

# **intra-mart WebPlatform/AppFramework Ver.7.0**

---

**画面デザインガイドライン**

**2012/03/26 第3版**



＜＜ 変更履歴 ＞＞

変更年月日	変更内容
2008/07/07	初版
2009/10/30	第 2 版 「5.3 検索種別」の誤字を修正しました。
2012/03/26	第 3 版 「3.3.4.5.2.3 属性」に<TEXTAREA>タグ読み出し専用入力フォームの記述例を追加しました。 「3.4.4.5.2.3 属性」に<TEXTAREA>タグ読み出し専用入力フォームの記述例を追加しました。



## &lt;&lt; 目次 &gt;&gt;

1	はじめに.....	1
1.1	開発環境条件.....	1
2	ユーザインタフェース・ガイドライン.....	2
2.1	ガイドラインの全体構成.....	2
2.1.1	デザイン性に関するガイドラインについて.....	2
2.1.2	操作性に関する指針について.....	2
2.2	デザイン性に関するガイドライン.....	3
2.2.1	すべてのページにデザインスタイルシートタグを記述する.....	4
2.2.2	すべてのページに内容を的確に示すタイトルバーをつける.....	5
2.2.3	処理や画面を切り替えるための「処理リンク」は、ツールバー内に配置する.....	6
2.2.4	処理リンクは画面デザイン共通モジュールを使用する.....	7
2.2.5	一覧のヘッダは画面デザイン共通モジュールを使用する.....	8
2.2.6	ソート切り替え、ページ切り替えは画面デザイン共通モジュールを使用する.....	9
2.2.7	入力項目のデザインを統一する.....	10
2.2.8	処理ボタンのデザインを統一する.....	15
2.2.9	テーブルタグ<TABLE>を使用して表を作成するときのクラスを統一する.....	16
2.2.10	文字色と背景色のコントラスト(明度差など)を充分に取る.....	18
2.2.11	画像で文字を使用する時は、文字フォント・サイズ・コントラストなどを考慮する.....	19
2.2.12	スタイルシートで文字サイズを指定しない.....	20
2.2.13	背景色を統一する.....	20
2.3	操作性に関するガイドライン.....	21
2.3.1	各画面ごとにヘルプ画面を用意する.....	21
2.3.2	新たなウィンドウ(ポップアップ画面)を開くことは、必要最小限にする.....	23
2.3.3	画像には、画像の内容を的確に示す alt属性をつける.....	24
2.3.4	コンボボックスで大量データを表示しない.....	25
2.3.5	フレーム化を廃止し、ウィザード形式に変更する.....	26
2.3.6	横方向のスクロールが発生しないようにする.....	27
3	画面デザイン共通モジュール.....	28
3.1	画面デザイン共通モジュールの全体構成.....	28
3.1.1	スクリプト開発モデルについて.....	28
3.1.2	JavaEE開発モデルについて.....	28
3.2	重要事項.....	29
3.2.1	デザインスタイルシートの宣言.....	29
3.3	スクリプト開発モデル.....	30
3.3.1	デザインスタイルシート.....	30
3.3.2	タイトルバー.....	31
3.3.3	ツールバー.....	32
3.3.4	入力項目.....	39
3.3.5	リストコントロール.....	51
3.3.6	リストヘッダ.....	57
3.4	JavaEE開発モデル.....	62
3.4.1	デザインスタイルシート.....	62
3.4.2	タイトルバー.....	63
3.4.3	ツールバー.....	64
3.4.4	入力項目.....	70
3.4.5	リストコントロール.....	84

3.4.6	リストヘッダ .....	89
4	画面デザインサンプル .....	96
4.1	検索系画面 .....	96
4.1.1	検索条件入力画面 .....	96
4.1.2	検索結果一覧画面 .....	97
4.2	登録系画面 .....	98
4.3	更新・削除系画面 .....	99
4.4	一覧系画面 .....	100
4.4.1	ユーザー一覧 .....	100
4.4.2	ローラー一覧 .....	102
5	共通アイコン .....	103
5.1	タイトルバー .....	103
5.2	処理系 .....	103
5.3	検索種別 .....	103
5.4	ページ切替 .....	104
5.5	昇順／降順切替 .....	104
5.6	選択リストボックス .....	104
5.7	ツリー表示 .....	104
5.8	ボタン・タブ .....	105

# 1 はじめに

---

本ドキュメントは、ユーザインタフェースの大幅な向上を目的とした、画面デザインガイドラインです。

intra-mart 開発者が、本ドキュメント内で示す、**ユーザインタフェース・ガイドライン**や、**画面デザイン共通モジュール** を利用することで、製品全体の画面デザインや、操作性が統一されることを目指します。

また、操作性とともに画面デザインを見直すことで、見た目といった製品の格調向上だけでなく、利用者がわかりやすく、使いやすい製品を目指します。

## 1.1 開発環境条件

以下の環境条件を整えた上で、開発作業を進めてください。

- 画面の解像度は 基本的に「1024×768」とする
- ブラウザのスクロールバー、アドレスバー、ツールバー、メニューバーなどは表示する
- 文字サイズは、ブラウザのメニュー [表示]-[文字サイズ] を「中」とする(IE の場合)

本ドキュメントは、IE を基にして作成されています。

他のブラウザを使用した場合、表示内容が異なる場合があります。

## 2 ユーザインタフェース・ガイドライン

---

ユーザインタフェース・ガイドライン は、intra-mart 製品の画面設計時に考慮すべき点を明確にすることで、製品全体の画面デザインや、操作性を統一させることを目的としています。

### 2.1 ガイドラインの全体構成

ユーザインタフェース・ガイドラインは、2つのテーマで構成しています。

- デザイン性に関するガイドライン
- 操作性に関するガイドライン

各ガイドラインで取り上げた内容について、「現状の問題点」や、それに対する「改善対策」を記述しています。

intra-mart 開発者は、これらのガイドラインを参考に画面設計を行ってください。

また、各ガイドラインには実装方法についてより理解しやすいように、具体的な「実装例」を記述しています。

実装を検討するときや、ガイドラインの意図が不明確なときに、参考にしてください。

#### 2.1.1 デザイン性に関するガイドラインについて

画面構成・カラー・文章(文言)・アイコンなど、画面全体で統一していただきたい要件を記載します。

詳細は「2.2 デザイン性に関するガイドライン」に示します。

#### 2.1.2 操作性に関する指針について

画面の遷移や、ポップアップ画面、フレームの扱いなどについて、改善する要件を記載します。

詳細は「2.3 操作性に関するガイドライン」に示します。



## 2.2 デザイン性に関するガイドライン

この章では、画面作成時における、全体のデザインを統一するためのカラー・文章・アイコンなどについて基準となる要件を記載します。

デザイン性に関する指針について「表 2-1 デザイン性に関するガイドライン」に示します。

表 2-1 デザイン性に関するガイドライン

項番	項目	ページ
2.2.1	すべてのページにデザインスタイルシートタグを記述する	4
2.2.2	すべてのページに内容を的確に示すタイトルバーをつける	5
2.2.3	処理や画面を切り替えるための「処理リンク」は、ツールバー内に配置する	6
2.2.4	処理リンクは画面デザイン共通モジュールを使用する	7
2.2.5	一覧のヘッダは画面デザイン共通モジュールを使用する	8
2.2.6	ソート切り替え、ページ切り替えは画面デザイン共通モジュールを使用する	9
2.2.7	入力項目のデザインを統一する	10
2.2.8	処理ボタンのデザインを統一する	15
2.2.9	テーブルタグ<TABLE>を使用して表を作成するときのクラスを統一する	16
2.2.10	文字色と背景色のコントラスト(明度差など)を充分に取る	18
2.2.11	画像で文字を使用する時は、文字フォント・サイズ・コントラストなどを考慮する	19
2.2.12	スタイルシートで文字サイズを指定しない	20
2.2.13	背景色を統一する	20

## 2.2.1 すべてのページにデザインスタイルシートタグを記述する

各ページで別々のスタイルシートを使用していると、サイト内全体の画面デザインに統一感が無くなる場合があります。「デザインスタイルシートタグ」を使用してスタイルシートの設定を固定化することで、サイト内全体の画面デザインを統一します。

デザインスタイルシートタグは、以下の「表 2-2 デザインスタイルシートの画面デザイン共通モジュール」に示す画面デザイン共通モジュールを使用してください。

### 2.2.1.1 実装例

- 各ページの<HEAD>タグ内には、必ずデザインスタイルシートタグを記述する
- デザインスタイルシートタグは、以下に示す画面デザイン共通モジュールを使用する。

表 2-2 デザインスタイルシートの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imDesignCss">	3.3.1	デザインスタイルシート
JavaEE 開発モデル	<imarttag:imartDesignCss>	3.4.1	デザインスタイルシート

- 良い例：デザインスタイルシートタグを使用して、スタイルシートを設定

スクリプト開発モデルの場合（HTML ファイルのサンプル）

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  :
  :
</BODY>
</HTML>
```

JavaEE 開発モデルの場合（JSP ファイルのサンプル）

```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/....." %>
:
<HTML>
<HEAD>
  <imarttag:imartDesignCss />
</HEAD>
<BODY>
  :
  :
</BODY>
</HTML>
```

- × 悪い例：<LINK>タグにより各自でスタイルシートを設定している

