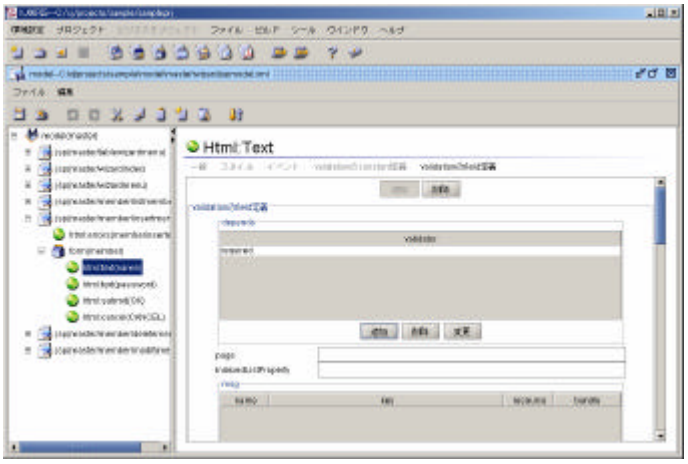
constant 領域のボタンで定数定義を追加、変更または削除します。

■　formset constant の定義

formset の枠の中の constant 領域は validation.xml の formset 要素の下の constant 要素に相当します。同様に定義してください。

第 2 項 validation の field 定義

validation の field 定義の画面は以下の様になります。



この画面ではフォームの下の 1 つの field に対する validation を定義します。

該当する部分の XML ファイルの例：

```
<formset>
    <form name="xj_management_loginstaff_loginstaffloginForm">
```

```
    <field property="password" depends="required">
      <arg0 key="management.loginloginstaff.password"/>
    </field>
  </form>
</formset>
```

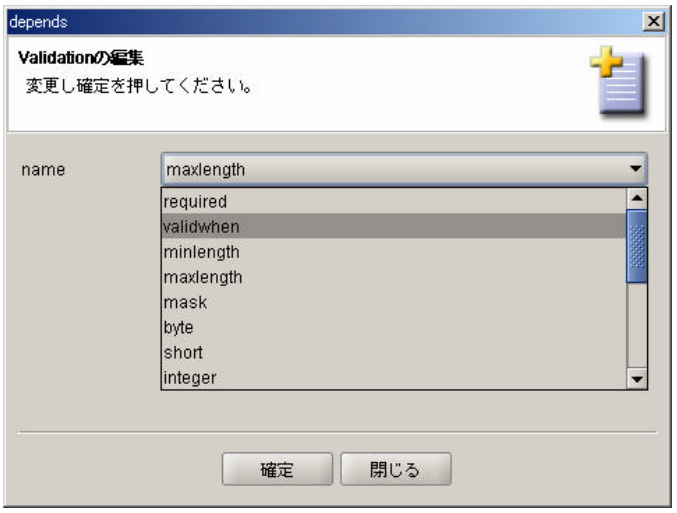
■　上部の追加、削除ボタン

この画面の上部の追加、削除ボタンでこの入力フィールドの validation を有効にするか無効にするかを変更します。

■　depends 枠

field 要素の depends 属性に相当します。

追加ボタンを押すと次のようなダイアログが表示されます。



追加する属性を選択します。確定ボタンを押すと画面中に追加した属性が表示されます。

page	field 要素の page 属性に相当します。
indexListProperty	field 要素の indexListProperty 属性に相当します。
msg	field 要素の msg 属性に相当します。
arg	field 要素の arg 属性に相当します。
arg0	field 要素の arg0 属性に相当します。
arg1	field 要素の arg1 属性に相当します。
arg2	field 要素の arg2 属性に相当します。
arg3	field 要素の arg3 属性に相当します。
var	field 要素の var 属性に相当します。

第 9 節　War ファイル

ここでは XJ@IRIS のプロジェクトで作成したアプリケーションの最終出力について説明します。War 生成ではこのプロジェクトが Struts フレームワークで動作するために必要な class ファイルや JSP ファイル、XML ファイルなどを含んだ War ファイルを作成します。

第 1 項 一致性検査

一致性検査はモデルの要素間の参照関係を検査します。この機能によって開発者はモデル定義の間違いを早い段階で修正しバグを防ぐことができます。

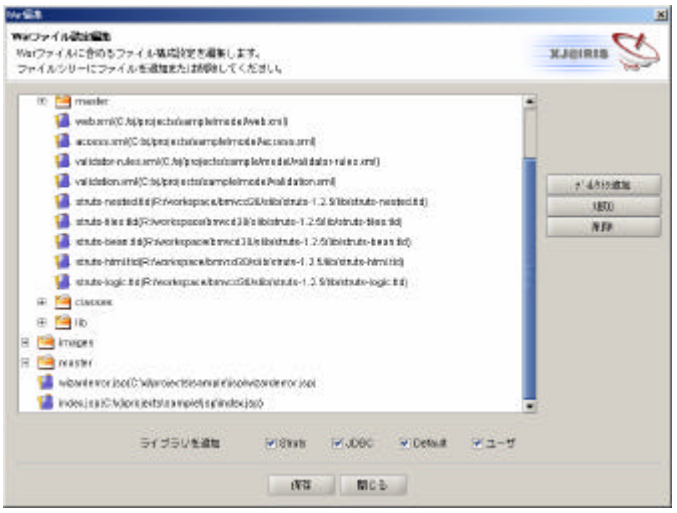
メニューからビルド>一致性検査を選択すると現在の プロジェクトのすべての モデルファイルについて一致性検査を実行します。結果はログ画面に表示されます。もしエラーがあった場合はログのエラーの行をクリックすると関連するモデルエディタを開くことができます。

War ファイル生成を実行しても先に一致性検査が実行されます。

第 2 項 War ファイル設定編集

War 生成で作成する War ファイルに含めるファイルを定義します。

メニューからビルド>War 編集を選択してダイアログを開きます。



■ 画面説明

ファイルツリー：

War ファイルに含めるファイルがツリー状に表示されます。ツリーから1 つのノードを選択しこのボタンを押すことで指定ファイルを変更します。

ライブラリを追加：

いくつかの規定の ライブラリ、及びユーザライブラリを含めるかどうかをチェックボックスで指定します。Strtus は Struts の提供する jar ファイル、JDBC は設定された DB プロファイルの JDBC ドライバの jar ファイル、Default は XJ@IRIS の提供する irislib.jar ファイルです。

ディレクトリ追加 ボタン：

名前を入力すると選択ノードがディレクトリの子として、ファイルの場合はその後ろに新しいディレクトリが追加されます。

追加ボタン：

ファイルを指定すると選択した位置にファイルを追加します。

削除ボタン：

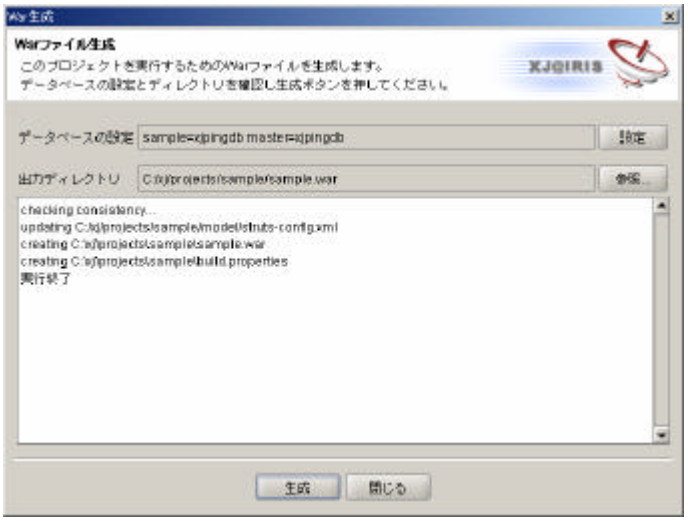
選択したファイルを削除します。

保存ボタン：

設定した情報を保存します。プロジェクトディレクトリ/model/buildwar.xml に保存されます。

第 3 項 War ファイル生成

War ファイルを生成します。先に War 編集で War ファイルの設定をしておく必要があります。メニュからビルド>War 生成を選択します。



このダイアログを開く前に XJ@IRIS はビルドの必要なファイルがないかチェックします。もしあった場合は警告が出ます。

■ 画面の説明

データベースの設定：

このプロジェクトの ビジネスオブジェクトごとにデータベースプロファイルを設定します。ここで指定された値がデータソースの決定や War に含む JDBC ドライバを決定します。変更する場合は設定ボタンを押して設定ダイアログを開いてください。

出力ディレクトリ：

War ファイルを出力するパスを設定します。

ログペイン：

ログが表示されます。

生成ボタン：

一致性検査と War 生成を実行します。

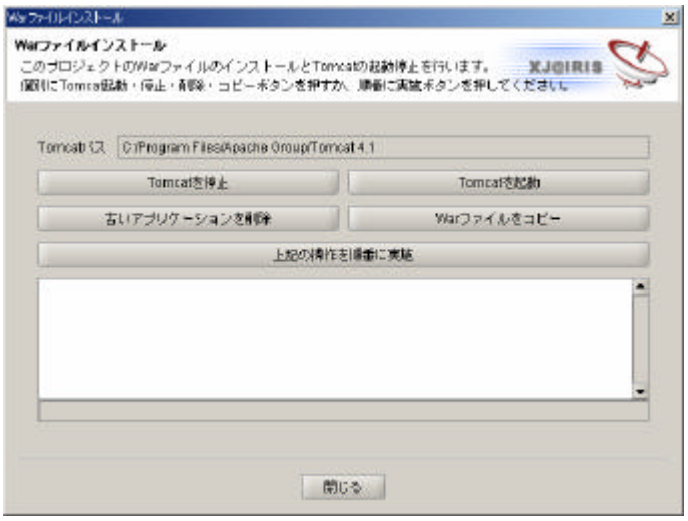
閉じるボタン：

このダイアログを閉じます。もし実行中にエラーがあった場合はログフレームを開きます。

第 4 項 War ファイルインストール

生成した War ファイルを Tomcat の指定のディレクトリにコピーします。また、Tomcat を再起動します。

メニューからツール->War ファイルインストールを選択してダイアログを開きます。



■ 画面の説明

Tomcat パス：

システム環境設定で指定された Tomcat のパスを表示します。

Tomcat を停止：(1)

Tomcat 終了命令を実行します。

古いアプリケーションを削除：(2)

古い web アプリケーションの関連しているファイルを削除します。tomcat インストールのディレクトリの webapps ディレクトリ下でプロジェクト名のディレクトリと同名の war ファイルを削除します。

war ファイルをコピー：(3)

プロジェクトの.war ファイルを tomcat の webapps ディレクトリの中にコピーします。

Tomcat を起動：(4)

Tomcat 起動命令を実行します。

上記の操作を順番に実施：

以上(1)-(4)を順次実行します。

ログペイン：

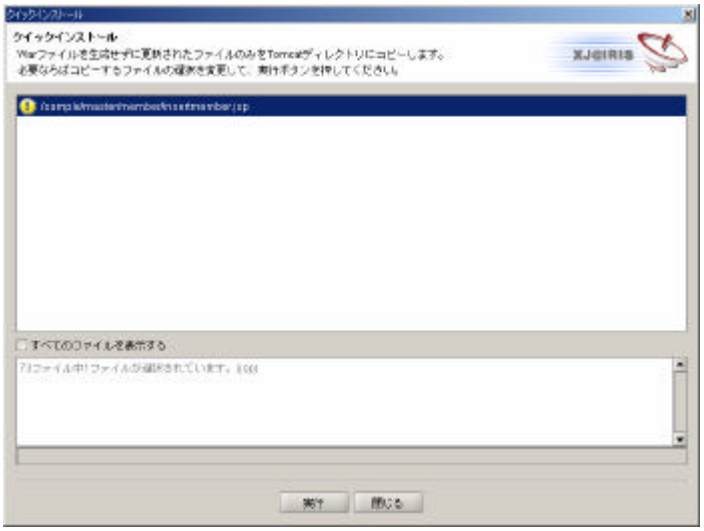
ログが表示されます。

閉じるボタン：

このダイアログを閉じます。もし実行中にエラーがあった場合はログフレームを開きます。

第 5 項 クイックインストール

War ファイルを生成せず更新されたファイルだけをコピーすることで、高速にプロジェクトのインストールを完成することができます。 クイックインストールツールは プロジェクトディレクトリの中のファイルと Tomcat の webapps ディレクトリのファイルの更新状況を判断して、必要なファイルだけをコピーします。さらに必要に応じて Tomcat の終了、起動も行います。
メニューからツール->クイックインストールを選択すると次のダイアログが表示されます。



■ 画面の説明

ファイル一覧：

War 定義で指定したファイルのうち更新されたもののみを表示します。選択されているファイルのみがコピーされます。

すべてのファイルを表示する：

ここをチェックするとファイル一覧には War 定義のすべてのファイルが表示されます。

ログペイン：

ログが表示されます。

実行ボタン：

選択されたファイルを Tomcat の webapps ディレクトリ以下にファイルをコピーします。 選択されたファイルの中に、 拡張子が “xml ”、“class ”、‘jar ”、“tld ”と ‘propertites ”の ファイルを含む時には Tomcat の再起動を行います。

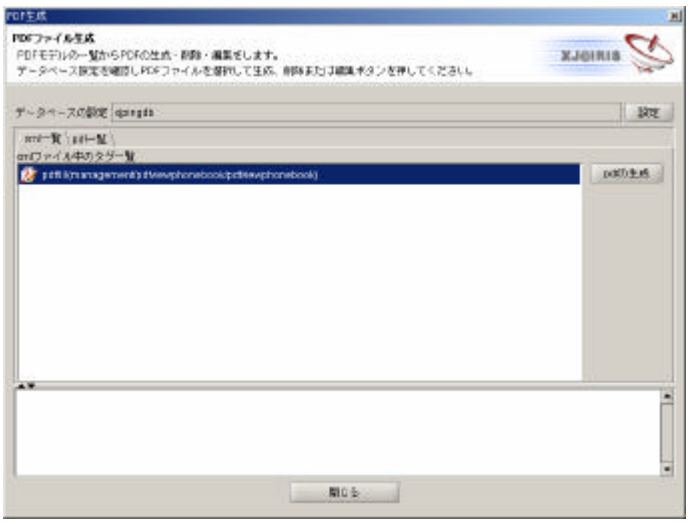
閉じるボタン：

このダイアログを閉じます。もし実行中にエラーがあった場合はログフレームを開きます。

第 10 節 PDF ファイル生成

XJ@IRIS は単体で PDF ファイルを出力する機能があります。PDF を出力する Web アプリケーションのデバッグ用などに利用します。

メニューからツール->PDF 生成を選択します。



PDF 実行のためには PDF モデルが定義してある必要があります。

■ 画面の説明

データベースの設定：

pdfdatasource の要素のデータベースプロファイルが設定されています。設定ボタンで変更することができます。

xml 一覧：

PDF モデルで定義された pdffill 要素の一覧が表示されます。1つまたは複数の要素を選択して対応する PDF ファイルを生成します。

pdf 一覧：

プロジェクト内の出力された PDF ファイルが表示されます。選択して編集及び削除ができます。

pdf の生成：

選択した要素の PDF ファイルを生成します。

pdf の削除：

選択した PDF を削除します。

pdf の編集：

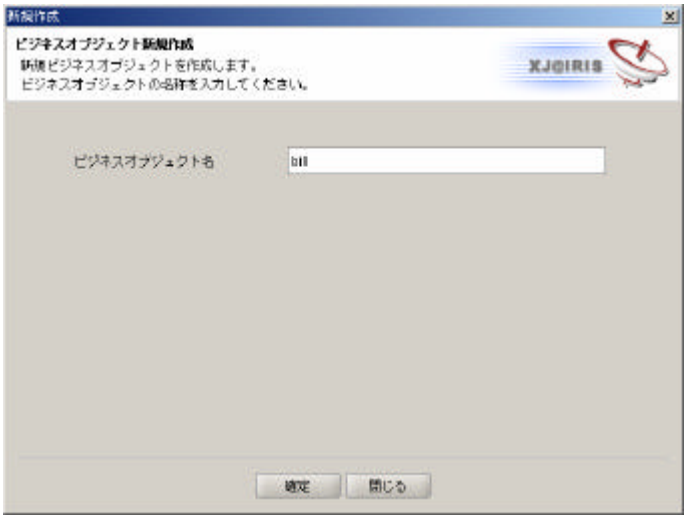
システム環境設定で指定したエディタで PDF ファイルを開きます。

第 11 節 ビジネス・オブジェクト

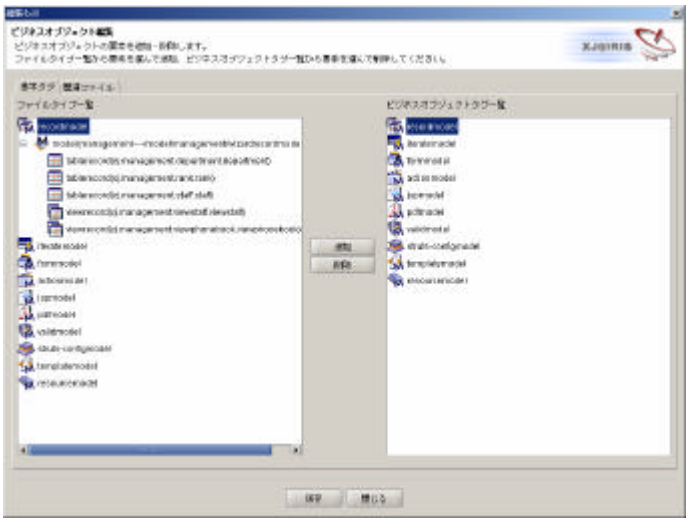
XJ@IRIS で作成されたプロジェクトはビジネスオブジェクト(business object)で構成されています。関連したグループごとにビジネスオブジェクトを分割しておくことで別のプロジェクトでの再利用を容易にします。