```
<HTML>
<HEAD>
  <LINK rel="stylesheet" type="text/css" href="css/common.css">
</HEAD>
<BODY>
  :
```

## 2.2.2 すべてのページに内容を的確に示すタイトルバーをつける

現在位置の表示がないと、利用者はサイト全体、もしくはコンテンツ内のどこを参照しているか、わからなくなることがあります。

各画面(ページ)の最上部には、ページの内容を的確に示したタイトルバーを表示します。

タイトルバーの表示には、以下の「表 2-3 タイトルバーの画面デザイン共通モジュール」に示す画面デザイン共通モジュールを使用してください。

### 2.2.2.1 実装例

- 各画面(ページ)の最上部には、「タイトルバー」を配置する
- タイトルバーには、「アイコン+ページタイトル名」を表示する
- タイトルバーの表示には、以下の画面デザイン共通モジュールを使用する

表 2-3 タイトルバーの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imTitleBar">	3.3.2	タイトルバー
JavaEE 開発モデル	<imarttag:imartTitleBar>	3.4.2	タイトルバー

- 良い例：ページの最上部に、内容を的確に示したタイトルバーを表示している

- × 悪い例：タイトルバーの表示がない

## 2.2.3 処理や画面を切り替えるための「処理リンク」は、ツールバー内に配置する

処理や画面を切り替えるための「処理リンク」を、画面内のあらゆる場所に配置していると、次の処理に戸惑うことがあります。各画面（ページ）で、タイトルバーの下部にツールバーを表示し、各「処理リンク」はツールバー内に配置します。ツールバーの表示には、以下の「表 2-4 ツールバーの画面デザイン共通モジュール」に示す、画面デザイン共通モジュールを使用してください。

また、処理リンクの詳細については次章の「2.2.4 処理リンクは画面デザイン共通モジュールを使用する」で説明します。

### 2.2.3.1 実装例

- 各画面（ページ）のタイトルバーの下部に、「ツールバー」を表示する
- ページ内で、処理や画面を切り替えるための「処理リンク」は、すべてツールバーに配置する（処理リンクの詳細については「2.2.4 処理リンクは画面デザイン共通モジュールを使用する」を参照）
- ツールバーの表示には、以下の画面デザイン共通モジュールを使用する

表 2-4 ツールバーの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imToolbarFrame">	3.3.3	ツールバー
	<IMART type="imToolbarLeft">		
	<IMART type="imToolbarRight">		
JavaEE 開発モデル	<imarttag:imartToolbarFrame>	3.4.3	ツールバー
	<imarttag:imartToolbarLeft>		
	<imarttag:imartToolbarRight>		

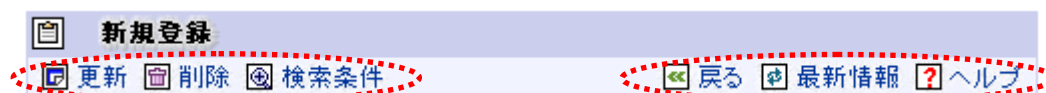
- 処理リンク配置位置の方針  
メイン処理系（登録・更新・検索）のリンクは、ツールバーの 左側 に配置する。  
サブ処理系（最新情報・ヘルプ）のリンクは、ツールバーの 右側 に配置する。  
[参考] よく使用される処理リンクをツールバーの右側／左側に分けて、以下に示す。

表 2-5 ツールバー内の処理リンク配置位置例

使用箇所		アイコン	内容
ツールバー	左側 :メイン処理系 (登録、更新、検索など)		new.gif
			edit.gif
			print.gif
			search.gif
	右側 :サブ処理系 (戻る、最新情報・ヘルプなど)		arrow_left.gif
			reload.gif
			help.gif
			close.gif

※ その他のアイコンは、「5 共通アイコン」を参照すること。

- 良い例 : 各ページのタイトルバーの下部にツールバーを表示し、処理リンクを配置している



## 2.2.4 処理リンクは画面デザイン共通モジュールを使用する

画面内に多数ある「処理リンク」を、個人で配置するとデザインや表示方法に統一感が無くなる場合があります。処理リンクを配置する場合は、以下の「表 2-6 処理リンクの画面デザイン共通モジュール」に示す、画面デザイン共通モジュールを使用してください。

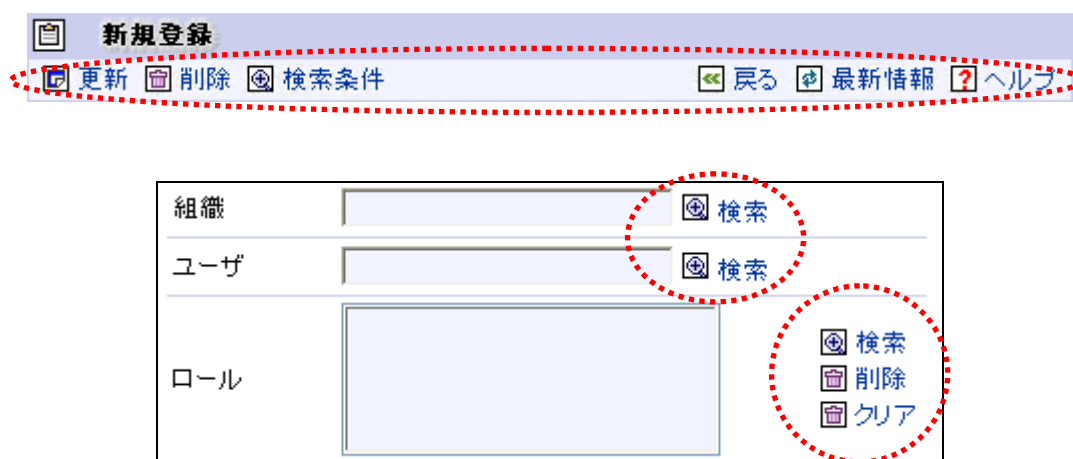
### 2.2.4.1 実装例

- 処理リンクは、画面デザイン共通モジュールすべてツールバー内に配置する  
(ツールバーの詳細については「2.2.3 処理や画面を切り替えるための「処理リンク」は、ツールバー内に配置する」を参照)
- 処理リンクには、「アイコン+処理名」を表示する
- 処理リンクの表示には、以下の画面デザイン共通モジュールを使用する

表 2-6 処理リンクの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imlcon">	3.3.3.5.4	<IMART type="imlcon">
JavaEE 開発モデル	<imarttag:imartlcon>	3.4.3.5.4	<imarttag:imartlcon>

- 良い例：画面デザイン共通モジュールを使用して「処理リンク」を作成している



#### ■ スクリプト開発モデルの場合（HTML のサンプル）

```
<HTML>
<HEAD>
<IMART type = "imDesignCss"></IMART>
</HEAD>
<BODY>
:
<IMART type = "imlcon"
  name = "検索"
  icon = "images/standard/serch.gif"
  href = "javascript:onSerch();">
</IMART>
:
</BODY>
</HTML>
```

#### ■ JavaEE 開発モデルの場合（JSP のサンプル）

```
<HTML>
<HEAD>
<imarttag:imartDesignCss />
</HEAD>
<BODY>
:
<imarttag:imartlcon
  name = "検索"
  icon = "/images/standard/serch.gif"
  href = "javascript:onSerch();" />
:
</BODY>
</HTML>
```

### 2.2.5 一覧のヘッダは画面デザイン共通モジュールを使用する

画面内に一覧(リスト)を表示するときの「一覧のヘッダ」を個人で作成すると、デザインや表示方法に統一感が無くなる場合があります。

一覧(リスト)を表示するときの一覧ヘッダは、以下の「表 2-7 一覧ヘッダの画面デザイン共通モジュール」に示す、画面デザイン共通モジュールを使用してください。

#### 2.2.5.1 実装例

- 一覧ヘッダの表示には、以下の画面デザイン共通モジュールを使用する

表 2-7 一覧ヘッダの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imListHeader">	3.3.6	リストヘッダ
JavaEE 開発モデル	<imarttag:imartListHeader>	3.4.6	リストヘッダ

- 良い例：画面デザイン共通モジュールを使用して、一覧ヘッダを作成している

<input type="checkbox"/>	ユーザコード	ユーザ名	会社／組織
<input type="checkbox"/>	user001	ユーザ001	サンプル会社／サンプル部門01
<input type="checkbox"/>	user002	ユーザ002	サンプル会社／サンプル部門01／サンプル課11

## 2.2.6 ソート切り替え、ページ切り替えは画面デザイン共通モジュールを使用する

画面内に一覧(リスト)を表示するとき必要となる、「ソート切り替え」および「ページ切り替え」のコントロール(以下、リストコントロール)を個人で作成すると、デザインや表示方法に統一感が無くなる場合があります。

一覧表示をするときのリストコントロールは、以下の「表 2-8 リストコントロールの画面デザイン共通モジュール」に示す、画面デザイン共通モジュールを使用してください。

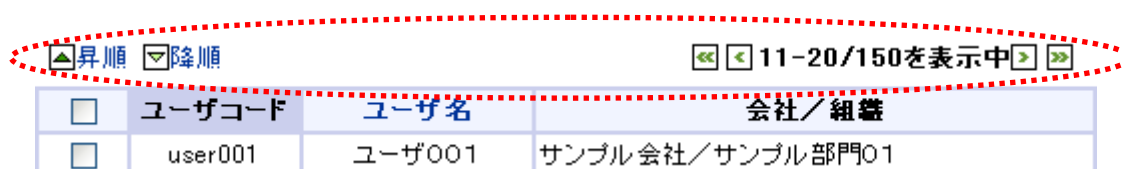
### 2.2.6.1 実装例

- 一覧表示をするときの「リストコントロール」には、以下の画面デザイン共通モジュールを使用する

表 2-8 リストコントロールの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imListControl">	3.3.5	リストコントロール
JavaEE 開発モデル	<imarttag:imartListControl>	3.4.5	リストコントロール

- 良い例：画面デザイン共通モジュールを使用して、リストコントロールを作成している



スクリプト開発モデルの場合（HTML ファイルのサンプル）

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  :
  :
  <TABLE class="" width="100%">
    <TR>
      <IMART type="imListControl"
        maxRecord   = all_dataCnt // 全件数
        currentPage = page_num    // 現在のページ番号
        pageLine    = view_max    // 1 ページの表示件数
        sortOrder   = sort_order  // ソート昇順・降順を指定
      </IMART>
    </TR>
    :
  </TABLE>
  :
</BODY>
</HTML>
```