ビジネスオブジェクトの操作を実行する前にプロジェクト内のファイルをすべてクローズしてください。

第 1 項 ビジネスオブジェクト新規作成



新しいビジネスオブジェクトの名称を指定して確認ボタンを押します。ビジネスオブジェクトの編集画面が開きますので、新しく作成したビジネスオブジェクトを編集します。編集の操作方法は編集の節を参照してください。



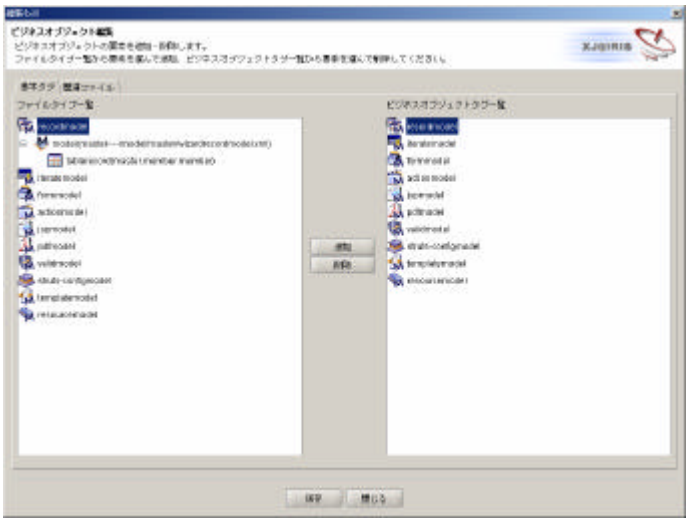
第 2 項 ビジネスオブジェクトコピー

新しいビジネスオブジェクトの名称と既存のビジネスオブジェクトを指定して内容をコピーします。確定ボタンを押すと新しいビジネスオブジェクトの編集画面が開きます。

第 3 項 ビジネスオブジェクト編集

ビジネスオブジェクトを編集します。
メニューからビジネスオブジェクト>編集を選択し、編集するビジネスオブジェクトを選択してく

ださい。



基本タグ画面と関連ファイルの画面をタブで切り替えます。

■　基本タグ画面の説明

ファイルタイプ一覧：

　ツリーをダブルクリックして展開し、このプロジェクトに含まれる各種モデルの要素を表示します。

ビジネスオブジェクトタグ一覧：

　編集中のビジネスオブジェクトに含まれる各モデルの要素を表示します。

追加ボタン：

　ファイルタイプ一覧で選択したモデルファイルまたは要素を現在のビジネスオブジェクトに追加します。

削除ボタン：

　ビジネスオブジェクトタグ一覧で選択したモデルファイルまたは要素を削除します。

■　関連ファイル画面の説明

ビジネスオブジェクトファイル一覧：

　ビジネスオブジェクトに含めるモデル要素以外のファイルを表示します。

ディレクトリ追加ボタン：

　名前を指定して現在選択中のノード下または子としてディレクトリを追加します。

追加ボタン：

　ファイルを指定して現在のノード位置にファイルを追加します。

削除ボタン：

　選択したノードを削除します。

第 4 項　ビジネスオブジェクト削除

　ビジネスオブジェクト名称を選択して現在のプロジェクトからビジネスオブジェクトを削除します。

第 5 項 ビジネスオブジェクトインポート

エクスポートしたビジネスオブジェクトを現在のビジネスオブジェクトに取り込む機能です。インポートパスにエクスポート済みのアーカイブファイル（zip ファイル）を指定してください。もしインポートするビジネスオブジェクトの名称、ファイル名と現在のプロジェクト内のビジネスオブジェクト名が重複する場合は名称を変更します。ログペインにログが表示されます。

第 6 項 ビジネスオブジェクトエクスポート

プロジェクト内のビジネスオブジェクトに含まれるモデル要素とその他のファイルをまとめてファイルに出力する機能です。出力したファイルは別のプロジェクトにインポートすることができます。

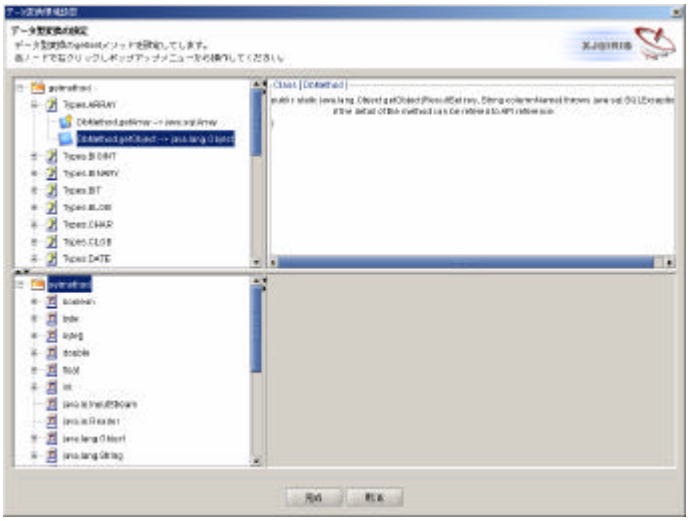
出力するファイルのディレクトリを指定し出力する現在のプロジェクト内のビジネスオブジェクトを選択します。実行ファイルを押すと zip ファイルが出力されます。

第 12 節 データ型変換設定

XJ@IRIS でレコードモデルを定義するときにデータベーステーブルのフィールドごとにデータ型変換方法を指定する必要があります。これには resultset から JDBC データ型への変換方法である getMethod と、その他の Java データタイプからそのフィールドの Java データ型への変換方法である setMethod があります。この節では主にこれらの定義方法について説明します。

第 1 項 データ変換設定画面

メニューから設定->データ型変換設定を選択します。



■　画面の説明

JDBC データ型一覧：

　左上のペインにはサポートしている JDBC データ型がツリー状に表示されます。getMethod を右クリックしてポップアップメニューから JDBC データ型を追加することができます。

getMethod 内容：

　右上のペインには選択した getMethod の内容が表示されます。ユーザメソッドの場合はソースコードが表示されます。ビルドインメソッドの場合はソースは表示されません。内容は API ドキュメントを参照してください。

Java データ型一覧：

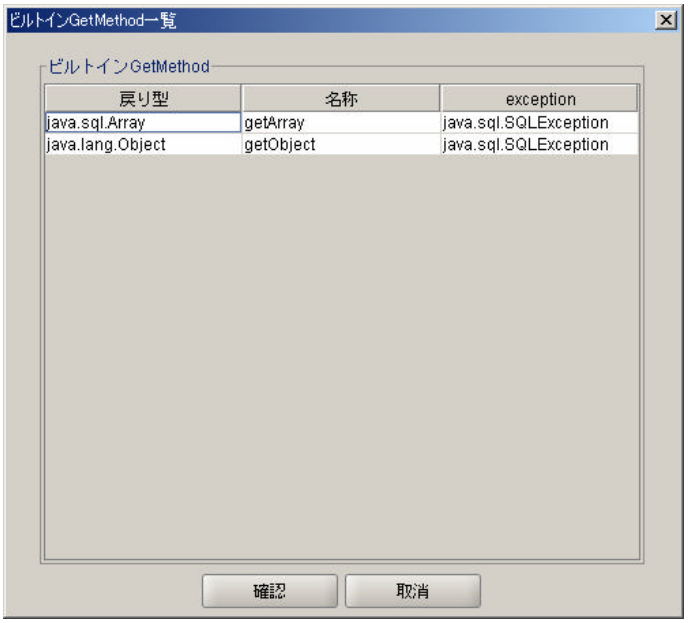
　左下のペインにはサポートしている Java データ型がツリー状に表示されます。setMethod を右クリックしてポップアップメニューから Java データ型を追加することができます。

setMethod 内容：

　右下のペインには選択した setMethod の内容が表示されます。ビルトインメソッド場合は表示されません。

第 2 項 ビルトインメソッドの追加

　JDBC データ型を選択しダブルクリックするとその型をサポートしている変換メソッドが表示されます。右クリックでポップアップメニューを開きビルトインメソッドを追加を選択します。

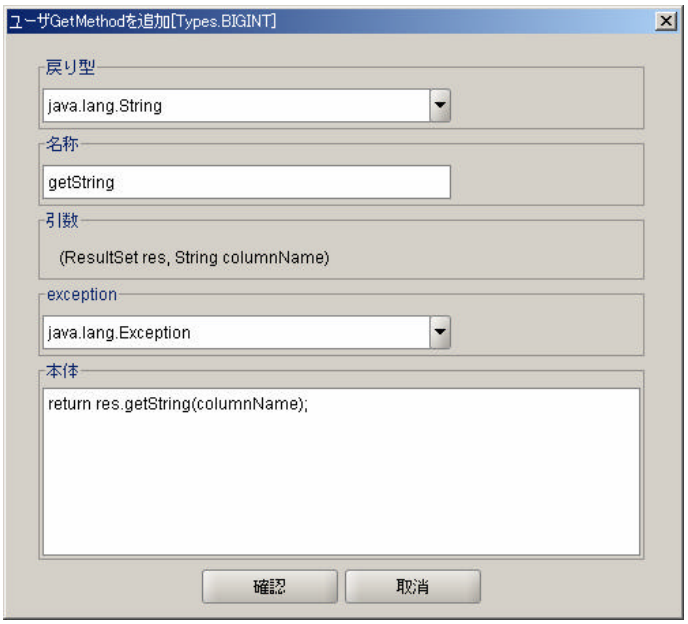


　XJ@IRIS の提供する変換メソッドで JDBC データ型に対応するもののなかからサポートするメソッドを選択して確定ボタンを押します。

　Java データ型に対しても同様にビルトインメソッドを追加することができます。

第 3 項 ユーザメソッド追加

　JDBC の変換メソッドで右クリックしポップアップメニューからユーザメソッドを追加を選択します。



戻り型、名称、exception と内容を入力します。確認ボタンを押すと新しいユーザメソッドが追加されます。

Java データ型に対しても同様にユーザメソッドを追加することができます。

第 4 項 デフォルトの設定と取り消し

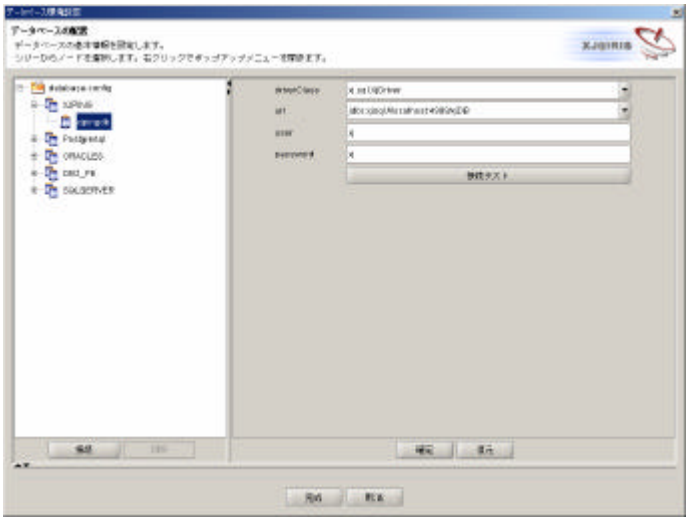
ここで指定するデフォルトget/set メソッドはレコードモデル定義やテーブルウィザードでデフォルトとして使われるメソッドになります。

JDBC データ型の get メソッドあるいは Java データ型の set メソッドを選択します。右クリックからデフォルトにするまたはデフォルトをやめるを選択してデフォルト設定を変更します。

第 13 節 データベース環境設定

XJ@IRIS は xjPing、PostgreSQL、Oracle、DB2、SQLServer などのデータベースをサポートしています。データベース環境設定では JDBC ドライバのパス DataSource の URL、データベースごとのパラメータなどを設定します。

データベースの設定を変更した場合は各ノードごとに確定ボタンを押してください。最後に完成ボタンを押すと変更内容が設定ファイルに保存されます。



第 1 項 データベース接続情報設定

以下のノードで各データベースの接続に必要な情報を設定します。

database-config ノード:

ルートノードです。このノードを選択し、右クリックのポップアップメニューからInterface の新規作成を選択すると新しいデータベースの種類を追加することができます。

データベース種類ノード:

2 段目のノードです。データベースの種類ごとの設定を行います。このノードを選択して右側の編集ペインで JDBC ドライバファイルを追加・削除して設定します。

データベースプロファイルノード:

3 段目のノードです。データベースプロファイルごとのパラメータを設定します。設定すべき値については各データベースのドキュメントを参照してください。接続テストボタンを押すと現在の値で DB サーバーに接続できるかどうか試験することができます。

第 2 項 データベース型設定

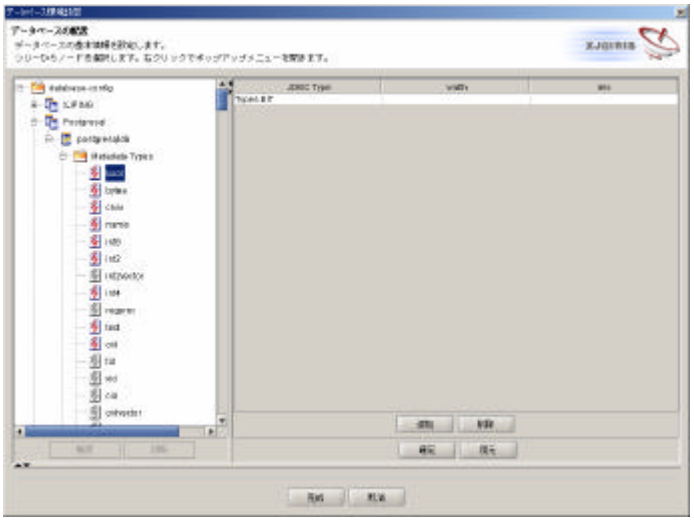
各データベース依存のデータ型の設定をします。

データベースプロファイルノードを選択し下部の接続ボタンを押します。データベースプロファイルのパラメータが正しい場合は接続に成功します。接続状態のときにはデータベースデータ型の設定及びデータベースに存在するテーブルやビューの情報を閲覧することができます。

データベースに接続後、データベースプロファイルノードの子として MetadataTypes と Schemas ノードが表示されます。

MetadataTypes ノード:

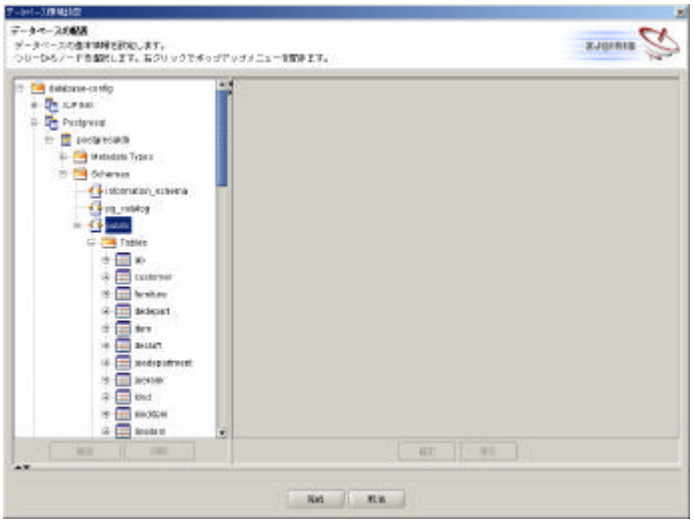
このノードの子ノードとしてデータベースデータ型が表示されます。



データベースデータ型を選択し右クリックからこの データ型をサポートするかどうかを選択することができます。データベースデータ型を選択して右側の編集ペインで対応するJDBC データ型とその width、dec 属性（データベースデータ型によってこれらのパラメータが必要かどうかが決まります。詳しくは各データベースの ドキュメントを参照してください。）が表示されます。追加、削除ボタンで編集することができます。

Schemas ノード：

このノードではデータベースのインスタンス、テーブル ・ビューなどの情報を表示します。インスタンス名で右クリックのポップアップメニューからテーブル ・ビュー一覧を選択します。



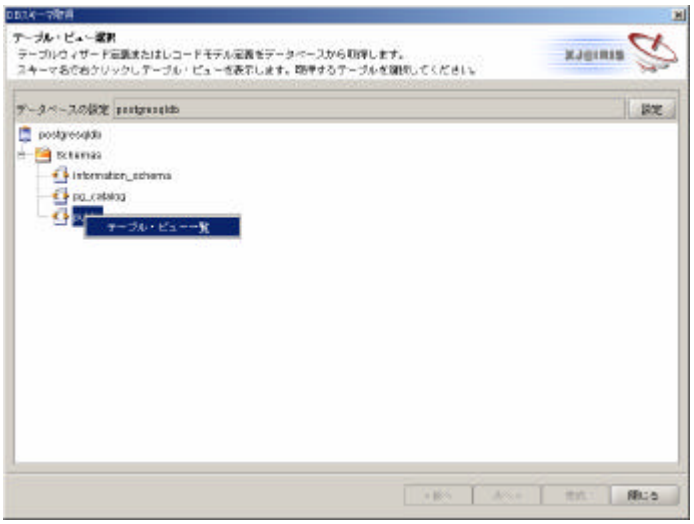
テーブル ・ビュー名を選択して右クリックのポップアップメニューからデータを表示を選択するとそのテーブル内のデータを表示します。

第 14 節　データベーススキーマ取得

データベースに既に存在するテーブルから、XJ@IRIS のテーブルモデルや テーブルウィザードに必要な情報を取得する機能です。

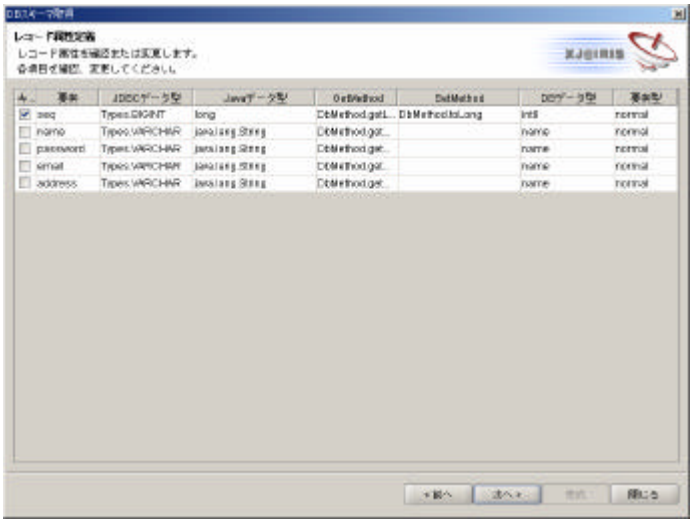
第 1 項 テーブル・ビュー選択

メニューからツール->DB スキーマ取得を選択します。
データベース設定を選択し、ツリーの Schemas ノードの子のインスタンスの 1 つを選択し、右クリックのポップアップメニューからテーブル・ビュー一覧を選択します。



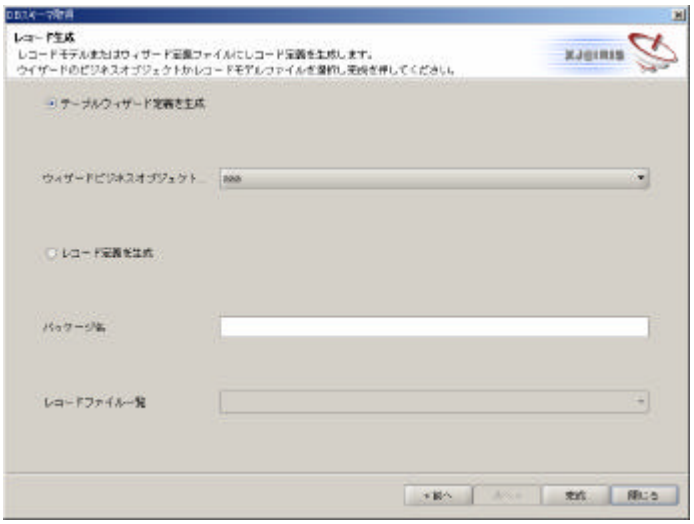
Tables ノードの中のテーブルの中から 1 つを選び次へボタンを押します。

第 2 項 レコード属性設定



選択したテーブルのスキーマが表示されます。Java データ型、GetMethod、SetMethod、要素型はデフォルト値が設定されています。必要ならば変更することができます。確認後次へボタンを押します。

第 3 項 レコード生成



テーブルスキーマ情報をテーブルウィザードに取り込むかレコードモデルに取り込むかを選択します。

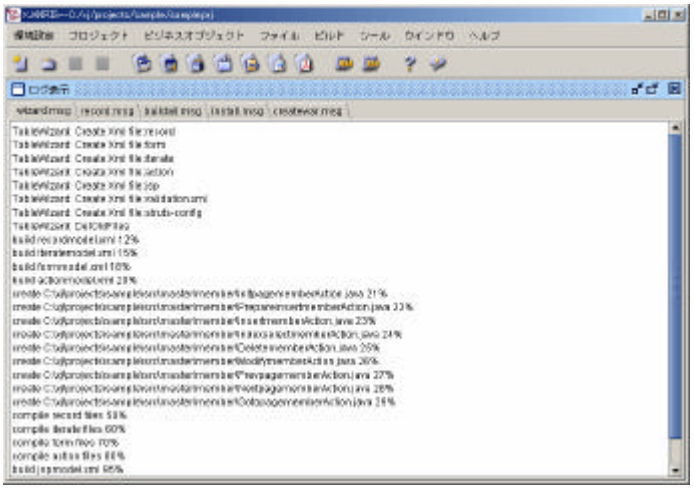
テーブルウィザードの場合はウィザードのビジネスオブジェクト名を選択してください。レコードモデルの場合はファイル名を選択してください。パッケージ名を設定し完成ボタンを押します。

第 4 項 結果表示

取得の結果が表示されます。次へボタンを押して引き続き別のテーブルを作成するか、閉じるを押してください。

第 15 節 ログファイル

メニューからツール->ログ表示を選択します。



このプロジェクトのログファイルがすべて表示されます。ログファイルは XJ@IRIS の機能ごと

に分かれています。ファイルはタブで切り替えます。同じ種類の機能が再度実行されたとき、古いログファイルは上書きされます。

第 16 節 その他

第 1 項 Apache Ant ビルド

XJ@IRIS で作成されたプロジェクトは通常 XJ@IRIS のビルド機能を使って Java ソースファイルのコンパイルを行い War ファイルの生成を行いますが、完成後のプロジェクトの一部ソースを修正した後に XJ@IRIS なしでもコンパイルや War の生成ができるように Apache Ant でのビルドのためのファイルが用意されています。プロジェクトを生成するとプロジェクトのディレクトリに build.xml と build.properties が生成されます。デフォルトのターゲットは War ファイルを生成します。また ant javahelp コマンドでプロジェクトソースファイルから API リファレンスを作成します。

ant コマンドのインストールとセットアップ、環境変数の設定については Apache Ant のドキュメントを参照してください。

http://ant.apache.org/

第 2 項 CVS によるバージョン管理

XJ@IRIS ではプロジェクトを複数の開発者で共有する場合や、バージョンの管理をしたい場合に CVS(Concurrent Versions System)でのバージョン管理を考慮しています。プロジェクト内の model ディレクトリや src、html ディレクトリを cvs 管理にした場合 cvs クライアントはここに CVS という名前のディレクトリを作成しますが、XJ@IRIS ではこのディレクトリを特別な名前として無視しますので動作に影響はありません。cvs に付いては gnu cvs や各種のクライアントツールのドキュメントを参照してください。

http://www.gnu.org/software/cvs/

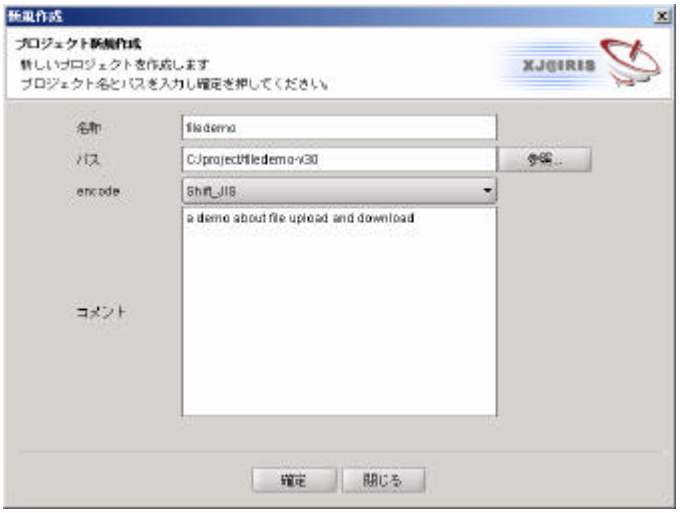
第 6 章　実用的テクニック

第 1 節　ファイルアップロード･ダウンロード

　ここでは、プロジェクトfiledemo を作って、ファイルアップロード･ダウンロードについての実例を説明します。

第 1 項　プロジェクト

　プロジェクトfiledemo を作成します。



第 2 項　formmodel.xml

　formmodel.xml を新規作成します。
　upload の form タグを追加して，このタグの子タグとして、srcname とdstname の comp タグを追加してください。
　srcname は元ファイル名で、dstname はファイルアップロード後のフルパスです。

　それぞれのパラメータは以下の様に入力します。
　model

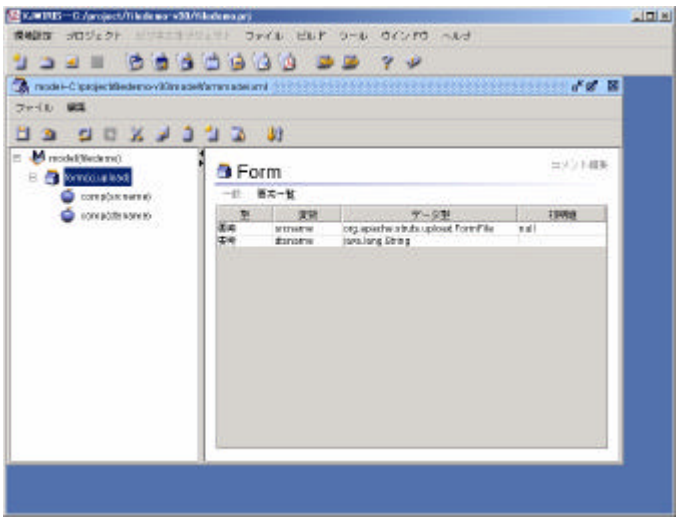
名称	filedemo

form

名称	upload
パッケージ名	xj

comp

名称	srcname	dstname
データ型	org.apache.struts.upload.FormFile	java.lang.String



第 3 項　iteratemodel.xml

iteratemodel.xml を新規作成します。
download の iterate タグを追加して、このタグの子タグとして、name とfullname の comp タグを追加してください。
name はファイル名で、fullname はファイルのフルパスです。

それぞれのパラメータは以下の様に入力します。
model

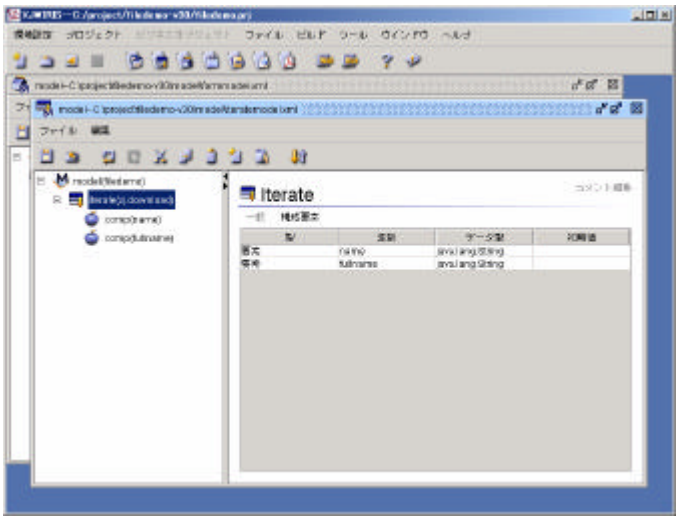
名称	filedemo

iterate

名称	download
パッケージ名	xj

comp

名称	name	fullname
データ型	java.lang.String	java.lang.String



第 4 項　jspmodel.xml

jspmodel.xml を新規作成します。

upload の jsp タグを追加して、jsp タグの子タグとして upload の form タグを追加して、更に、form タグの子タグとして srcname の html:file タグと upload の html:submit タグを追加してください。

download の jsp タグを追加して、jsp タグの子タグとして download の iterate タグを追加して、更に、iterate タグの子タグとして name の bean:write タグと fullname の bean:write タグを追加してください。

それぞれのパラメータは以下の様に入力します。

model

名称	filedemo

jsp

名称	upload	download
タイトル	upload	download

form

名称	upload
name	xj_uploadForm
type	xj.UploadForm
action	/ipload
enctype	multipart/form-data

html:file

名称	srcname
property	srcname

html:submit

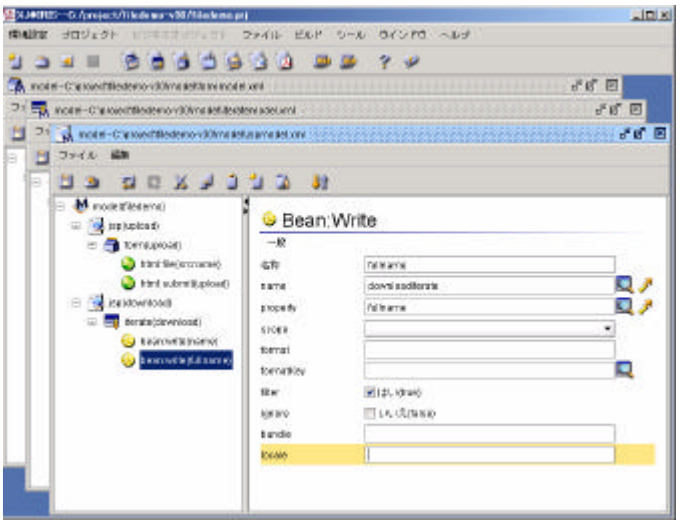
名称	upload

form

名称	download
name	xj_downloadCollection
Id	downloadIterate
type	xj.DownloadIterate

bean:write

名称	name	fullname
name	downloadIterate	downloadIterate
property	name	fullname



メニュー「JSP ファイルをビルド」により最初の download.html を生成した後に、JSP タグのところの「HTML ファイル合成」を使って、download.html を以下の様に編集して下さい。
すなわち
<!--bean:write xjid="downloadJsp.downloadIterate.nameBean:write" -->${bean:write}$
<!--bean:write xjid="downloadJsp.downloadIterate.fullnameBean:write" -->${bean:write}$
を
<a href=
‘<!--bean:write xjid="downloadJsp.downloadIterate.fullnameBean:write"
-->${bean:write}$’>
<!--bean:write xjid="downloadJsp.downloadIterate.nameBean:write" -->${bean:write}$
</br>

に変更してください。

第 5 項　actionmodel.xml

actionmodel.xml を新規作成します。
upload の action タグを追加して、このタグの子タグとして定型文 upload を使うunit タグを追加してください。
download の action タグを追加して、このタグの子タグとして定型文 nothing を使うunit タグを追加してください。