※ JavaEE開発モデルの場合のサンプルは「3.4.5 リストコントロール」の「3.4.5.4 使用例」を参照してください。

## 2.2.7 入力項目のデザインを統一する

新規登録や、更新・削除、および検索条件入力画面などで、入力項目や項目名称(ラベル)を表示するときに、個人で作成するとデザインや表示方法に統一感が無くなる場合があります。

入力項目および、項目名称のデザインを統一するために、これらを作成するときには、「2.2.7.1 入力項目の画面デザイン共通モジュール」に示す画面デザイン共通モジュールを使用することを推奨します。

また、画面デザイン共通モジュールを使用しない場合や、画面デザイン共通モジュールだけでは賅えない入力項目の部品を各自で作成する場合は、「2.2.7.2 入力項目の書式方法」に示す書式で作成してください。

### 2.2.7.1 入力項目の画面デザイン共通モジュール

入力項目および、項目名称を作成するときには、以下の「表 2-9 入力項目を作成するために使用する画面デザイン共通モジュール」に示す画面デザイン共通モジュールを使用してください。

#### 2.2.7.1.1 実装例

- 入力項目の表示には、以下の画面デザイン共通モジュールを使用する

表 2-9 入力項目を作成するために使用する画面デザイン共通モジュール

入力項目部品	画面デザイン共通モジュール	詳細	
項目名	<IMART type="imItemName">	3.3.4.5.1	項目名称
	<imarttag:imartItemNameTd>	3.4.4.5.1	項目名称
入力フィールド	<IMART type="imInputTd">	3.3.4.5.2	入力フィールド
	<imarttag:imartInputTd>	3.4.4.5.2	入力フィールド
選択ボックス	<IMART type="imSelectTd">	3.3.4.5.3	選択ボックス
	<imarttag:imartSelectTd>	3.4.4.5.3	選択ボックス

- 良い例：入力項目および、項目名称を画面デザイン共通モジュールで作成している

ユーザID	<input type="text" value="User00011"/>
ユーザ名(必須)	<input type="text"/>
パスワード(必須)	<input type="password"/>
出身地	<input type="text" value="東京都"/> ▼
転送ファイル	<input type="text"/> <input type="button" value="参照..."/>

※ 記述例を次頁の「表 2-10 入力項目を作成する画面デザイン共通モジュールの記述例」に示します



表 2-10 入力項目を作成する画面デザイン共通モジュールの記述例

```

<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
<%
    Vector from = new Vector();
    HashMap hashTemp = new HashMap();

    // 月データ
    String[] fromList = new String[]{"東京都", "神奈川県", "埼玉県", "千葉県"};
    String[] fromValue = new String[]{"Tokyo", "Kanagawa", "Saitama", "Chiba"};

    // コンボ値(月)
    for(int i = 0 ; i < fromList.length ; i++) {
        hashTemp = new HashMap();
        hashTemp.put("value", fromValue[i]);
        hashTemp.put("from", fromList[i]);
        from.add(hashTemp);
    }
%>
<HTML>
<HEAD>
<imarttag:imartDesignCss />
</HEAD>
<BODY>
<!-- 入力項目表示 -->
<TABLE width="100%" border="0" cellpadding="0" cellspacing="0">
<TR>
<TD align="center">
<TABLE class="edit">
<TR>
<imarttag:imartItemNameTd name="ユーザ ID" />
<imarttag:imartInputTd
type="text" name="userId" size="30" value="User00011" readonly />
</TR>
<TR>
<imarttag:imartItemNameTd name="ユーザ名" require />
<imarttag:imartInputTd
type="text" name="userId" size="50" value="ユーザ 00011" />
</TR>
<TR>
<imarttag:imartItemNameTd name="パスワード" require />
<imarttag:imartInputTd type="password" name="passWord" size="40" />
</TR>
<TR>
<imarttag:imartItemNameTd name="出身地" />
<imarttag:imartSelectTd
list="<%= from %>" name="hometown" optionValue="value" optionText="from" />
</TR>
<TR>
<imarttag:imartItemNameTd name="転送ファイル" />
<imarttag:imartInputTd type="file" name="forwarFile" size="50" />
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>

```

### 2.2.7.2 入力項目の書式方法

前章の「2.2.7.1 入力項目の画面デザイン共通モジュール」を使用しない場合で、項目名称(ラベル)や、入力項目および、選択項目を各自で作成するときには、画面デザインを統一するために、各HTMLタグに指定するクラスや、書式方法を必ず守ってください。

以下に、指定するクラスや、書式方法の詳細を示します。

#### 2.2.7.2.1 入力項目、項目名称に指定するクラス

入力項目を表示する <TABLE>、<TD> および <INPUT>タグには、以下の「表 2-11 入力項目の作成時に指定するタグのクラス」に示すclassを必ず指定してください。

表 2-11 入力項目の作成時に指定するタグのクラス

タグ	クラス	用途
<TABLE>	edit	全体のデザインを整える
<TD>	bottom	アンダーラインを表示する
<INPUT>	default	入力項目の背景色
	default_right	入力項目の背景色 (テキスト右寄せ表示)
<SELECT>	default	選択項目の背景色
<FONT>	attention	必須項目名称

#### ■ HTML の記述例

```
<HTML>
<BODY>
  <TABLE class="edit" width="100%">
    <TR>
      <TD class="bottom">
        <STRONG>ユーザ ID</STRONG> <FONT class="attention">(必須)</FONT>
      </TD>
      <TD class="bottom"><INPUT class="default" type="text" ..... ></TD>
    </TR>
    <TR>
      <TD class="bottom">ソート番号</TD>
      <TD class="bottom"><INPUT class="default_right" type="text" ..... ></TD>
    </TR>
    <TR>
      <TD class="bottom">出身地</TD>
      <TD class="bottom"><SELECT class="default" ..... ></TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

#### ■ 画面表示例

ユーザID(必須)	<input type="text" value="user001"/>
ソート番号	<input type="text" value="100"/>
出身地	<input type="text" value="▼"/>

### 2.2.7.2.2 項目名称の書式

入力項目、および項目名称(ラベル)を表示する場合は、以下の書式を必ず守って作成してください。

#### 2.2.7.2.2.1 必須項目の場合

- 項目名称を 黒色の『太字』で表示し、赤色で「(必須)」の文字を記述する。
- (1) 入力項目を作成する <TABLE>タグに class="edit" を指定する。
- (2) <TD>タグに class="bottom" を指定する。
- (3) <STRONG></STRONG> タグを記述して、項目名称を『太字』にする。  
「必須」の文字を <FONT class="attention">(必須)</FONT> で記述する。

```
<TABLE class="edit"> ..... (1)
<TR>
<TD class="bottom"> ..... (2)
  <STRONG>ユーザ名</STRONG> <FONT class="attention">(必須)</FONT> ... (3)
</TD>
<TD class="bottom">
<INPUT .....>
</TD>
</TR>
:
</TABLE>
```

(表示結果)

ユーザ名(必須)

#### 2.2.7.2.2.2 通常項目の場合

- 項目名称を 黒色の標準の太さで表示する。
- (1) 入力項目を作成する <TABLE>タグに class="edit" を指定する。
- (2) <TD>タグに class="bottom" を指定して、項目名称を記述する。

```
<TABLE class="edit"> ..... (1)
<TR>
<TD class="bottom">ユーザ名</TD> ..... (2)
<TD class="bottom">
<INPUT .....>
</TD>
</TR>
:
</TABLE>
```

(表示結果)

ユーザID

### 2.2.7.2.3 入力フィールドの書式

入力項目の「入力フィールド」を表示する場合は、以下の書式を必ず守って作成してください。

- (1) 入力項目を作成する <TABLE>タグに class="edit" を指定する。
- (2) 項目名称を記述する。（「2.2.7.2.2 項目名称の書式」を参照）
- (3) <TD>タグに class="bottom" を指定する。
- (4) 作成する入力フィールドの <INPUT>タグや <SELECT>タグに class="default" を指定する。
- (5) 数値入力などで入力文字を「右寄せ」で表示したい場合は、<INPUT>タグに class="default\_right" を指定する。

#### ■ <INPUT> タグで入力項目の作成

```
<TABLE class="edit"> ..... (1)
<TR>
<TD class="bottom"> ... 項目名称を記述 ...</TD> ..... (2)
<TD class="bottom"> ..... (3)
    <INPUT class="default" type="text" value="user001" ...> .... (4)
</TD>
</TR>
:
</TABLE>
```

(表示結果)

user001
---------

#### ■ <INPUT> タグで入力項目の作成

```
<TABLE class="edit"> ..... (1)
<TR>
<TD class="bottom"> ... 項目名称を記述 ...</TD> ..... (2)
<TD class="bottom"> ..... (3)
    <INPUT class="default_right" type="text" value="100" ...> ... (5)
</TD>
</TR>
:
</TABLE>
```

(表示結果)

100
-----

#### ■ <SELECT> タグで選択項目の作成