それぞれのパラメータは以下の様に入力します。
model

名称	filedemo

action

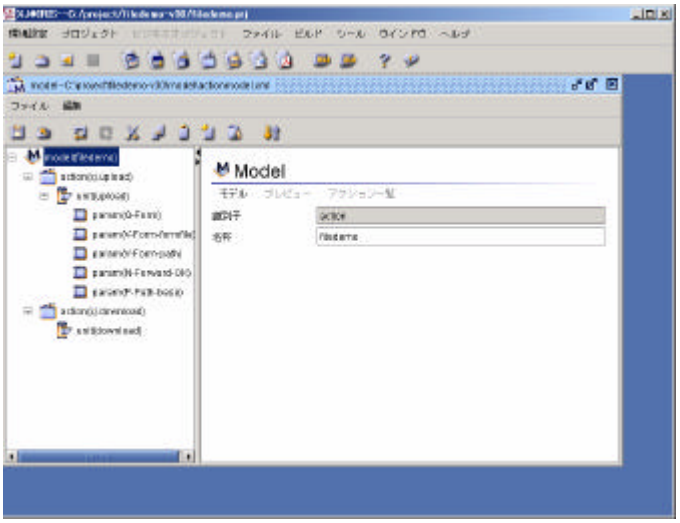
名称	upload	download
パッケージ名	xj	xj

unit

名称	upload	download
型	upload	nothing

param

引数名	Q-Form	V-Form-formfile	V-Form-path
引数値	xj.upload	srcname	dtsname



メニュー「アクションをビルド」により最初の download.java を生成した後に、具体的な download の準備処理のソースを編集してください。ソースを以下に示します。

```
String path = context.getRealPath("/");
File dir = new File(path);
File[] files = dir.listFiles();
XjCollection downloadFiles = new XjCollection();
for ( int i=0; i<files.length; i++ ) {
    if (files[i].isFile()) {
        xj.DownloadIterate downloadIte = new xj.DownloadIterate();
        String name = files[i].getName();
         downloadIte.setname(name);
        downloadIte.setfullname(path + name);
         downloadFiles.add(downloadIte);
        log.info(path + name);
    }
}
session.setAttribute("xj_downloadCollection", downloadFiles);
```

第 6 項　struts-config.xml

struts-config.xml を編集します。

Action-mappings タグの子タグとして upload の action タグと download の action タグを追加してください。

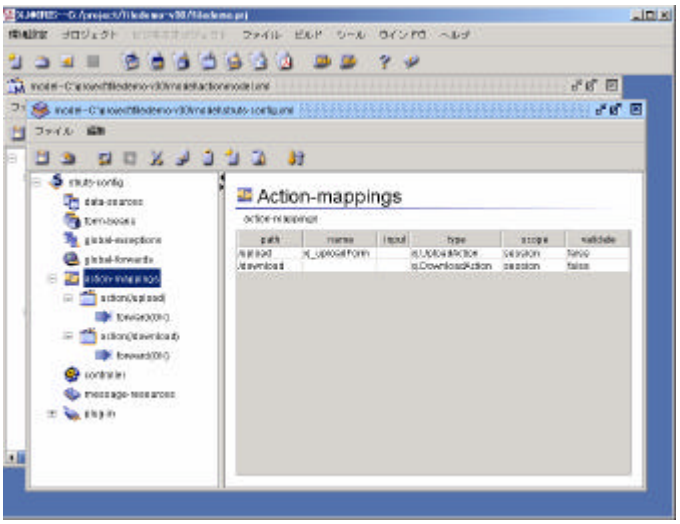
それぞれのパラメータは以下の様に入力します。

action

path	/upload	/download
name	xj_uploadForm	
type	xj.UploadAction	xj.DownloadAction

forward

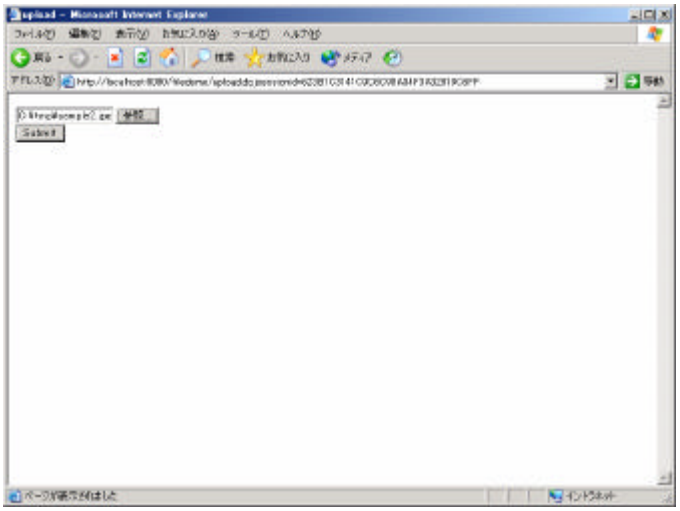
name	OK	OK
path	/upload.jsp	/download.jsp



第 7 項 動作確認

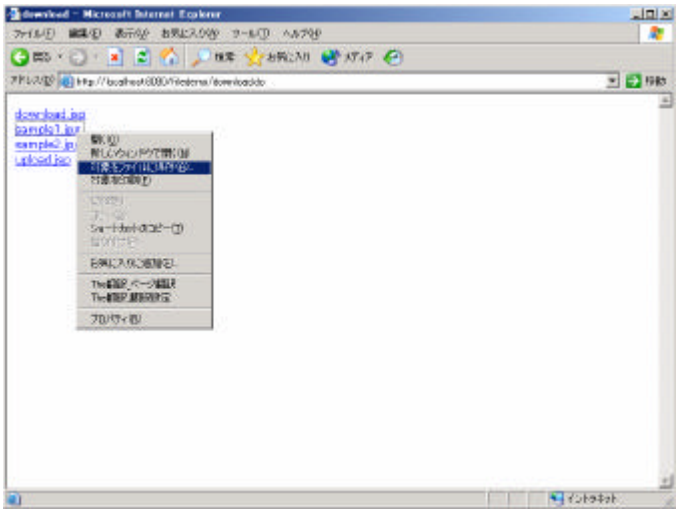
ビルド･WAR 生成 ･インストールして、fildemo の動作を確認します。

IE で http://localhost:8080/filedemo/upload.jsp の url を指定し、以下の画面が表示されます。



ファイルを設定し、確定ボタンを押し、ファイルのアップロードができます。

IE で http://localhost:8080/filedemo/download.do を入力し、以下の画面が表示されます。



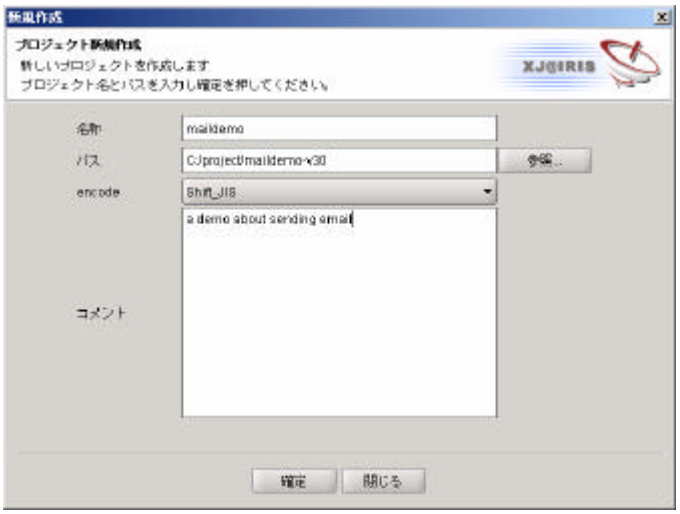
ファイルを選択し、 名前付けて保存」により、ファイルのダウンロードができます。

第 2 節　メール送信

ここでは、プロジェクトmaildemo を作って、メール送信についての実例を説明します。

第 1 項　プロジェクト

プロジェクトmaildemo を作成します。



第 2 項　formmodel.xml

formmodel.xml を新規作成します。

sendemail の form タグを追加して，このタグの子タグとして、mto,mfrom,cc,title,manager,customer,content と smtp の comp タグを追加してください。mto,mfrom,cc は email の基本情報です。title,manager,customer,content は email の内容引数です。smtp はメール送信サーバーのアドレスです。 to と from が予約語ですから、to を mto に、 from を mfrom にします。

それぞれのパラメータは以下の様に入力します。

model

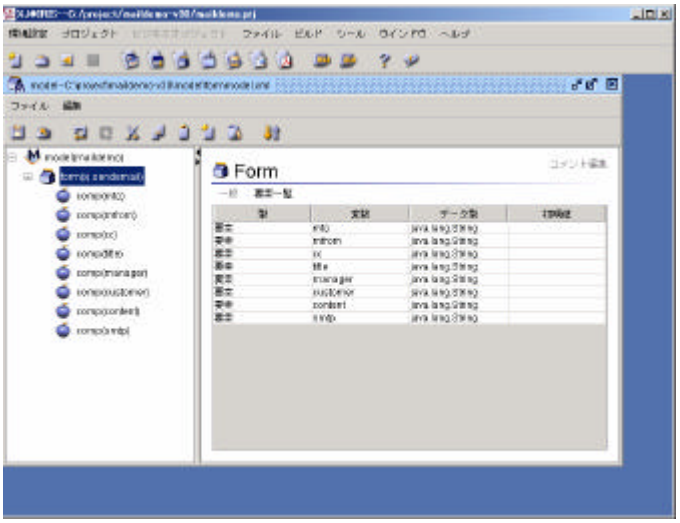
名称	maildemo

form

名称	sendemail
パッケージ名	xj

comp

名称	データ型
mto	java.lang.String
mfrom	java.lang.String
cc	java.lang.String
title	java.lang.String
manager	java.lang.String
customer	java.lang.String
content	java.lang.String
smtp	java.lang.String



第 3 項　jspmodel.xml

jspmodel.xml を新規作成します。

sendemail の jsp タグを追加して、jsp タグの子タグとして sendemail の form タグを追加して、更に、form タグの子タグとして to,from,cc,title,manager,customer,content と smtp の html:text タグと sendmail の html:submit タグを追加してください。

それぞれのパラメータは以下の様に入力します。

model

名称	maildemo

jsp

名称	sendemail
タイトル	send email

form

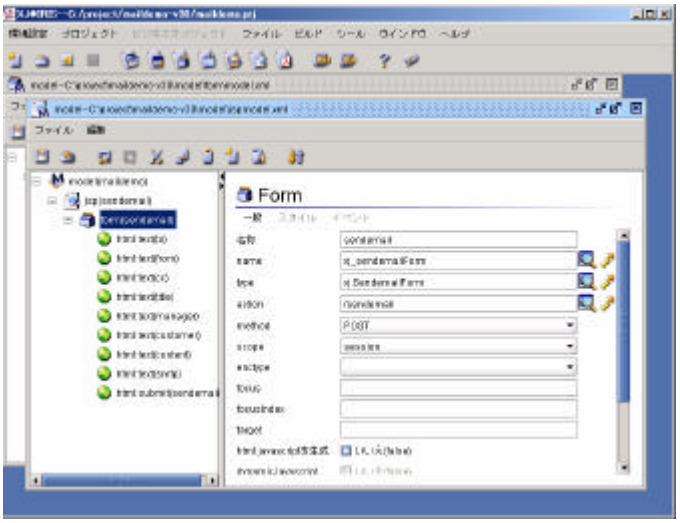
名称	sendemail
name	xj_sendemailForm
type	xj.SendemailForm
action	/sendemail

html:text

名称	property
To	mto
from	mfrom
Cc	cc
title	title
manager	manager
customer	customer
content	content
smtp	smtp

html:submit

名称	sendemail



メニュー「JSP ファイルをビルド」により最初の sendemail.html を生成した後に、入力し易くするため、個々の input の前に名称を追加して下さい。

例えば、

<input type="text" disabled="false" readonly="false" xjid="sendemailJsp.sendemailForm.toHtml:text">

を

to: <input type="text" disabled="false" readonly="false" xjid="sendemailJsp.sendemailForm.toHtml:text">

に変更します。

第 4 項　emailtemplate.xml

emailtemplate.xml を新規作成します。
emailsubject の unit タグを追加して、unit タグの子タグとして title の param タグを追加してください。
emailtextpart の unit タグを追加して、unit タグの子タグとして text とparam タグを追加してください。

それぞれのパラメータは以下の様に入力します。
model

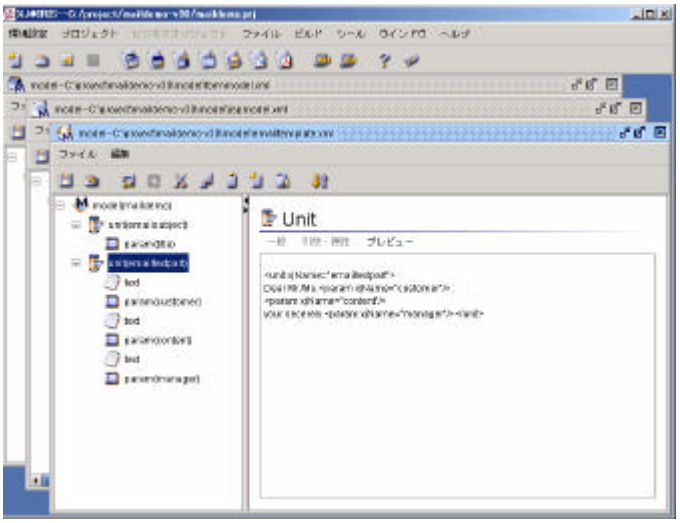
名称	maildemo

unit

名称	emailsubject	emailtextpart

param

名称	title	customer	content	manager



第 5 項　actionmodel.xml

actionmodel.xml を新規作成します。
sendemail の action タグを追加して、このタグの子タグとして定型文 nothing を使うunit タグを追加してください。

それぞれのパラメータは以下の様に入力します。

model

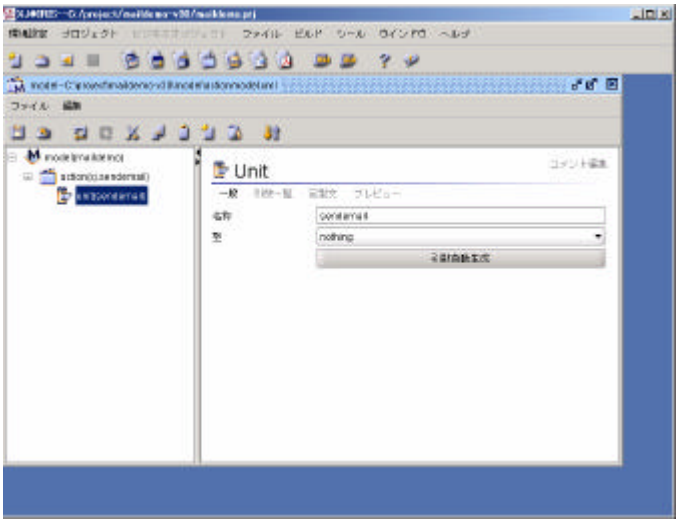
名称	maildemo

action

名称	sendemail
パッケージ名	xj

unit

名称	sendemail
型	nothing



メニュー「アクションをビルド」により最初の sendemail.java を生成した後に、具体的なメール送信の処理のソースを編集してください。ソースを以下に示します。

```
xj.SendemailForm myform = (xj.SendemailForm)form;
try {
    String root = context.getRealPath("/");
    String emailtemplate = root + "WEB-INF/emailtemplate.xml";

    HashMap map = new HashMap();
    map.put("title", myform.gettitle());
    map.put("manager", myform.getmanager());
    map.put("customer", myform.getcustomer());
    map.put("content", myform.getcontent());
    PrepareMail pm = new PrepareMail(emailtemplate, "emailsubject", "emailtextpart", map);

    log.info("subject="+pm.getSubject());
    log.info("content="+pm.getContents());
    log.info("to="+myform.getmto());
    log.info("from="+myform.getmfrom());
```

```
            log.info("cc="+myform.getcc());
            log.info("smtp="+myform.getsmtp());
          SendMail sm = new SendMail(pm.getSubject(),
                                     pm.getContents(),
                                     myform.getmfrom(),
                                     myform.getmto(),
                                     myform.getcc(),
                                     null,//bcc
                                     myform.getsmtp());
            sm.send();
        }catch(Exception e){
            e.printStackTrace();
        }
```

第 6 項　struts-config.xml

struts-config.xml を編集します。
Action-mappings タグの子タグとして sendemail の action タグを追加してください。

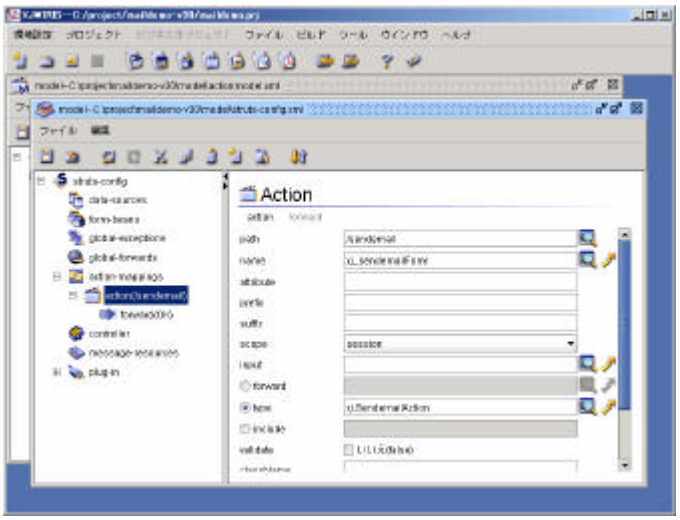
それぞれのパラメータは以下の様に入力します。

action

path	/sendemail
name	xj_sendemailForm
type	xj.SendemailAction

forward

name	OK
path	/sendemail.jsp

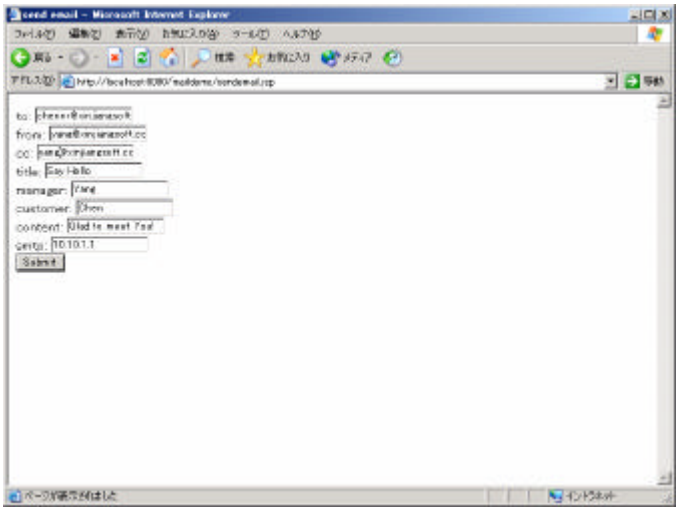


第 7 項 動作確認

ビルド・WAR 編集 ・WAR 生成 ・インストールして、fildemo の動作を確認します。

WAR 編集する時に、emailtemplate.xml を WEB-INF に追加し、javamail に関する jar ファイルを WEB-INF/lib に追加して下さい。

IE で http://localhost:8080/maildemo/sendemail.jsp の url を指定し、以下の画面が表示されます。



確定ボタンを押して、chennr と yang にはそれぞれ以下の中身のメールが届きます。

Dear Ms./Ms Chen:

Glad to meet you!

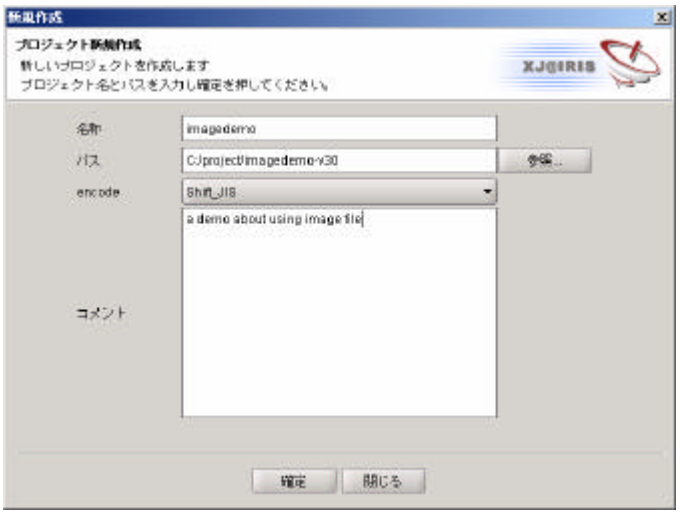
Your secerely Yang

第 3 節　画像ファイル

ここでは、プロジェクトimagedemo を作って、画像ファイルの表示についての実例を説明します。

第 1 項　プロジェクト

プロジェクトimagedemo を作成します。



第 2 項　ウィザード

テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル imagedemo が三つフィールドを持ちます。seq はシーケンスで、imagename は画像名で、imagefile が画像ファイルです。

それぞれのパラメータは以下の様に入力します。

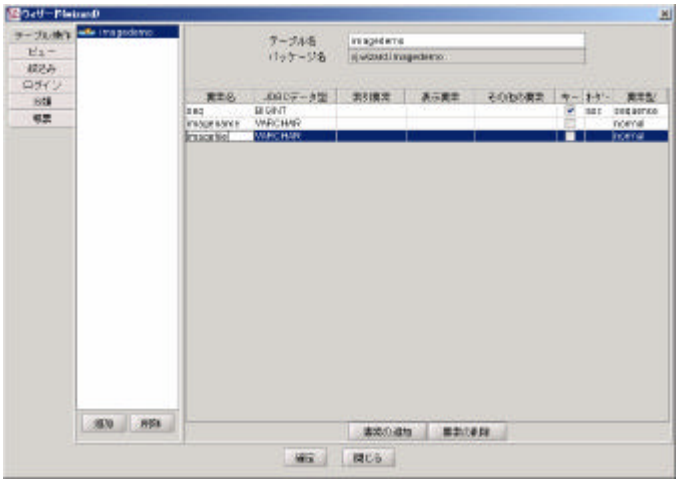
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

テーブルウィザード

テーブル名	imagedemo

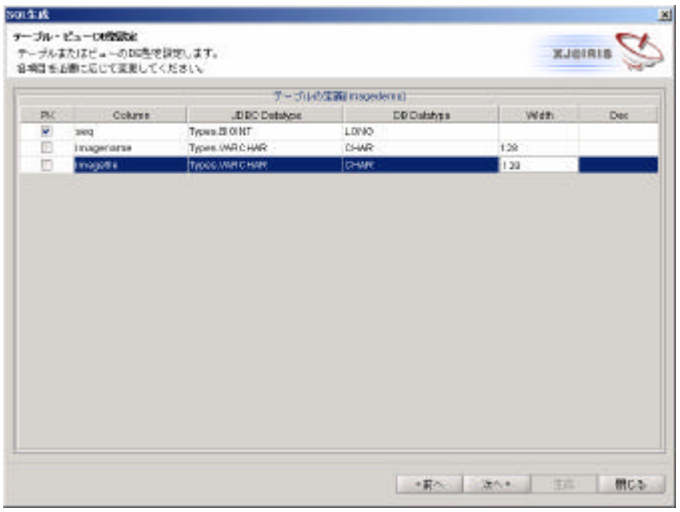
フィールド

seq	BIGINT
imagename	VARCHAR
imagefile	VARCHAR



第 3 項　sql

テーブルウィザードの後に、sql 生成と sql 実行を行いでください。sql 文を生成する時に、imagename とimagefile の width はデフォルトの 32 を 128 に変更して下さい。

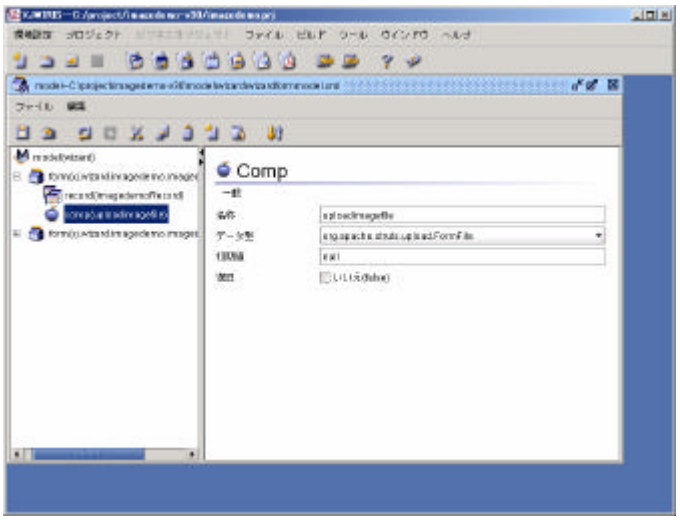


第 4 項　formmodel.xml

formmodel.xml を編集します。
imagedemo の form タグを選択して，このタグの子タグとして、uploadimagefile の comp タグを追加してください。

それぞれのパラメータは以下の様に入力します。
comp

名称	uploadimagefile
データ型	org.apache.struts.upload.FormFile



第 5 項　jspmodel.xml

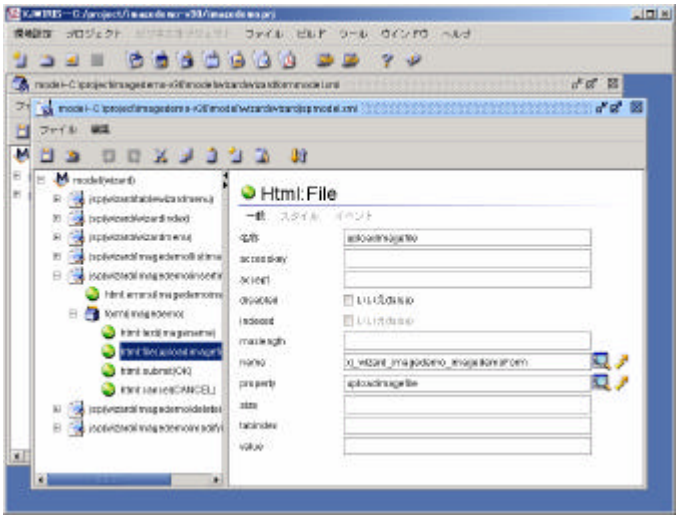
jspmodel.xml を編集します。
insertimagedemo の jsp タグを選択して、jsp タグの子タグの imagedemo の form タグを選択して、form タグの子タグの html:text タグ(imagefile)を html:file タグ(uploadimagefile)に変更してください。

それぞれのパラメータは以下の様に入力します。
form

名称	imagedemo
enctype	multipart/form-data

html:file

名称	uploadimagefile
name	xj_wizard_imagedemo_imagedemoForm
property	uploadimagefile



HTML ファイル合成を使って、以上の変更に対応し、html ファイル中の html:text タグ(imagefile)を html:file タグ(uploadimagefile)に更新してください。

そして、deleteimagedemo の jsp タグを選択して、HTML ファイル合成を使って、bean:write(imagefile) を img の src 属性に変更して下さい。

すなわち、

<!--bean:write xjid="deleteimagedemoJsp.imagedemoForm.imagefileBean:write" -->${bean:write}$

を

<img src=’ <!--bean:write xjid="deleteimagedemoJsp.imagedemoForm.imagefileBean:write" -->${bean:write}$’>

に変更します。

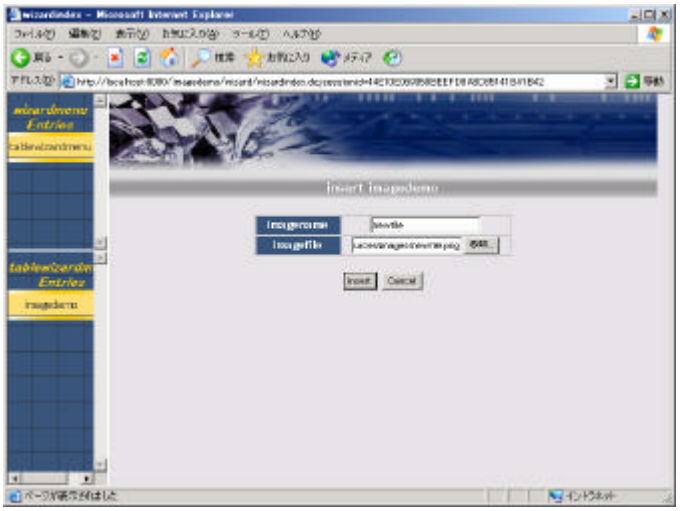
第 6 項 java

画像ファイルのアップロードに対応するため、insertimagedemoAction.java ファイルを編集し、以下のソースを追加します。

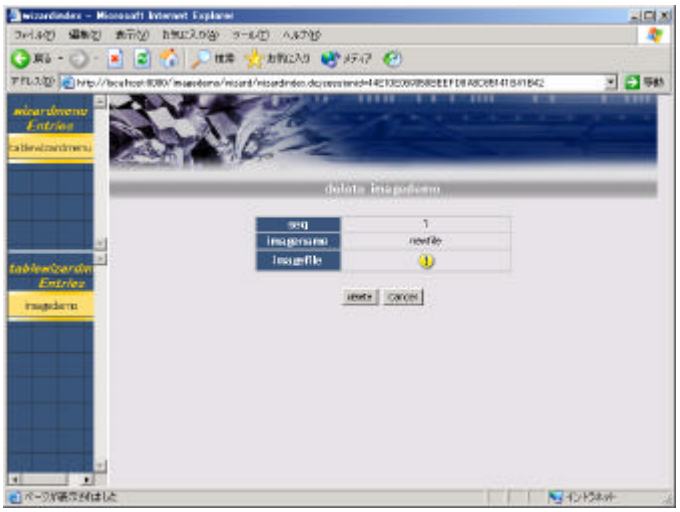
```
// get the file and set the path
 FormFile formfile = myForm.getuploadimagefile();
 String file = formfile.getFileName();
 String path ="context:/";
if( path.startsWith( "context:" ) ) {
     path = context.getRealPath( path.substring( 8 ) );
  }
path = path + "/" + file;

 // open the in/out file
  InputStream   in = formfile.getInputStream();
OutputStream out = new FileOutputStream( path );

 // read/write the in/out file
 int n = 0;
  byte[] buff = new byte[8192];

while( (n = in.read( buff, 0, 8192 )) != -1 ) {
      out.write( buff, 0, n );
  }

 // close the in/out file
  out.close();
  in.close();



  ......
        rcd.setimagefile(path);
  ......
```

第 7 項　動作確認

ビルド・WAR 生成・インストールして、imagemo の動作を確認します。
IE で http://localhost:8080/imagedemo の url を指定したあとに、挿入の画面に入ります。



挿入処理が終わって、削除画面に入り、さきほどの画像が表示されます。

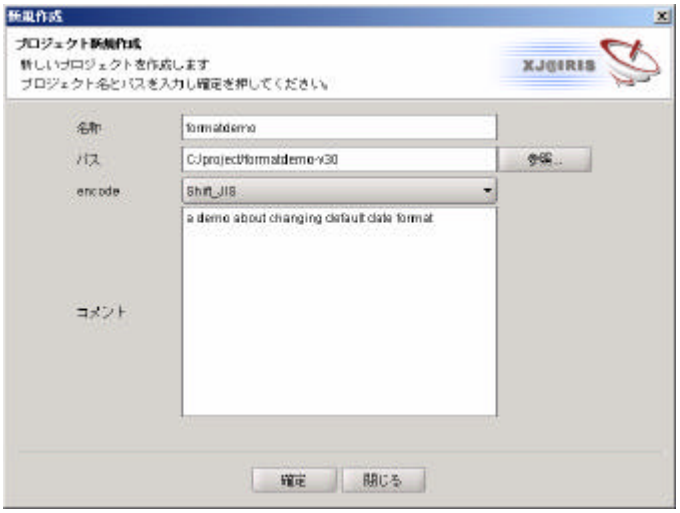


第 4 節　日付フォーマット

ここでは、プロジェクト formatdemo を作って、違うフォーマットの日付の入力と表示についての実例を説明します。

第 1 項　プロジェクト

プロジェクトformatdemo を作成します。



第 2 項　ウィザード

テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル formatdemo が七つフィールドを持ちます。seq はシーケンスです。datehyphen,dateslash,datedot は yyyy-MM-dd, yyyy/MM/dd, yyyy.MM.dd の日付です。stamphyphen,stampslash,stampdot は yyyy-MM-dd hh:mm:ss, yyyy/MM/dd hh:mm:ss, yyyy.MM.dd hh:mm:ss の時間です。