```
<TABLE class="edit"> ..... (1)
<TR>
<TD class="bottom"> ... 項目名称を記述 ...</TD> ..... (2)
<TD class="bottom"> ..... (3)
    <SELECT class="default" ...> ..... (4)
</TD>
</TR>
:
</TABLE>
```

(表示結果)

--

## 2.2.8 処理ボタンのデザインを統一する

画面内に多数ある「処理ボタン」を、個人で配置するとデザインや表示方法に統一感が無くなる場合があります。デザインを統一するために、処理ボタンを配置する場合は、以下の「2.2.8.1 実装例」で示す書式で作成してください。

### 2.2.8.1 実装例

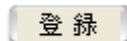
- デザインを統一するために、処理ボタンを作成するときは、必ず以下の書式を使用すること

- 良い例：処理ボタンを、決められた書式で作成している

[処理ボタンを作成する HTML 部分のサンプル]

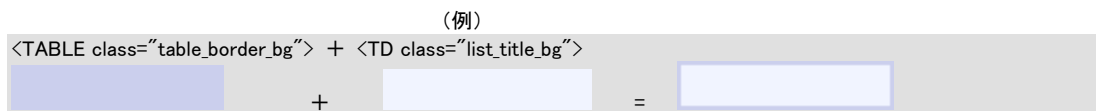
```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  :
  :
  <!-- 「登録」ボタンの作成 -->
<TABLE class="button">
  <TR>
    <TD class="button_padding">
      <IMG src="images/standard/button_left.gif">
    </TD>
    <TD class="button_bg">
      <INPUT name="regist" type="submit" class="button_bg" value=" 登 録 ">
    </TD>
    <TD class="button_padding">
      <IMG src="images/standard/button_right.gif">
    </TD>
  </TR>
</TABLE>
  :
  :
</BODY>
</HTML>
```

[上記のサンプルにより、表示される処理ボタン]



### 2.2.9 テーブルタグ<TABLE>を使用して表を作成するときのクラスを統一する

<TABLE>タグを使用して表(リスト)を作成する場合は、テーブルのボーダラインを統一するために、<TABLE>タグおよび <TD>タグには、以下の「表 2-12 表を作成時に指定するクラス」に示す classを設定します。  
<TABLE>タグと<TD>タグに指定する class の組み合わせによって、表の作成を実現することができます。



#### 2.2.9.1 実装例

- 表を作成する場合は、必ず <TABLE>タグ および <TD>タグに class の指定を記述する
- <TABLE>タグの class には、「list\_border\_bg」または「table\_border\_bg」を使用する
- <TD>タグには、使用する用途によって class の記述を変更する

表 2-12 表を作成時に指定するクラス

タグ	クラス	用途	表示例
<TABLE>	class="list_border_bg"	表全体の枠 横幅:100%固定	
	class="table_border_bg"	表全体の枠 横幅:任意指定	
	<TD> class="list_title_bg"	項目名(通常) 表示位置:任意	
	class="list_title_bg_center"	項目名(通常) 表示位置:中央	
	class="list_title_bg_left"	項目名(通常) 表示位置:左寄せ	
	class="list_title_bg_right"	項目名(通常) 表示位置:右寄せ	
	class="list_title_sort_bg"	項目名(強調) 表示位置:任意	
	class="list_title_sort_bg_center"	項目名(強調) 表示位置:中央	
	class="list_title_sort_bg_left"	項目名(強調) 表示位置:左寄せ	
	class="list_title_sort_bg_right"	項目名(強調) 表示位置:右寄せ	
	class="list_data_bg"	データ部 表示位置:中央	
	class="list_data_bg_left"	データ部 表示位置:左寄せ	
	class="list_data_bg_right"	データ部 表示位置:右寄せ	

- 良い例 : 表を作成する<TABLE>および<TD>タグに、規定の class を指定している

スクリプト開発モデルの場合 (HTML ファイルのサンプル)

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  <TABLE class="list_border_bg">
    <TR>
      <TD class="list_title_bg_center">.....</TD>
      :
    </TR>
    <TR>
      <TD class="list_data_bg">.....</TD>
      :
    </TR>
  </TABLE>
</BODY>
</HTML>
```

- × 悪い例 : <TABLE>および<TD>タグに、各自で属性を設定している

スクリプト開発モデルの場合 (HTML ファイルのサンプル)

```
<HTML>
<BODY>
  <TABLE border="0" cellpadding="0" cellspacing="2">
    <TR>
      <TD>.....</TD>
      :
    </TABLE>
  :
```

## 2.2.10 文字色と背景色のコントラスト(明度差など)を充分に取る

文字色と背景色のコントラスト(明度差など)が小さいほど、文字は読みにくくなります。

また、ディスプレイやプリンタ、OSの種類によって色の再現性が微妙に異なるため、明度コントラストが小さいと文字が読みにくくなる可能性が高くなります。

文字色と背景色のコントラストを充分に取って、文字を読みやすくしてください。

### 2.2.10.1 実装例

- 背景色と文字色の明度の差を充分に確保する
- 特に、赤と緑、白と黄、白と水色、青と黒、青紫と黒の組み合わせなどに注意する

○ 良い例 : 背景色と文字色に明度差があるので、文字が読みやすい

The screenshot shows a '新規登録' (New Registration) form. It has a light blue header bar with navigation links like '更新', '削除', '検索条件', '戻る', '最新情報', and 'ヘルプ'. The form fields are white with blue borders and labels. The labels 'ユーザID(必須)', 'ユーザ名(必須)', 'パスワード', and '組織' are in bold black text. The password field is masked with dots. A blue '検索' (Search) button is at the bottom right.

The screenshot shows a '検索結果一覧' (Search Results List) page. It has a light blue header bar with navigation links. Below the header, there are search criteria: '検索条件項目1: xxxxxxxxxx', '検索条件項目2:', and '検索条件項目3: 2005年01月31日'. There are sorting options '昇順' (Ascending) and '降順' (Descending), and a status bar '11-20/132 表示中'. The table below has a light blue header and white data rows.

	ユーザID	ユーザ名	会社/組織
<input type="checkbox"/>	User0011	ユーザ11	サンプル会社/サンプル部門01/サンプル課11
<input type="checkbox"/>	User0012	ユーザ12	サンプル会社/サンプル部門01/サンプル課11
<input type="checkbox"/>	User0013	ユーザ13	サンプル会社/サンプル部門01/サンプル課11

<input type="checkbox"/>	ユーザコード	ユーザ名	会社/組織
--------------------------	--------	------	-------

× 悪い例 : 背景色と文字色に明度差がなく、文字が読みにくい

The screenshot shows a 'ユーザプロフィール - 新規登録' (User Profile - New Registration) form. The header bar is blue with white text. The form fields are white with blue borders. The labels 'ユーザコード(\*)', '名前(\*)', 'カナ', '英字', and '電話番号' are in blue text. The password field is masked with dots.

The screenshot shows a 'ユーザプロフィール 一覧' (User Profile List) page. It has a blue header bar with navigation links. Below the header, there are search criteria: '検索条件項目1: xxxxxxxxxx', '検索条件項目2:', and '検索条件項目3: 2005年01月31日'. There are sorting options '昇順' (Ascending) and '降順' (Descending), and a status bar '1-15/15を表示中'. The table below has a blue header and white data rows.

ユーザコード	名前	ユーザプロフィール	ログインアカウント
soysai	青柳辰巳	編集	編集
guest	ゲスト	編集	編集

ユーザコード	名前	ユーザプロフィール	ログインアカウント
--------	----	-----------	-----------



## 2.2.11 画像で文字を使用する時は、文字フォント・サイズ・コントラストなどを考慮する

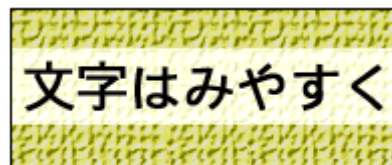
画像内に描かれた文字は、ブラウザでサイズや色のコントラストを変更できません。

無意味に文字を画像にすることは避け、画像にする場合は、サイズや色のコントラストに配慮し、読みやすくしてください。

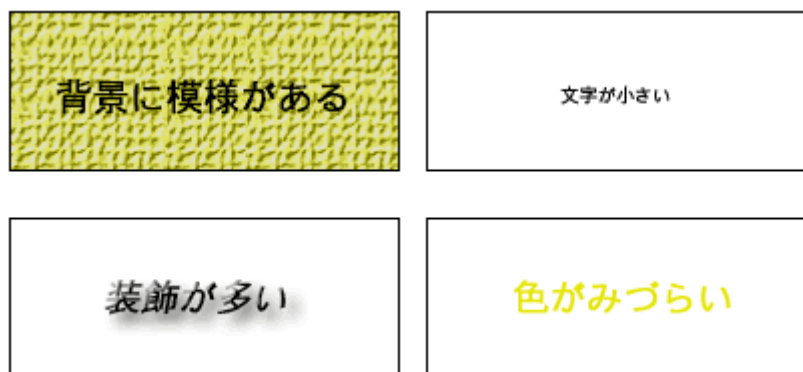
### 2.2.11.1 実装例

- 文字（特に漢字）の装飾（斜体など）は少なくする
- 文字の背景に模様のある画像や写真などを使用する時は、文字の周りを縁取るなどし、文字を見やすくする
- 文字のフォントは、ゴシック系を使用することが望ましい（画面上では、明朝系よりも、ゴシック系のフォントのほうが見やすいため）

○ 良い例：読みやすい



× 悪い例：いずれも文字が読みにくい



## 2.2.12 スタイルシートで文字サイズを指定しない

スタイルシートを使用して文字サイズを指定する場合、絶対単位 (in, cm, mm, pt, pc) を使用すると、ブラウザの文字サイズを変更しても、大きさが変わりません。

## 2.2.13 背景色を統一する

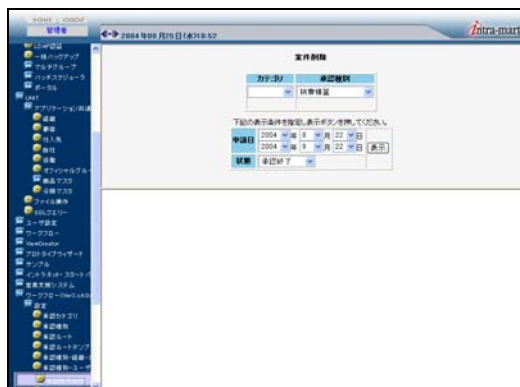
各ページや画面によって、背景色 (バックグラウンドカラー) が異なっています。また、同一画面内でも、フレーム分けされた背景色が異なっている場合があります。