それぞれのパラメータは以下の様に入力します。

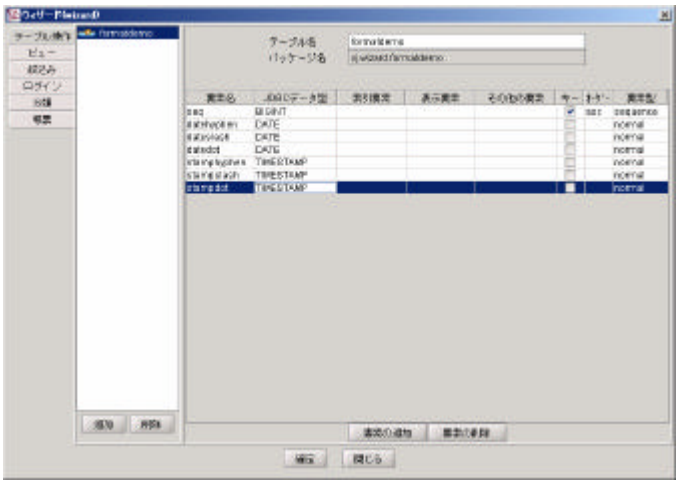
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

テーブルウィザード

テーブル名	formatdemo

フィールド

seq	BIGINT
datehyphen	DATE
dateslash	DATE
datedot	DATE
stamphyphen	TIMESTAMP
stampslash	TIMESTAMP
stampdot	TIMESTAMP



テーブルウィザードの後に、sql 生成と sql 実行を行いで ください。

第 3 項　recordmodel.xml

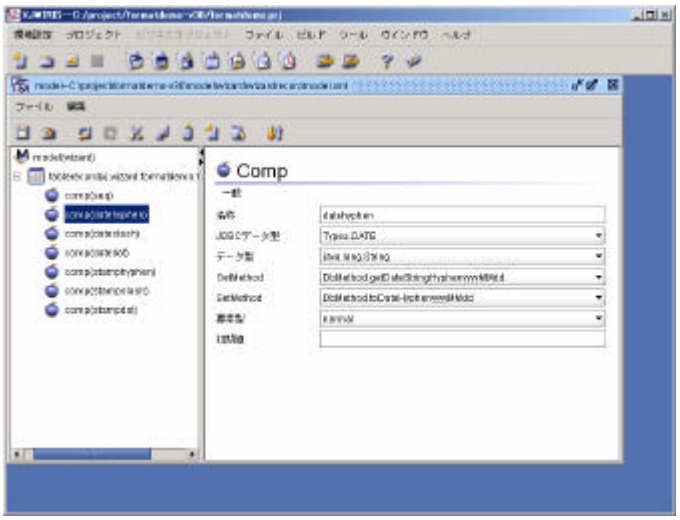
recordmodel.xml を編集します。

formatdemo の tablerecord タグを選択して，このタグの子タグの GetMethod 属性と SetMethod 属性を変更してください。

それぞれのパラメータは以下の様に入力します。

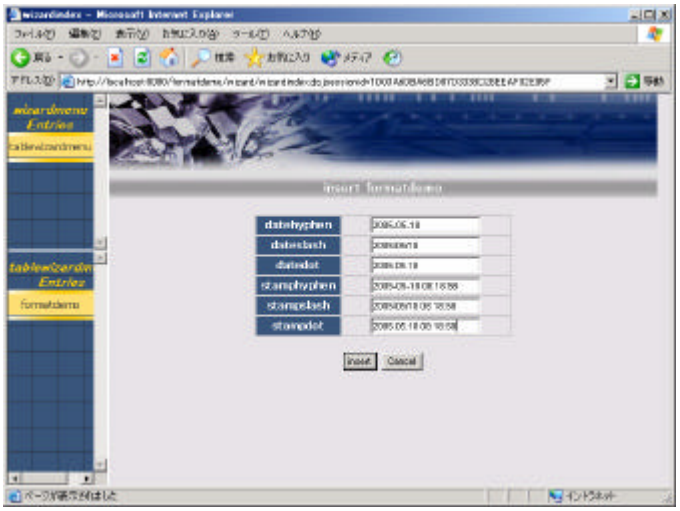
comp

名称	GetMethod	SetMethod
datehyphen	GetMethod.getDateStringHyphenyyyyMMdd	SetMethod.toDateHyphenyyyyMMdd
dateslash	GetMethod.getDateStringSlashyyyyMMdd	SetMethod.toDateSlashyyyyMMdd
datedot	GetMethod.getDateStringDotyyyyMMdd	SetMethod.toDateDotyyyyMMdd
stamphyphen	GetMethod.getTimestampStringHyphenyyyyMMdd	SetMethod.toTimestampHyphenyyyyMMdd
stampslash	GetMethod.getTimestampStringSlashyyyyMMdd	SetMethod.toTimestampSlashyyyyMMdd
stampdot	GetMethod.getTimestampStringDotyyyyMMdd	SetMethod.toTimestampDotyyyyMMdd

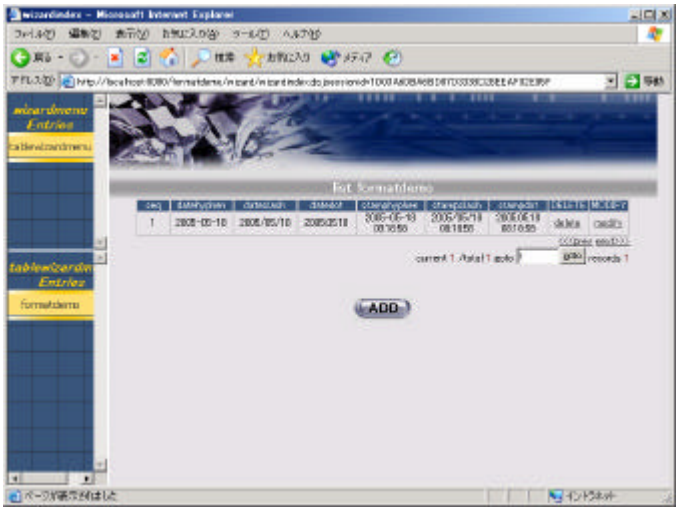


第 4 項　動作確認

ビルド・WAR 生成・インストールして、formatdemo の動作を確認します。
IE で http://localhost:8080/formatdemo を入力した後に、挿入の画面に入ります。



挿入処理が終わって、一覧画面に戻り、さきほどの日付と時間が表示されます。

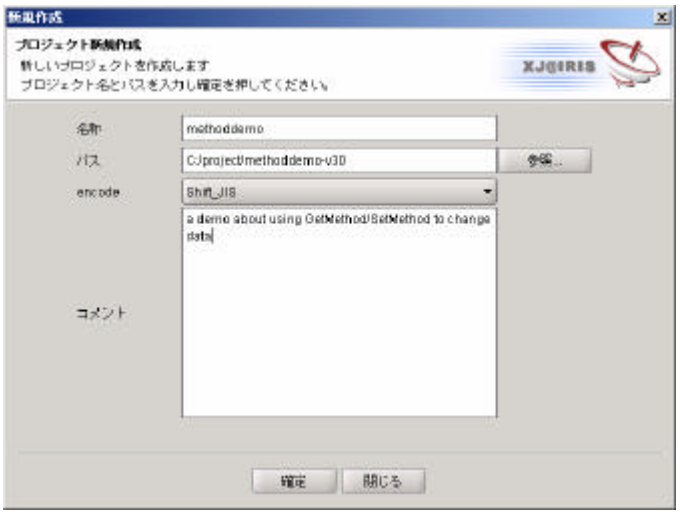


第 5 節　データ変換

ここでは、プロジェクトmethoddemo を作って、ユーザメソッドGetMethod/SetMethod を使って秒のない時間についての実例を説明します。

第 1 項　プロジェクト

プロジェクトmethoddemo を作成します。



第 2 項　ウィザード

テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル methoddemo が三つフィールドを持ちます。seq はシーケンスです。withsecond は秒ありの hh:mm:ss の時間で、nosecond は秒なしの hh:mm の時間です。

それぞれのパラメータは以下の様に入力します。

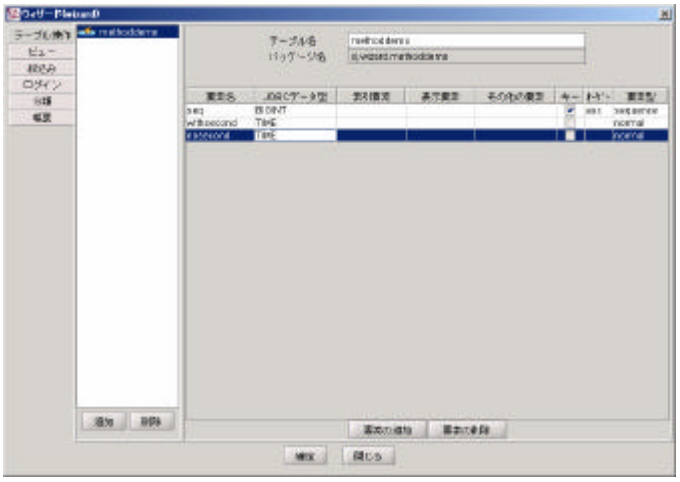
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

テーブルウィザード

テーブル名	methoddemo

フィールド

seq	BIGINT
withsecond	TIME
nosecond	TIME



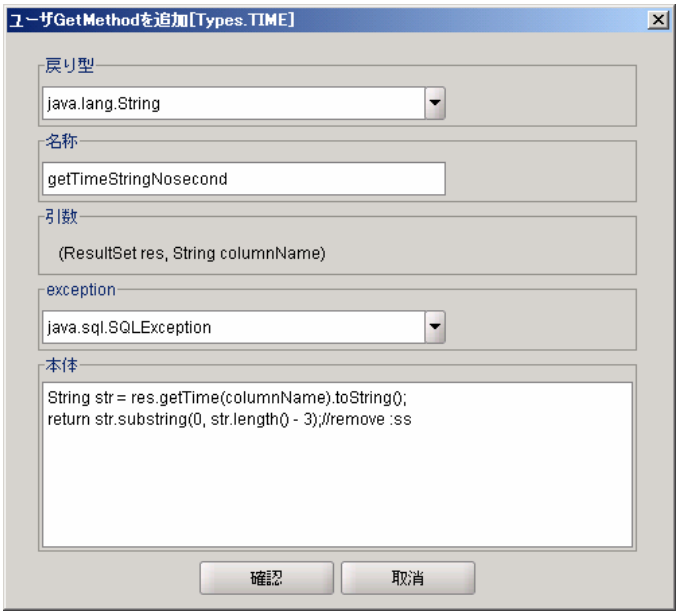
テーブルウィザードの後に、sql 生成と sql 実行を行いで ください。

第 3 項　ユーザメソッド

メニュー「データ変換環境設定」を選択し、ダイアログボックスが表示されます。GetMethod の Types.TIME を選択し、ユーザメソッドを追加します。

それぞれのパラメータは以下の様に入力します。

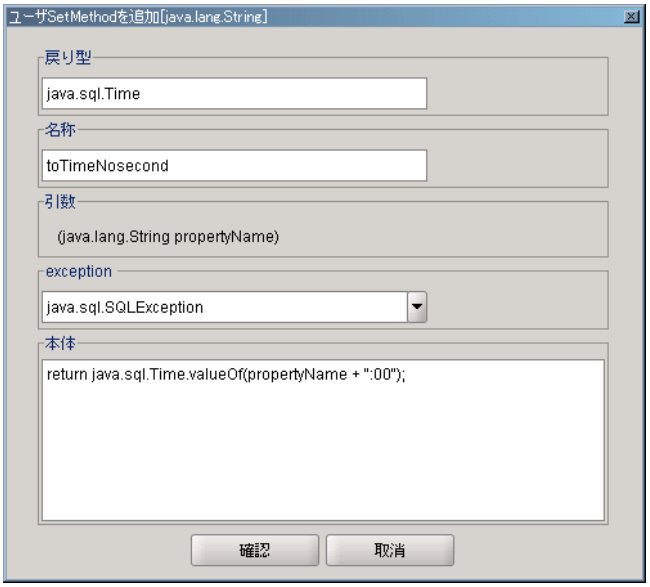
戻り型	java.lang.String
名称	getTimeStringNosecond
exception	java.sql.SQLException
本体	String str = res.getTime(columnName).toString(); return str.substring(0, str.length() - 3);//remove :ss



更に、SetMethod の java.lang.String を選択し、ユーザメソッドを追加します。

それぞれのパラメータは以下の様に入力します。

戻り型	java.sql.Time
名称	toTimeNosecond
exception	java.sql.SQLException
本体	return java.sql.Time.valueOf(propertyName + ":00");



第 4 項　recordmodel.xml

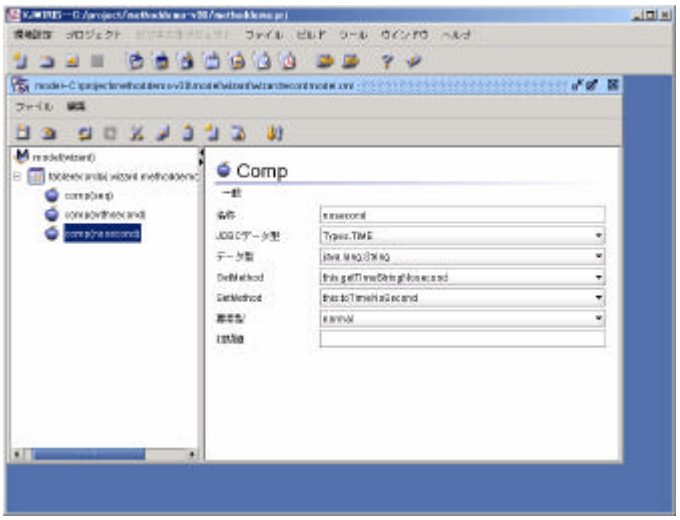
recordmodel.xml を編集します。

methoddemo の tablerecord タグを選択して，このタグの nosecond 子タグの GetMethod 属性と SetMethod 属性を変更してください。

それぞれのパラメータは以下の様に入力します。

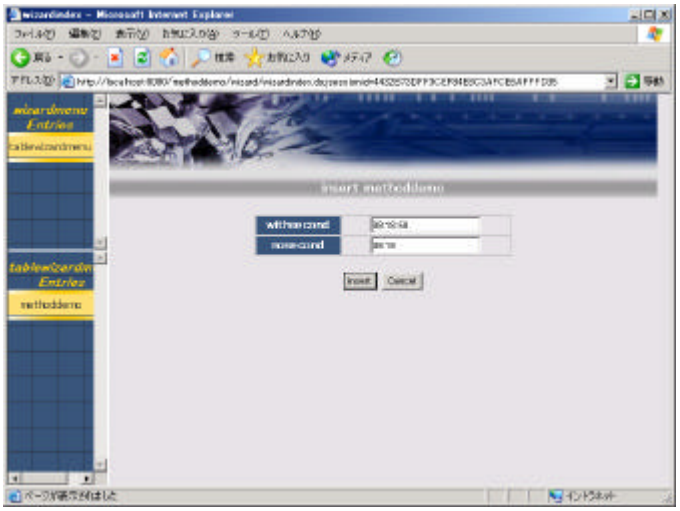
comp

名称	GetMethod	SetMethod
nosecond	this.getTimeStringNosecond	this. toTimeNosecond

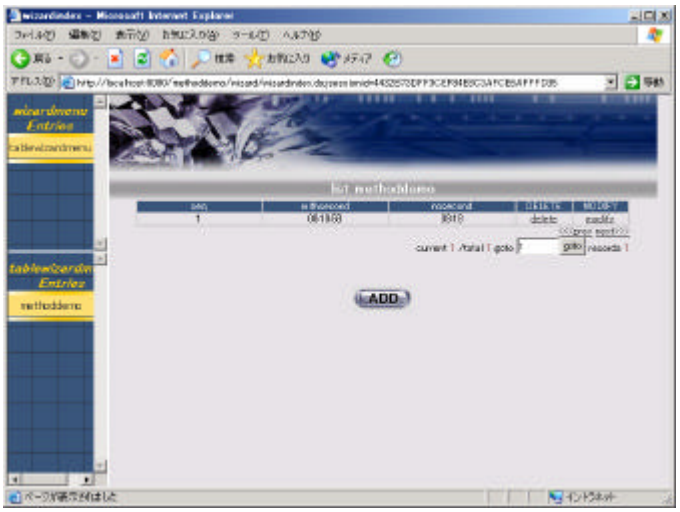


第 5 項 動作確認

ビルド・WAR 生成・インストールして、methoddemo の動作を確認します。
IE で http://localhost:8080/methoddemo の url を指定した後に、挿入の画面に入ります。



挿入処理が終わって、一覧画面に戻り、さきほどの時間が表示されます。

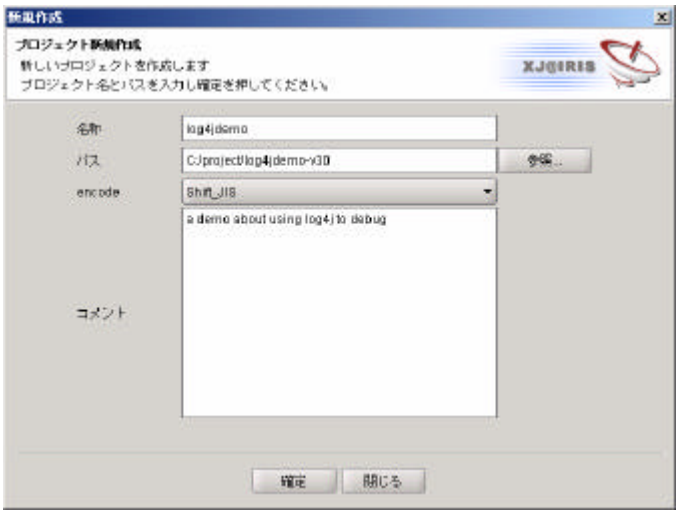


第 6 節　ログ･デバッグ

ここでは、プロジェクトlog4jdemo を作って、log4j を使ってログ･デバッグについての実例を説明します。

第 1 項　プロジェクト

プロジェクトlog4jdemo を作成します。



第 2 項　ウィザード

テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル log4jdemo が二つフィールドを持ちます。seq はシーケンスです。

それぞれのパラメータは以下の様に入力します。

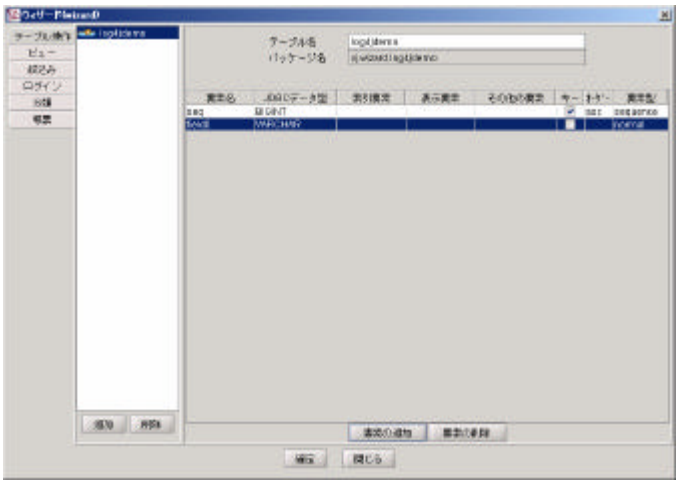
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

テーブルウィザード

テーブル名	log4jdemo

フィールド

seq	BIGINT
field0	VARCHAR



テーブルウィザードの後に、sql 生成と sql 実行を行ってください。

第 3 項　log4j.propertites

テキスト編集ツールで、プロジェクトの model ディレクトリにある log4j.propertites ファイルを編集して下さい。編集後、ファイルの中身は以下のようになります。

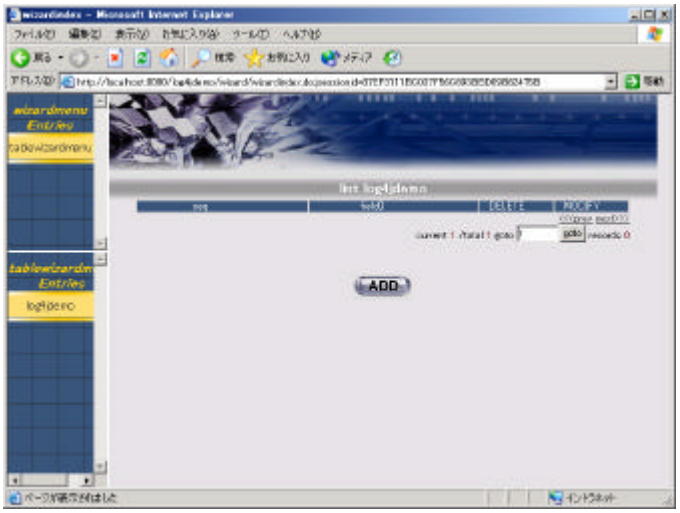
```
log4j.rootLogger=WARN, STDOUT, LOGFILE
log4j.logger.xj.wizard.log4jdemo=INFO

log4j.appender.STDOUT=org.apache.log4j.ConsoleAppender
log4j.appender.STDOUT.layout=org.apache.log4j.PatternLayout
log4j.appender.STDOUT.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

log4j.appender.LOGFILE=org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE.File=C:/log/log4jdemo.log
log4j.appender.LOGFILE.MaxFileSize=100KB
log4j.appender.LOGFILE.MaxBackupIndex=1
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n
```

そして、ログ情報は標準出力とログファイル C:/log/log4jdemo.log 両方に出力します。C ドライブに log というディレクトリを先に作って下さい。

log4j.logger.xj.wizard.log4jdemo=INFO の設定により、xj.wizard.log4jdemo のソース中 INFO 以上のログ情報が出力されます。

第 4 項 動作確認

ビルド・WAR 生成・インストールして、log4jdemo の動作を確認します。
IE で http://localhost:8080/log4jdemo の url を指定した後に、一覧の画面に入ります。



そして、標準出力とログファイルでは以下のログ情報が見られます。

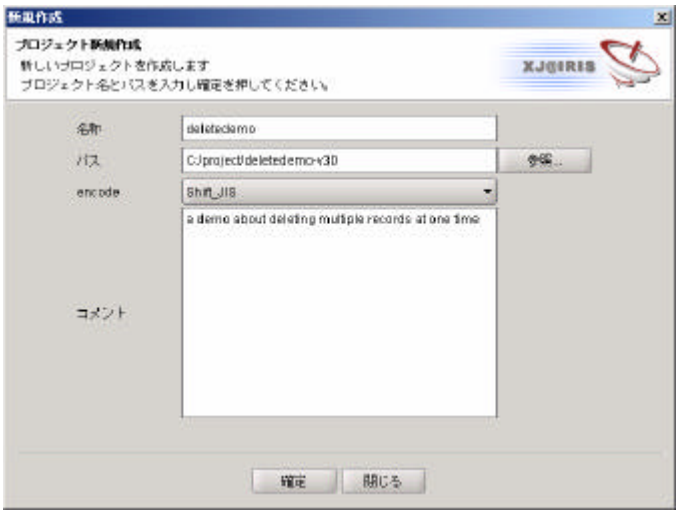
0 INFO [http-8080-Processor23] xj.wizard.log4jdemo.InitpagelogdjdemoAction - in initPage
1833 INFO [http-8080-Processor23] xj.wizard.log4jdemo.InitpagelogdjdemoAction - out initPage

第 7 節　一括削除

ここでは、プロジェクトdeletedemo を作って、一括削除についての実例を説明します。

第 1 項　プロジェクト

プロジェクトdeletedemo を作成します。



第 2 項　ウィザード

テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル deletedemo が二つフィールドを持ちます。seq はシーケンスです。

それぞれのパラメータは以下の様に入力します。

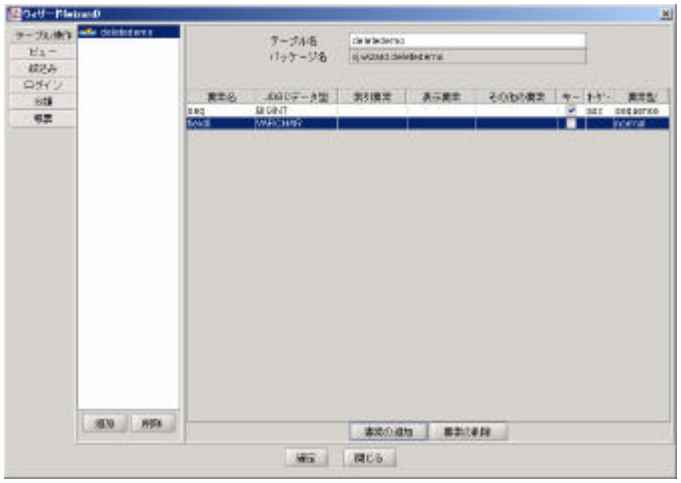
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

テーブルウィザード

テーブル名	deletedemo

フィールド

seq	BIGINT
field0	VARCHAR



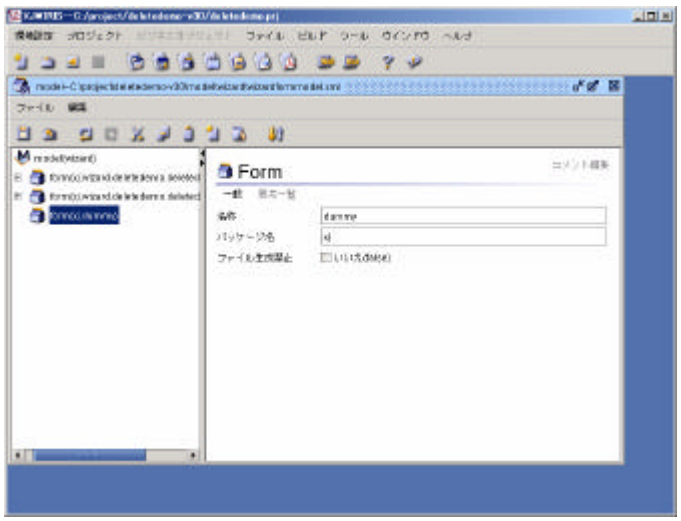
テーブルウィザードの後に、sql 生成と sql 実行を行いで ください。

第 3 項　formmodel.xml

formmodel.xml を編集します。
一個 dummy の form タグを追加してください。

それぞれのパラメータは以下の様に入力します。
form

名称	dummy
パッケージ名	xj



第 4 項　jspmodel.xml

jspmodel.xml を編集します。

listdeletedemo の jsp タグを選択して，このタグの子タグとして delete の form タグを追加して下さい。

それぞれのパラメータは以下の様に入力します。

form

名称	delete
name	xj_dummyForm
type	xj.DummyForm
action	/multipledelete

本来は jsp タグの子タグの iterate タグを新規追加した delete の form タグの下に移動します。この中の delete の html:link タグを html:checkbox タグに変更します。delete の html:submit タグも追加して下さい。

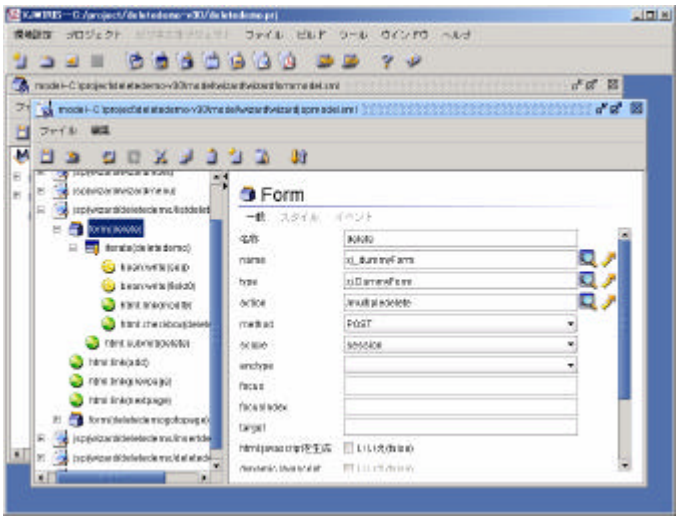
それぞれのパラメータは以下の様に入力します。

html:checkbox

名称	delete
indexed	true
name	deletedemoIterate
property	deletedemoRecord.seq

html:submit

名称	delete
value	delete



HTML ファイル合成で、先ほどの変更に対応するように編集します。すなわち、form を追加して、iterate と子タグを更新して、html:submit も追加します。

第 5 項　actionmodel.xml

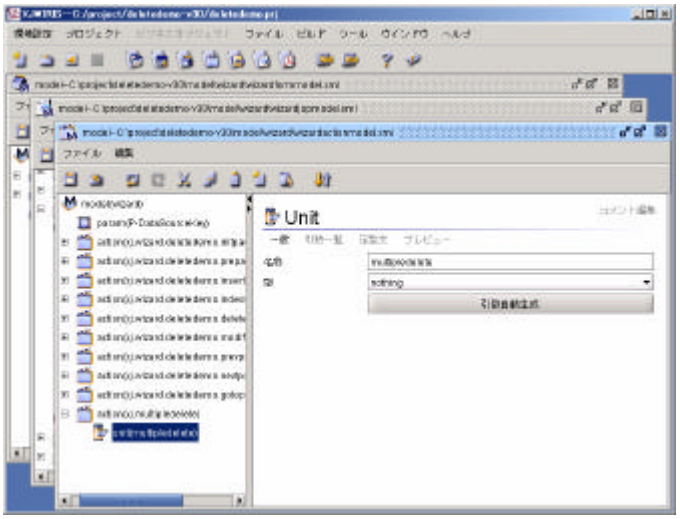
actionmodel.xml を編集します。
multipledelete の action タグを追加して、このタグの子タグとして定型文 nothing を使うunit タグを追加してください。

それぞれのパラメータは以下の様に入力します。
action

名称	multipledelete
パッケージ名	xj

unit

名称	multipledelete
型	nothing



メニュー「アクションをビルド」により最初の multipledelete.java を生成した後に、具体的な一括削除の処理のソースを編集してください。ソースを以下に示します。

```
// get dbConnector first, if not exist then create it
 dataSource = getDataSource(request, "wizard");
 con = Utility.getDbConnector(session, dataSource, "wizard");

// delete the record in table
 DbTable tbl = null;
```

```
        try {
            DbLock.lock(con, "deletedemo");

            tbl = new DbTable(con, "deletedemo",
xj.wizard.deletedemo.DeletedemoRecord.class);

            // get the collection
            XjCollection clt =
(XjCollection)session.getAttribute("xj_wizard_deletedemo_deletedemoCollection");

            for( int i = 0; i < clt.size(); i++ ) {
                if( request.getParameter("deletedemoIterate[" + i +
"].deletedemoRecord.seq") != null ) {
                    xj.wizard.deletedemo.DeletedemoRecord rcd =
                        ((xj.wizard.deletedemo.DeletedemoIterate)clt.get(i))
                        .getdeletedemoRecord();
                    tbl.delete( rcd );
                }
            }
            con.commit();

            DbLock.unlock(con, "deletedemo");

            tbl.close();
        }catch(Exception e){
            con.rollback();

            DbLock.unlock(con, "deletedemo");

            if (tbl != null) tbl.close();

            messages.add("sqlerror", new ActionMessage("errors.bmvcd.printf",
e.toString()));
            saveErrors(request, messages);

            log.error("deleteTable", e);

            return (mapping.findForward("NG"));
        }
```

第 6 項　struts-config.xml

struts-config.xml を編集します。
Action-mappings タグの子タグとして multipledelete の action タグを追加してください。

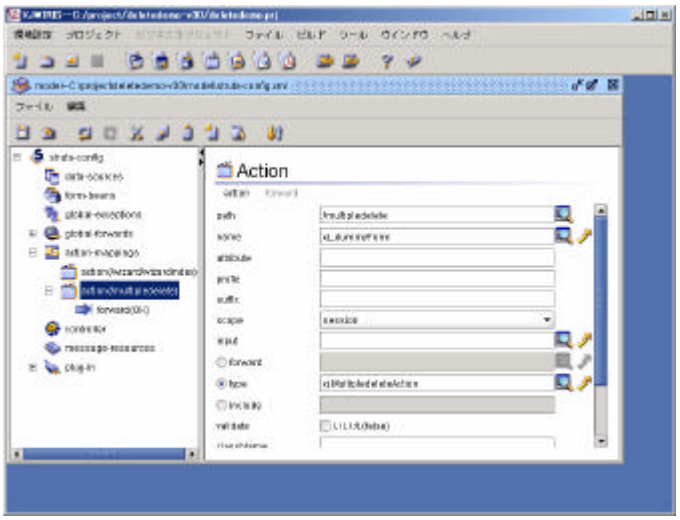
それぞれのパラメータは以下の様に入力します。

action

path	/multipledelete
name	xj_dummyForm
type	xj.MultipledeleteAction

forward

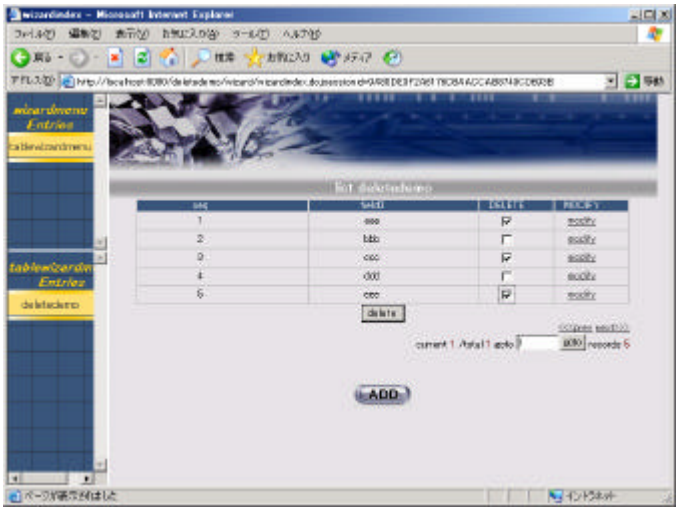
name	OK
path	/wizard/deletedemo/initpagedeletedemo.do