サイト内で背景色を統一してください。

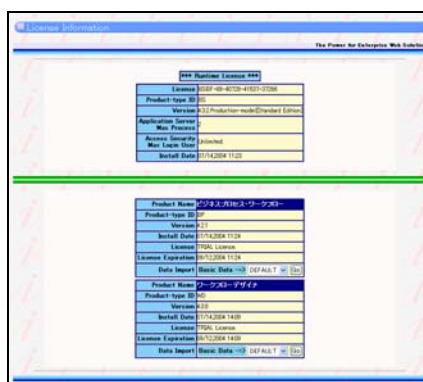
### 2.2.13.1 実装例

- 背景色 (バックグラウンドカラー) を規定し、AP 内で統一する
- 文字や画像の邪魔にならない背景色を使用する

× 悪い例：同一画面内で、背景色が異なる



× 悪い例：ページによって背景色が異なる



## 2.3 操作性に関するガイドライン

操作性に関する指針では、画面作成時における、全体のデザインを統一するためのカラー・文章・アイコンなどについて基準となる要件を記載します。

操作性に関する指針について「表 2-13 操作性に関する指針」に示します。

表 2-13 操作性に関する指針

項番	項目	ページ
2.3.1	各画面ごとにヘルプ画面を用意する	21
2.3.2	新たなウィンドウ(ポップアップ画面)を開くことは、必要最小限にする	23
2.3.3	画像には、画像の内容を的確に示す alt属性をつける	24
2.3.4	コンボボックスで大量データを表示しない	25
2.3.5	フレーム化を廃止し、ウィザード形式に変更する	26
2.3.6	横方向のスクロールが発生しないようにする	27

### 2.3.1 各画面ごとにヘルプ画面を用意する

メインの画面(ページ)に解説やヘルプとなる文章を記載していると、画面全体がゴチャゴチャし、まとまりがなくなってしまう。

ヘルプ画面を別途用意することで、入力項目とヘルプ解説部分が切り分けられ、画面全体の見やすさだけでなく、操作性も向上します。

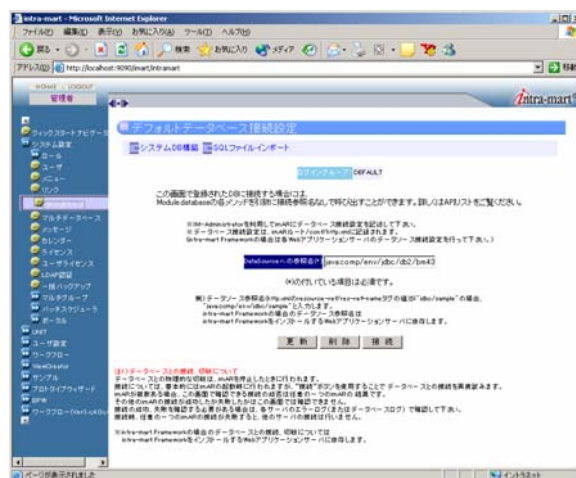
#### 2.3.1.1 実装例

- 各画面ごとに、ヘルプ画面を用意する
- ヘルプ画面はポップアップで表示する

○ 良い例 : 解説部分が別途ヘルプ画面として用意された



× 悪い例：メイン画面内に解説を書き過ぎていて見づらい



### 2.3.2 新たなウィンドウ(ポップアップ画面)を開くことは、必要最小限にする

必要以上に多くのウィンドウを開くと、サイトを表示している機器に負担がかかるため、コンテンツの表示速度が遅くなる場合があります。

また、新しいウィンドウが開いたことに気づかなかったり、どのウィンドウで作業していたのかわからなくなったり、その変化に戸惑う場合があります。

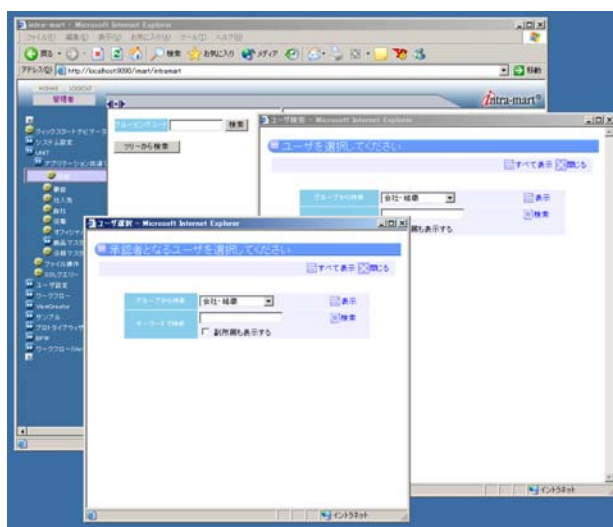
さらに、多くのウィンドウが開いた場合、不要なウィンドウを閉じなければならず、余計な操作が発生することになります。

新たなウィンドウ(ポップアップ画面)を開かずに、ウィザード形式で画面を遷移するようにしてください。

#### 2.3.2.1 実装例

- 新たなウィンドウ(ポップアップ画面)を開かずに、ウィザード形式で画面遷移をする。
- ポップアップ画面のほうが、内容を参照しやすい場合は、あらかじめリンク元で新しいウィンドウが開くことを明示しておくほうが望ましい。  
例えば、「ユーザ検索 (新しいウィンドウで表示)」などと表記する。

× 悪い例：新しいウィンドウ(ポップアップ画面)がいくつも表示されている



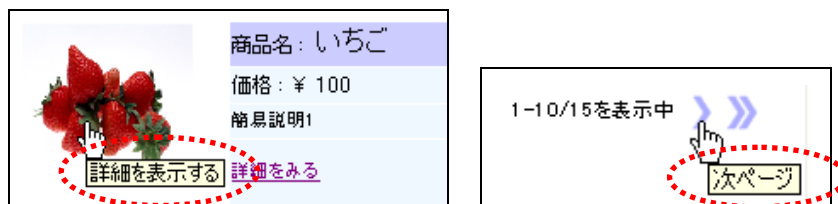
### 2.3.3 画像には、画像の内容を的確に示す alt 属性をつける

利用者は、画像に alt 属性が指定されていないと、画像の内容を把握できない場合があります。  
また、リンクのある画像の場合は、リンクでの遷移先が明確でないと操作に戸惑う場合があります。  
画像には alt 属性をつけて、画像の内容を的確に示してください。

#### 2.3.3.1 実装例

- 画像にリンクがない場合
  - ◆ alt 属性で画像の内容を記述する
  - ◆ 意味を持たない画像（箇条書きのポインタなど）や、テキストが併記されている画像には、alt=""と記述する（""の中には何も入力しない）
- 画像にリンクがある場合
  - ◆ alt 属性でリンク先を記述する
  - ◆ リンク先を alt 属性として記述することで、画像の説明が不要となる場合は、画像の説明を省略してよい
  - ◆ 画像の内容を詳細に解説する必要がある場合は、リンク先は alt 属性に記述し、画像の解説は画像と同じ HTML 内にテキストで記述する
- その他
  - ◆ 画像のボタン(image タイプの<input>タグに type="image" を使用する場合)にも、alt 属性を指定する

○ 良い例：alt 属性でリンク先を記述



○ 良い例：alt 属性で画像を解説

グループコード	グループ名	種別
comp_sample_01	サンプル会社	
orgn_sample_10	サンプル会社/サンプル部門01	会社・組織

× 悪い例：alt 属性なし



### 2.3.4 コンボボックスで大量データを表示しない

大量データをコンボボックスで表示すると、画面の内容が隠れてしまったり、スクロールによってデータを探さなければならぬため、操作に戸惑う場合があります。

大量データや可変データの場合は、コンボボックス表示ではなく、リスト表示形式にすることで、参照や操作性が向上します。

#### 2.3.4.1 実装例

- 大量データや可変データは、コンボボックスではなくリスト形式で表示する

○ 良い例：大量データをリスト形式で表示している



× 悪い例：大量のデータをコンボボックスで表示している



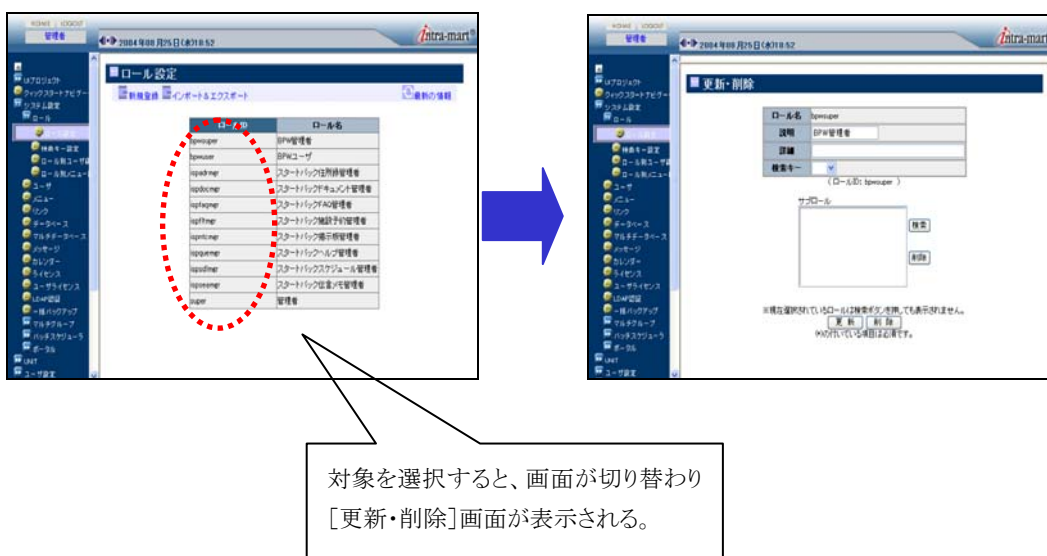
### 2.3.5 フレーム化を廃止し、ウィザード形式に変更する

無駄なフレーム化は、ユーザビリティだけでなく、デザイン性やレスポンスの低下を招く原因となっています。フレーム化を廃止し、ウィザード形式やタブによるページ切り替えに変更することで、操作性がアップし、見やすさも向上します。