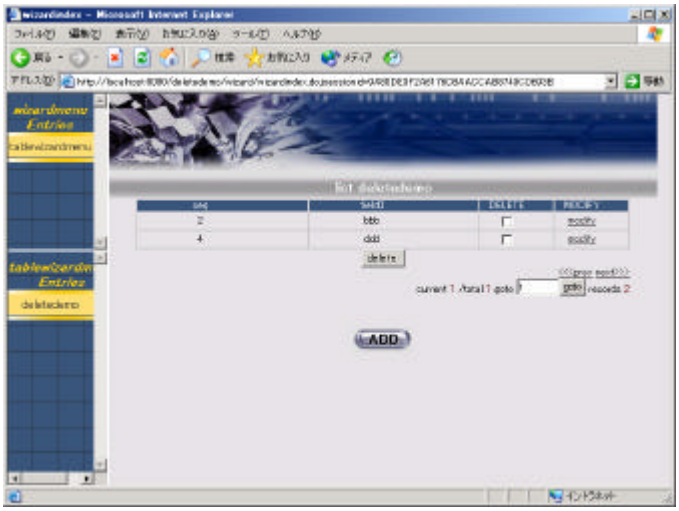
第 7 項　動作確認

ビルド・WAR 生成・インストールして、log4jdemo の動作を確認します。

IE で http://localhost:8080/deletedemo の url を指定した後に、いくつかのデータを挿入しておきます。



チェックボックスを選択し、削除ボタンを押して、選択されたデータは一括で削除されます。

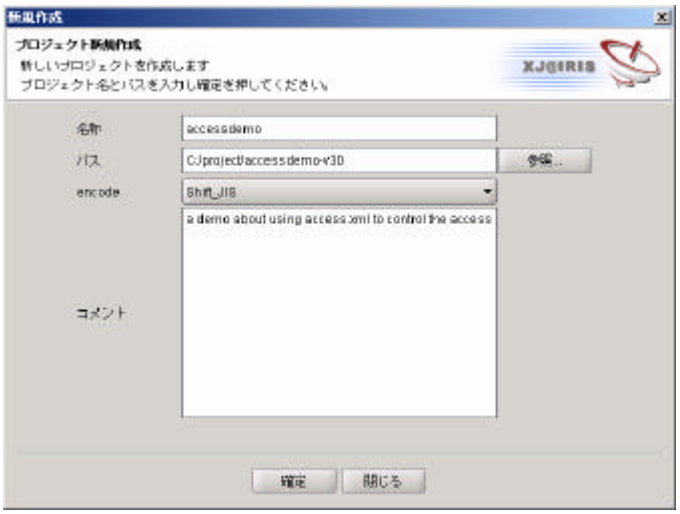


第 8 節　アクセス制御

　ここでは、プロジェクトaccessdemo を作って、ログインと未ログインユーザのアプリケーションへのアクセス制御についての実例を説明します。

第 1 項　プロジェクト

　プロジェクトaccessdemo を作成します。



第 2 項　ウィザード

　テーブルウィザードを使って、プロジェクトの基礎を作ります。テーブル accessdemo が三つフィールドを持ちます。seq はシーケンスです。name とpassword はユーザ情報です。

　それぞれのパラメータは以下の様に入力します。

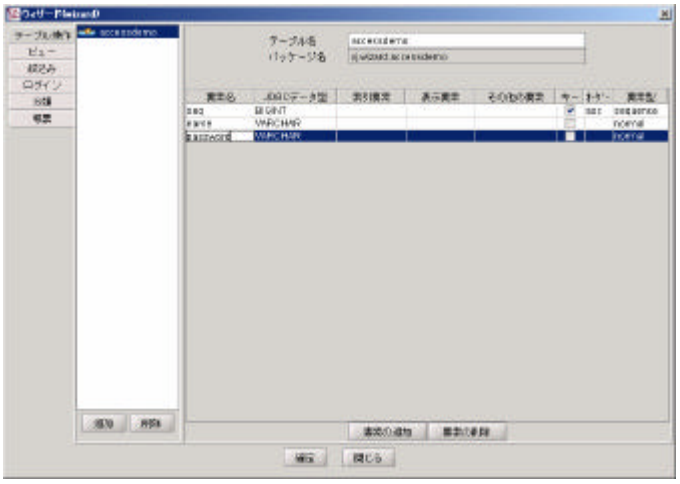
ビジネスオブジェクト名	wizard
パッケージプレフィックス	xj

　テーブルウィザード

テーブル名	accessdemo

フィールド

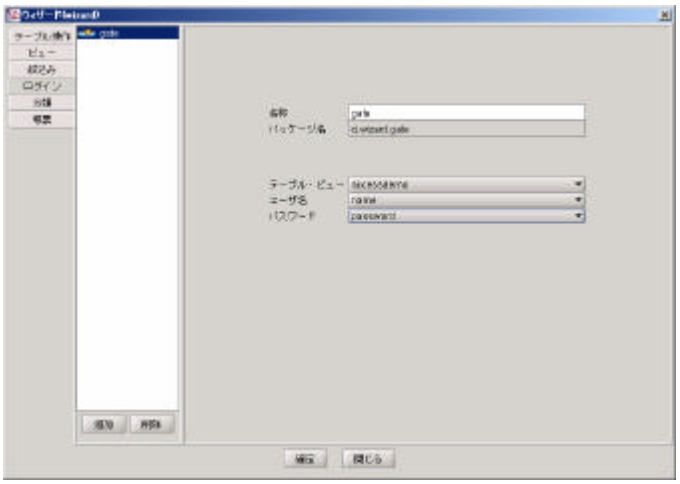
seq	BIGINT
name	VARCHAR
password	VARCHAR



そして、ログインウィザードで gate を作成します。

ログインウィザード

名称	gate
テーブル・ビュー	accessdemo
ユーザ名	name
パスワード	password



第 3 項　データの準備

ここで、もしaccessdemo テーブルやレコードにデータが存在しない場合は SQL の生成、実行とこのプロジェクトの web アプリケーションなどを使ってデータを用意します。

ここでは以下のレコードを用意したものとします。

accessdemo テーブル

seq	1
name	admin
password	xj

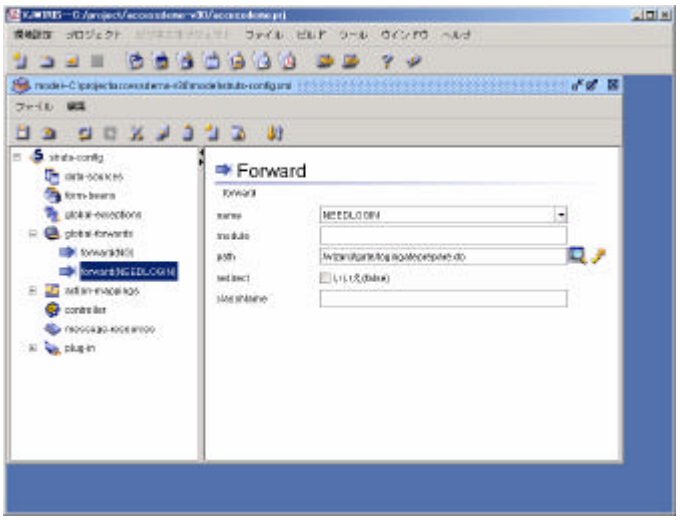
第 4 項　struts-config.xml

struts-config.xml を編集します。

global-forwards タグの子タグとして NEEDLOGIN の forward タグを追加します。

それぞれのパラメータは以下の様に入力します。

name	NEEDLOGIN
path	/wizard/gate/logingateprepare.do

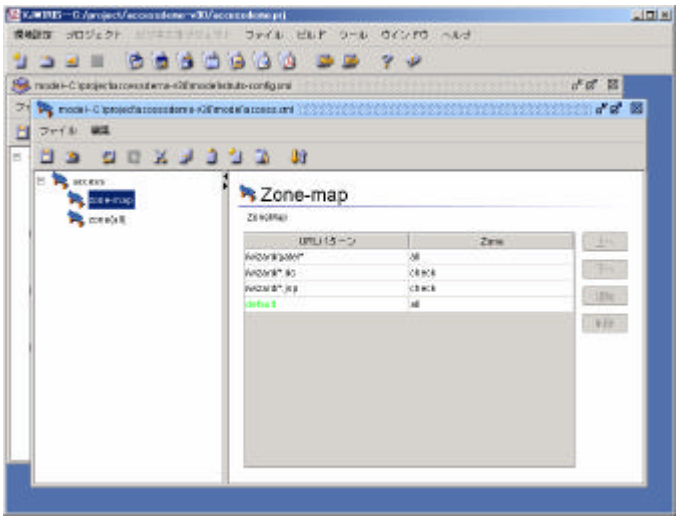


第 5 項　access.xml

access.xml を編集します。

zone-map タグを選択し値を次のように変更します。

URL パターン	Zone
/wizard/gate/*	all
/wizard/*.do	check
/wizard/*.jsp	check
default	all



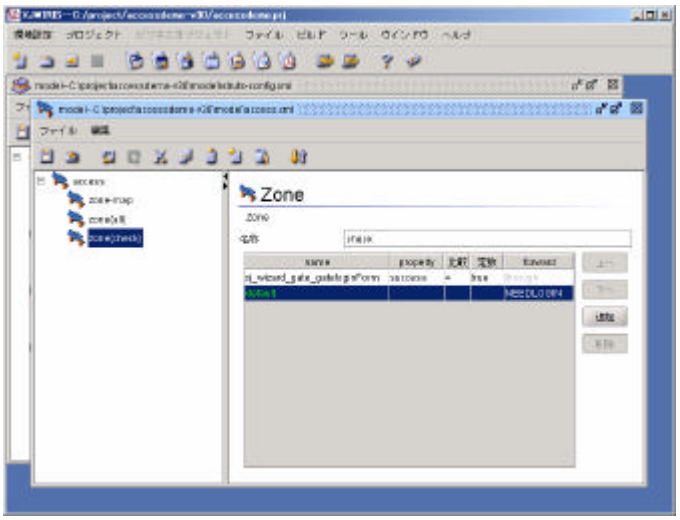
さらに access タグの子として zone タグを追加し値を次のように変更します。

名称	check

name	property	比較	定数	forward
Xj_wizard_gate_gateloginForm	success	=	true	
default				NEEDLOGIN

forward は何も入力しないとthrough と表示されます。この式が成り立てばユーザの要求した url を表示するという意味になります。

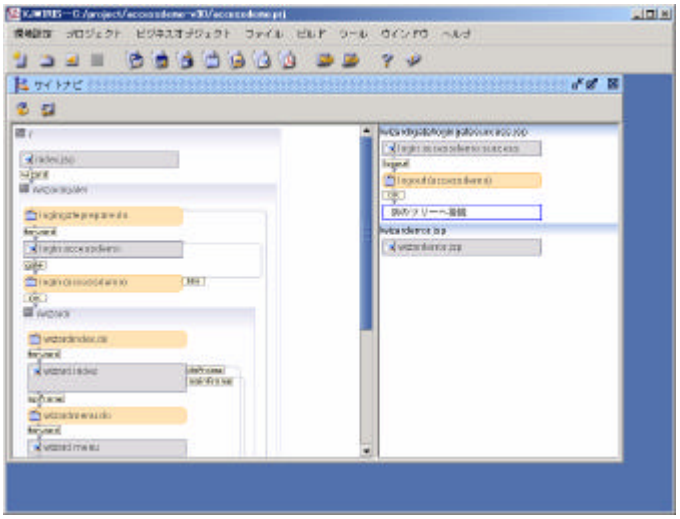
最後の default の行の forward には NEEDLOGIN を入力してください。



第 6 項　遷移の変更

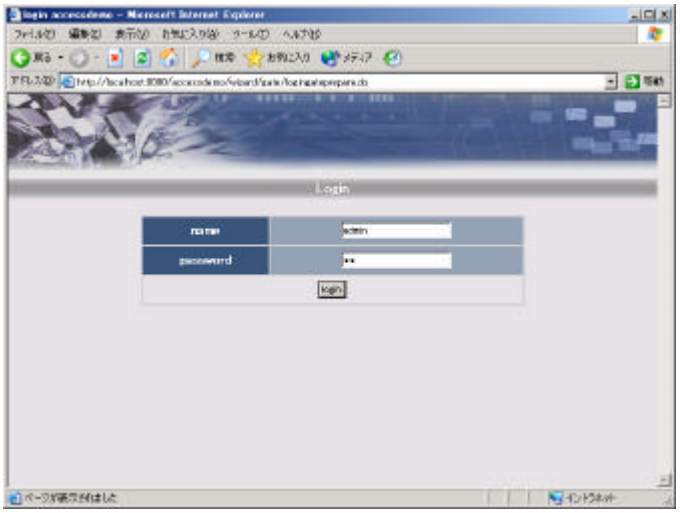
サイトナビを使ってページ遷移を変更します。

遷移元	遷移先
/index.jsp の wizard	/wizard/gate/logingateprepare.do
/wizard/gate/logingatecheck.do の OK	/wizard/wizardindex.do



第 7 項 動作確認

ビルド・WAR 生成・インストールして、accessdemo の動作を確認します。
IE で http://localhost:8080/accessdemo の url を指定します。wizard のリンクでログイン画面が表示されます。正しくユーザ名、パスワードを入力すると wizard のメニュー画面が表示されます。仮にユーザがログイン画面を経由せず/wizard/wizardindex.do などの url を直接指定した場合には、セッションにログインフォームが存在しないため強制的にログイン画面に遷移します。



付録

デフォルトのアクションテンプレート

XJ@IRIS でプロジェクト生成時にデフォルトのアクションテンプレートが作成されます。この中に含まれるタイプについて説明します。
1 つのアクションは head、init、xxx、tail の 4つの部分から構成されます。head、init、tail の 3つの部分は固定部分です。xxx が可変部でこのアクションのタイプになります。

model/actiontemplate.xml で提供するアクションタイプには次の種類があります。

定型文名	定型文の機能概要
login	Form 中の username と password 属性を使って、テーブル中の username と password フィールドと比較を行います。同時に Form 中のレコードに検索したレコードを保存します。
forwardRequestPath	セッション中に保存したユーザリクエストパスがある場合は、forward を書き換えます。AccessFilter でパスが保存されます。
logout	現在のセッションを切断します。
insertTable	トークンが有効の場合 Form 中のレコードをテーブルに insert します。
updateTable	Form 中のレコードを update します。
deleteTable	Form 中のレコードをテーブルから delete します。
deleteTableFromView	Form のビュー要素の中のレコードをテーブルから削除します。
getRecordFromIterateByIndex	index の値によってセッション中に保存されたテーブルの配列（collection）から1つのレコードを取り出し、別のフォームの中に保存します。
getTableFromIterateByIndex	index の値によってセッション中に保存されたテーブルの配列（collection）から1つのビューの中のレコードを取り出し、別のフォームの中に保存します。
initPage	データベースから1ページ分のレコードを取得し、セッション中の配列（collection）に保存します。同時に次のページへ移動をサポートします。レコード数の多いテーブルに適します。
nextPage	次ページへ移動します。initPage と連携します。
prevPage	前ページへ移動します。initPage と連携します。
gotoPage	指定ページへ移動します。initPage と連携します。
prepareSelectCollection	データベースからすべてのレコードを一度に取得し、セッシ

	ョン中の配列（collection）に保存します。レコード数の少ないテーブルに適します。
createPdfFromDB	データベースの中からデータを得て、PDF モデルファイルの定義によりPDF ファイルを出力します。
nothing	空の操作です。ユーザはこのタイプを指定して、独自の業務処理に書き換えます。
saveToken	saveToken()を使ってトークンを取得します。
upload	JSPの中html:fileを使ってファイルのアップロードを行います。