#### 2.3.5.1 実装例

- フレーム化を廃止する → ウィザード形式やタブによるページ切り替えに変更

○ 良い例：ウィザード形式に変更し、操作性やデザイン性がアップ



× 悪い例：フレーム分けが多く、操作がわかりづらい。またデザイン性も悪い。





### 2.3.6 横方向のスクロールが発生しないようにする

ブラウザで縦と横のスクロールが表示されている場合、ページ全体の把握が困難になります。スクロールを縦方向だけにすることで、ページを参照しやすくなります。

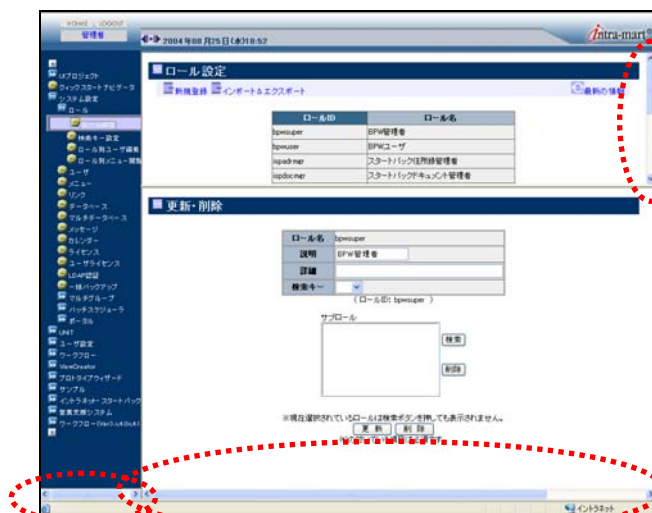
#### 2.3.6.1 実装例

- 画面を作成する際には、横方向のスクロールが表示されないように設計する

- 良い例：横方向スクロールが表示されないように、画面を設計



- × 悪い例：横方向スクロールと、縦方向スクロールが混同していて操作が困難。またデザイン性も悪い。



## 3 画面デザイン共通モジュール

intra-mart 製品の画面（ページ）内で、共通して利用可能なコントロール部品を、画面デザイン共通モジュール（スクリプト開発モデル、および JavaEE 開発モデル）として用意しています。

本章で示す、各 画面デザイン共通モジュール は、製品全体の画面デザインや、操作性の統一を目的としています。

### 3.1 画面デザイン共通モジュールの全体構成

画面デザイン共通モジュールは、2 つの開発モデルで用意されています。

- スクリプト開発モデル
- JavaEE 開発モデル

※各画面デザイン共通モジュールでは、使用方法についてより理解しやすいように、具体的な「使用例」を記述しています。実装する際の参考にしてください。

#### 3.1.1 スクリプト開発モデルについて

スクリプト開発モデル用に用意されている、画面デザイン共通モジュールを以下に示します。

表 3-1 スクリプト開発モデルの画面デザイン共通モジュール

画面デザイン共通モジュール		ページ
3.3.1	デザインスタイルシート	30
3.3.2	タイトルバー	31
3.3.3	ツールバー	32
3.3.4	入力項目	39
3.3.5	リストコントロール	51
3.3.6	リストヘッダ	57

#### 3.1.2 JavaEE 開発モデルについて

JavaEE 開発モデル用に用意されている、画面デザイン共通モジュールを以下に示します。

表 3-2 JavaEE 開発モデルの画面デザイン共通モジュール

画面デザイン共通モジュール		ページ
3.4.1	デザインスタイルシート	62
3.4.2	タイトルバー	63
3.4.3	ツールバー	64
3.4.4	入力項目	70
3.4.5	リストコントロール	84
3.4.6	リストヘッダ	89

## 3.2 重要事項

### 3.2.1 デザインスタイルシートの宣言

本章で示す「画面デザイン共通モジュール」を使用する際は、必ず各画面を構成しているHTMLの<HEAD>タグ内に、

『デザインスタイルシート』の宣言を行ってください。

『デザインスタイルシート』は、画面デザイン共通モジュールとして用意されています。

詳細は、以下を参照してください。

表 3-3 デザインスタイルシートの画面デザイン共通モジュール

開発モデル	画面デザイン共通モジュール		詳細
スクリプト開発モデル	<IMART type="imDesignCss">	3.3.1	デザインスタイルシート
JavaEE 開発モデル	<imarttag:imartDesignCss>	3.4.1	デザインスタイルシート

以下に、デザインスタイルシートの記述例を示します。

【記述例 ①】：スクリプト開発モデルの場合（HTML ファイルのサンプル）

```
<HTML>
  <HEAD>
    <IMART type="imDesignCss"></IMART>
  </HEAD>
  <BODY>
    :
    :
  </BODY>
</HTML>
```

【記述例 ②】：JavaEE 開発モデルの場合（JSP ファイルのサンプル）

```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
:
<HTML>
  <HEAD>
    <imarttag:imartDesignCss />
  </HEAD>
  <BODY>
    :
    :
  </BODY>
</HTML>
```

## 3.3 スクリプト開発モデル

### 3.3.1 デザインスタイルシート

#### 3.3.1.1 機能詳細

- `<IMART type="imDesignCss">` により、スタイルシートを設定する。
- ServerManager 配下の `conf/system.xml` ファイルの `system/color-config/color-pattern` タグ内で設定したCSSファイルが利用可能になります。  
`system/color-config/color-pattern` に複数のCSSファイルを設定することも可能です。

#### 3.3.1.2 制約

本タグを配置する場所は、HTML の`<HEAD>` タグ内です。

#### 3.3.1.3 属性

`<IMART type="imDesignCss">` に指定する属性を、以下に示します。

- 属性 `type` には IMART タグ名「`imDesignCss`」を指定する。

属性	説明	必須	デフォルト値	書式
<code>type</code>	IMART タグ名	○	—	<code>type="imDesignCss"</code>

#### 3.3.1.4 使用例

デザインスタイルシートを設定するためのサンプルを、以下に示します。

##### HTML のサンプル

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  :
  :
</BODY>
</HTML>
```

#### 3.3.1.5 注意点

各画面(ページ)を作成する html ファイルの `<HEAD>` タグ内には、  
必ず デザインスタイルシート `<IMART type="imDesignCss"></IMART>` を記述してください。

### 3.3.2 タイトルバー

#### 3.3.2.1 機能詳細


- <IMART type="imTitleBar"> により、タイトルバーを表示する。
- タイトルバーには、「アイコン」と「ページタイトル名」を指定して表示することができる。

#### 3.3.2.2 制約

本タグを配置する場所は、HTML の<BODY> タグ内です。

#### 3.3.2.3 属性

<IMART type="imTitleBar"> に指定する属性を、以下に示します。

- 属性 type には IMART タグ名「imTitleBar」を指定する。
- 属性 title には「ページタイトル名」を指定する。
- 属性 icon には「アイコン」のファイル名を、ドキュメントルートからの相対パスで指定する。  
属性 icon を未指定にした場合は、アイコン  (images/standard/title.gif) が表示される。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imTitleBar"
title	ページタイトル名	○	—	title="新規登録"
icon	アイコンのファイル名 (ドキュメントルートからの相対パス)		list.gif	icon="images/standard/title.gif"

#### 3.3.2.4 使用例

タイトルバーを表示するためのサンプルを、以下に示します。

##### HTML のサンプル

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
</HEAD>
<BODY>
  <!-- タイトルバー表示 -->
  <IMART type="imTitleBar"
    title="新規登録"
    icon="images/standard/title.gif">
  </IMART>
</BODY>
</HTML>
```

##### 表示結果



### 3.3.3 ツールバー

#### 3.3.3.1 機能詳細

- <IMART type="imToolbarFrame"> タグにより、ツールバーの外枠を表示する。  
(タグの詳細は「3.3.3.5.1 <IMART type="imToolbarFrame">」を参照)
- ツールバーの中には、複数の「処理リンク」を配置することができる。
- <IMART type="imToolbarLeft"> タグに挟まれた「処理リンク」は、ツールバーの左側に配置される。  
(タグの詳細は「3.3.3.5.2 <IMART type="imToolbarLeft">」を参照)
- <IMART type="imToolbarRight"> タグに挟まれた「処理リンク」は、ツールバーの右側に配置される。  
(タグの詳細は「3.3.3.5.3 <IMART type="imToolbarRight">」を参照)
- <IMART type="imToolbarFrame"> タグに挟まれた範囲に、<IMART type="imToolbarRight"> タグおよび <IMART type="imToolbarLeft"> タグがどちらも存在しない場合の動作については未定義とする。
- <IMART type="imIcon"> タグにより、「アイコン＋処理名」の処理リンクを表示する。  
(タグの詳細は「3.3.3.5.4 <IMART type="imIcon">」を参照)

#### 3.3.3.2 制約

本タグを配置する場所は、HTML の<BODY> タグ内です。

#### 3.3.3.3 手順

ここでは、ツールバーを作成するための手順として、HTML ファイルに記述する内容を簡単に説明します。

※ ソースコードのサンプルについては、「3.3.3.4 使用例」を参照してください。

- (1) <IMART type="imDesignCss"> を記述して、スタイルシートの宣言を行います。  
(記述方法については「3.3.1 デザインスタイルシート」を参照)
- (2) <IMART type="imToolbarFrame"> タグを記述して、ツールバーの外枠を表示します。
- (3) <IMART type="imToolbarFrame"> タグに挟まれた範囲に、<IMART type="imToolbarLeft"> タグを記述します。(ツールバーの左側に配置する「処理リンク」がない場合は、省略可能。)
- (4) <IMART type="imToolbarFrame"> タグに挟まれた範囲に、<IMART type="imToolbarRight"> タグを記述します。(ツールバーの右側に配置する「処理リンク」がない場合は、省略可能。)
- (5) <IMART type="imIcon"···> タグを、<IMART type="imToolbarLeft"> タグ または、<IMART type="imToolbarRight"> タグに挟まれた範囲に記述して、「アイコン＋処理名」の処理リンクを表示します。
- (6) 「処理リンク」をツールバーの左側に表示するときは、<IMART type="imToolbarLeft"> タグに挟まれた範囲に <IMART type="imIcon"···> タグを記述してください。
- (7) 「処理リンク」をツールバーの右側に表示するときは、<IMART type="imToolbarRight"> タグに挟まれた範囲に <IMART type="imIcon"···> タグを記述してください。

## 3.3.3.4 使用例

ツールバーを表示するためのサンプルを、以下に示します。

HTML のサンプル

```

<HTML>
  <HEAD>
    <IMART type = "imDesignCss"></IMART>
  </HEAD>
  <BODY>
    <!-- タイトルバー表示 -->
    :
    (省略)
    :
    <!-- ツールバー表示 -->
    <IMART type = "imToolbarFrame">
      <IMART type = "imToolbarLeft">
        <IMART type = "imIcon"
          name = "新規登録"
          icon = "images/standard/new.gif"
          href = "javascript:onAddAction();"
        </IMART>
        <IMART type = "imIcon"
          name = "検索条件"
          icon = "images/standard/search.gif"
          href = "javascript:onSearchAction();"
        </IMART>
      </IMART>
      <IMART type = "imToolbarRight">
        <IMART type = "imIcon"
          name = "戻る"
          icon = "images/standard/arrow_left.gif"
          href = "javascript:onBackAction();"
        </IMART>
        <IMART type = "imIcon"
          name = "最新情報"
          icon = "images/standard/refresh.gif"
          href = "javascript:onReloadAction();"
        </IMART>
      </IMART>
    </IMART>
  </BODY>
</HTML>

```

手順(1)

手順(5) (6)

手順(3)

手順(5) (6)

手順(2)

手順(5) (7)

手順(4)

手順(5) (7)

表示結果



### 3.3.3.5 ツールバーを作成するために必要な<IMART>タグ

この章では、ツールバーを作成するために必要な <IMART> タグの詳細を示します。

ツールバーは、以下の「表 3-4 ツールバーに必要な画面デザイン共通モジュール」に示す、<IMART> タグで構成されています。

表 3-4 ツールバーに必要な画面デザイン共通モジュール

参照		ページ
3.3.3.5.1	<IMART type="imToolbarFrame">	34
3.3.3.5.2	<IMART type="imToolbarLeft">	35
3.3.3.5.3	<IMART type="imToolbarRight">	36
3.3.3.5.4	<IMART type="imIcon">	37

#### 3.3.3.5.1 <IMART type="imToolbarFrame">

##### 3.3.3.5.1.1 詳細

- ツールバーの外枠を表示する。
- ツールバーの表示色は、テーマによって自動に設定される。

##### 3.3.3.5.1.2 制約

本タグを配置する場所は、HTML の<BODY> タグ内です。

##### 3.3.3.5.1.3 属性

<IMART type="imToolbarFrame"> に指定する属性を、以下に示します。

- 属性 type には IMART タグ名「imToolbarFrame」を指定する。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imToolbarFrame"



3.3.3.5.2 <IMART type="imToolbarLeft">

3.3.3.5.2.1 詳細

- ツールバー内の 左側 に「処理リンク」を表示させたいときに使用する。  
ツールバーの左側に表示する「処理リンク」がないときは、省略可能である。
- 必ず <IMART type="imToolbarFrame">・・・</IMART> に挟まれた範囲に記述すること。
- <IMART type="imToolbarLeft">・・・</IMART> に挟まれた範囲に記述された「処理リンク」は、ツールバー内で左寄せに表示される。
- 本タグに挟まれた範囲には、複数の「処理リンク」を記述することができる。
- 「処理リンク」には、<IMART type="imIcon"> タグ（詳細は「3.3.3.5.4 <IMART type="imIcon">」を参照）だけでなく、通常のHTMLタグ<A href="XXXX"></A> や、<IMART type="link"> なども記述することが可能である。

3.3.3.5.2.2 制約

本タグを単独で利用することはできない。  
<IMART type="imToolbarFrame"> タグの内部タグとして利用すること。  
(タグの詳細は、「3.3.3.5.1 <IMART type="imToolbarFrame">」を参照)  
本タグは<IMART type="imToolbarRight">・・・</IMART> タグの前に記述すること。  
(タグの詳細は、「3.3.3.5.3 <IMART type="imToolbarRight">」を参照)

3.3.3.5.2.3 属性

- <IMART type="imToolbarLeft"> に指定する属性を、以下に示します。
- 属性 type には IMART タグ名「imToolbarLeft」を指定する。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imToolbarLeft"

### 3.3.3.5.3 <IMART type="imToolbarRight">

#### 3.3.3.5.3.1 詳細

- ツールバー内の 右側 に「処理リンク」を表示させたいときに使用する。  
ツールバーの右側に表示する「処理リンク」がないときは、省略可能である。
- 必ず <IMART type="imToolbarFrame">...</IMART> に挟まれた範囲に記述すること。
- <IMART type="imToolbarRight">...</IMART> に挟まれた範囲に記述された「処理リンク」は、ツールバー内で右寄せに表示される。
- 本タグに挟まれた範囲には、複数の「処理リンク」を記述することができる。
- 「処理リンク」には、<IMART type="imIcon"> タグ（詳細は「3.3.3.5.4 <IMART type="imIcon">」を参照）だけでなく、通常のHTMLタグ<A href="XXXX"></A> や、<IMART type="link"> なども記述することが可能である。

#### 3.3.3.5.3.2 制約

本タグを単独で利用することはできない。

<IMART type="imToolbarFrame"> タグの内部タグとして利用すること。

（タグの詳細は、「3.3.3.5.1 <IMART type="imToolbarFrame">」を参照）

本タグは<IMART type="imToolbarLeft">...</IMART> タグの後に記述すること。

（タグの詳細は、「3.3.3.5.2 <IMART type="imToolbarLeft">」を参照）

#### 3.3.3.5.3.3 属性

<IMART type="imToolbarRight"> に指定する属性を、以下に示します。

- 属性 type には IMART タグ名「imToolbarRight」を指定する。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imToolbarRight"

### 3.3.3.5.4 <IMART type="imIcon">

#### 3.3.3.5.4.1 詳細

- ツールバー内に「処理リンク」を表示する。
- この <IMART type="imIcon"> タグにより、「アイコン+処理名」の処理リンクが表示される。
- 「アイコン」のみ、または「処理名」のみの表示も可能である。
- 属性 icon により、表示するアイコンを指定できる。
- 属性 name により、表示する処理名を指定できる。
- 処理を選択された時のリンク先として、js の関数名や URL を指定することができる。  
また、リンク先の表示ウィンドウを指定することができる。
- 本タグを <IMART type="imToolbarLeft"> タグに挟まれた範囲に記述すると、ツールバー内で左寄せに表示される。
- 本タグを <IMART type="imToolbarRight"> タグに挟まれた範囲に記述すると、ツールバー内で右寄せに表示される。
- 本タグを記述した順番に、ツールバーの左から「処理リンク」が配置される。

#### 3.3.3.5.4.2 制約

属性 href を指定した場合、<A>タグとして作成されます。

属性 その他 を指定する場合、属性 href が設定されている必要があります。

#### 3.3.3.5.4.3 属性

<IMART type="imIcon"> に指定する属性を、以下に示します。

- 属性 type には IMART タグ名「imIcon」を指定する。
- 属性 name には「処理名」を指定する。  
属性 name を未指定にした場合は、「アイコン」だけが表示される。
- 属性 icon には「アイコン」のファイル名を、ドキュメントルートから相対パスで指定する。  
属性 icon を未指定にした場合は、「処理名」だけが表示される。
- 属性 name、および 属性 icon がどちらも未指定の場合は、処理リンクの配置を無視する。
- 属性 href には処理を選択された際のリンク先を指定する。  
指定する値は、js の関数名 (javascript:XXXXXX();) や、URL などが可能である。  
属性 href を未指定にした場合は、リンク表示ではなく、ただのテキスト表示になる。
- 属性 target にはリンク先の表示ウィンドウを指定する。  
指定する値は、HTML の target 属性に従う。  
属性 target を未指定にした場合は、同ウィンドウに表示される。
- 属性 alt には画像の変わりに表示される文字列を指定する。
- 属性 その他 は<A>タグに設定する各属性値が指定できる。  
指定する値はHTMLの<A>タグ属性に従う。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imIcon"
name	処理名		—	name="新規登録"
icon	アイコンのファイル名 (ドキュメントルートからの相対パス)		—	icon="images/standard/new_item.gif"
href	処理選択時のジャンプ先 (js 実行関数名や、URL など)		—	href="javascript:onAddAction()" href="http://www.intra-mart.co.jp"
target	表示ウィンドウ		—	target="_top"
alt	画像の変わりに表示される文字列			alt="新規登録"
その他	<A>タグに設定する各属性値		—	tabindex="1"

### 3.3.4 入力項目

ここでは、登録系画面や、更新・削除系画面などで頻繁に使用される「項目名称＋入力フィールド」をまとめて、「入力項目」として説明しています。

#### 3.3.4.1 機能詳細

- 登録系画面や、更新・削除系画面で、入力項目を作成する際に使用するタグである。
- `<IMART type="imItemName">` により、項目名称を表示する。  
(タグの詳細は「3.3.4.5.1 項目名称」を参照)
- `<IMART type="imInputTd">` により、入力フィールドを表示する。  
(タグの詳細は「3.3.4.5.2 入力フィールド」を参照)
- `<IMART type="imSelectTd">` により、選択部品(セレクトボックス)を表示する。  
(タグの詳細は「3.3.4.5.3 選択ボックス」を参照)

#### 3.3.4.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。  
HTML の`<TD>`タグとして作成されるため、`<TABLE><TR>~</TR></TABLE>` 内に記述してください。  
また、デザインを統一するために本タグを使用する際には、必ず「3.3.4.4 使用例」に示す  
`<TABLE><TR>~</TR></TABLE>`の中に記述してください。

#### 3.3.4.3 手順

ここでは、入力項目を作成するための手順として、HTML ファイルに記述する内容を簡単に説明します。

- (1) `<IMART type="imDesignCss">` を記述して、スタイルシートの宣言を行います。  
(記述方法については「3.3.1 デザインスタイルシート」を参照)
- (2) デザインを統一するため、`<TABLE><TR>~</TR></TABLE>` を記述します。
- (3) `<IMART type="imItemName">` タグを (2) に挟まれた範囲に記述して、項目名称を表示します。
- (4) `<IMART type="imInputTd">` タグを (2) に挟まれた範囲に記述して、入力フィールドを表示します。
- (5) 選択ボックスに設定する「表示データ」を、ファンクション・コンテナ(.js ファイル)の `init()`関数内で構築します。
- (6) `<IMART type="imSelectTd">` タグを (2) に挟まれた範囲に記述して、選択ボックスを表示します。

※ ソースコードのサンプルについては、「3.3.4.4 使用例」を参照してください。

## 3.3.4.4 使用例

入力項目を表示するためのサンプルを、以下に示します。

HTML のサンプル

<HTML>		
<HEAD>		
<IMART type = "imDesignCss"></IMART>		手順(1)
</HEAD>		
<BODY>		
<!-- 入力項目表示 -->		
<TABLE width="100%" border="0" cellpadding="0" cellspacing="0">		
<TR>		
<TD align="center">		
<TABLE class="edit">		
<TR>		
<IMART type = "imItemName" name = "ユーザ ID"></IMART>		手順(3)
<IMART type = "imInputTd" style = "text" name = "userId" size = "30" value = "User00011" readonly></IMART>		手順(4)
</TR>		
<TR>		
<IMART type = "imItemName" name = "ユーザ名" require></IMART>		手順(3)
<IMART type = "imInputTd" style = "text" name = "userName" size = "50" value = "ユーザ.00011"></IMART>		手順(4)
</TR>		
<TR>		
<IMART type = "imItemName" name = "パスワード" require></IMART>		手順(3)
<IMART type = "imInputTd" style = "password" name = "password" size = "40"></IMART>		手順(4)
</TR>		手順(2)
<TR>		
<IMART type = "imItemName" name = "出身地"></IMART>		手順(3)
<IMART type = "imSelectTd" list = listData ... 次ページの「js のサンプル」参照 name = "hometown" size = "1"></IMART>		手順(6)
</TR>		
<TR>		
<IMART type="imItemName" name="転送ファイル"></IMART>		手順(3)
<IMART type = "imInputTd" style = "file" name = "forwarFile" size = "50"></IMART>		手順(6)
</TR>		
</TABLE>		
</TD>		
</TR>		
</TABLE>		
:		
:		
</BODY>		
</HTML>		

## js のサンプル

```
// バインド変数宣言
var listData; // 選択ボックスの表示データ

// ページの初期化関数
function init() {

    listData = new Object();

    listData["tokyo"] = "東京都";
    listData["kanagawa"] = "神奈川県";
    listData["saitama"] = "埼玉県";
    listData["chiba"] = "千葉県";

}
```

手順(5)

もしくは

```
// バインド変数宣言
var listData; // 選択ボックスの表示データ

// ページの初期化関数
function init() {

    listData = new Object();

    listData.tokyo = "東京都";
    listData.kanagawa = "神奈川県";
    listData.saitama = "埼玉県";
    listData.chiba = "千葉県";

}
```

手順(5)

## 表示結果

ユーザID	<input type="text" value="User00011"/>
ユーザ名(必須)	<input type="text"/>
パスワード(必須)	<input type="password"/>
出身地	<input type="text" value="東京都"/> ▼
転送ファイル	<input type="text"/> <input type="button" value="参照..."/>

### 3.3.4.5 入力項目を作成するために使用する<IMART>タグ

#### 3.3.4.5.1 項目名称

##### 3.3.4.5.1.1 詳細

- <IMART type="imItemName"> により、入力項目の「項目名称」を表示する。
- 登録系画面や、更新・削除系画面で、入力項目の項目名称を記述する際に使用する。
- 属性 name に指定した値が、項目名称として表示される。
- 属性 require を指定すると、「必須項目」として表示される。  
「必須項目」の場合は、項目名称が太字で表示され、赤色で「(必須)」の文字が記述される。
- 属性 require が未指定の場合は、「通常項目」として表示される。

##### 3.3.4.5.1.2 制約

HTML の<TD>タグとして作成されるため、<TABLE><TR>～</TR></TABLE> 内に記述してください。  
また、デザインを統一するために本タグを使用する際には、必ず「3.3.4.4 使用例」に示す  
<TABLE><TR>～</TR></TABLE>の中に記述してください。

##### 3.3.4.5.1.3 属性

<IMART type="imItemName"> に指定する属性を、以下に示します。

- 属性 type には IMART タグ名「imItemName」を指定する。
- 属性 name には「項目名称」を指定する。
- 属性 require を指定すると、「必須項目」として表示される。  
属性 require を未指定にした場合は、「通常項目」として表示される。
- その他、<TD> タグに設定する各属性値が指定できる。  
指定する値は、HTML の <TD>タグ属性に従う。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imItemName"
name	項目名称	○	—	name="ユーザ名"
require	必須項目の指定		—	require
その他	<TD>タグに設定する各属性値			width="150" colspan="2"

[記述例 ①]：属性 require を指定して、必須項目にした場合

```
<IMART type="imItemName" name="ユーザ名" require></IMART>
```

ユーザ名(必須)

[記述例 ②]：属性 require を未指定して、通常項目にした場合

```
<IMART type="imItemName" name="ユーザ名"></IMART>
```

ユーザ名



#### 3.3.4.5.2.1 詳細

#### 3.3.4.5.2.1 詳細

- 属性 style="text" を指定した場合は、テキスト入力フィールドが表示される。

■ 属性 style="textarea" 以外は、HTML の<INPUT>タグとして作成される。

#### 3.3.4.5.2.2 制約

<TABLE><TR>～</TR></TABLE>の中に記述してください。

## 3.3.4.5.2.3 属性

<IMART type="imInputTd"> に指定する属性を、以下に示します。

## 【共通属性】

- 属性 type には IMART タグ名「imInputTd」を指定する。
- 属性 style には、表示する部品の「種類」を指定する。  
(属性 style="textarea" 以外は、HTML の<INPUT>タグの type 属性同様)
- 属性 name には「名前」を指定する。  
(HTML の<INPUT> および <TEXTAREA>タグの name 属性同様)
- 属性 colspan には、<TD>タグに設定する「colspan 属性」を指定する。
- 属性 rowspan には、<TD>タグに設定する「rowspan 属性」を指定する。
- 属性 align には、<TD>タグに設定する「align 属性」を指定する。
- 属性 tdclass には、<TD>タグに設定する「class 属性」を指定する。
- 属性 class には、<INPUT> および <TEXTAREA> タグに設定する「class 属性」を指定する。  
属性 type="text" の場合、この属性 class に「class="default\_right"」を指定すると、テキスト入力フィールドの入力文字が右寄せで表示される。
- その他、<TD> タグに設定する各属性値が指定できる。  
指定する値は、HTML の <TD>タグ属性に従う。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imInputTd"
style	部品の種類 ・text: テキスト入力フィールド ・password: パスワード入力フィールド ・file: ファイル入力フィールド ・checkbox: チェックボックス ・radio: ラジオボタン ・textarea: テキストエリア入力フィールド	○	—	style="text" style="password" style="file" style="checkbox" style="radio" style="textarea"
name	名前	○	—	name="userId"
colspan	<TD>タグの colspan 属性		—	colspan="2"
rowspan	<TD>タグの rowspan 属性		—	rowspan="4"
align	<TD>タグの align 属性 (横方向表示位置) ・center: 中央表示 ・right: 右寄せ ・left: 左寄せ		—	align="center"
tdclass	<TD>タグの class 属性		default	tdclass ="default_right"
class	<INPUT>タグおよび、<TEXTAREA>の class 属性		default	class ="default_right"
その他	<TD>タグに設定する各属性値			width="150" colspan="2"

【style="text" の属性】・・・テキスト入力フィールド

- 属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)
- 属性 value には、入力フィールドの「初期値」を指定する。(HTML の<INPUT>タグの value 属性同様)
- 属性 readonly を指定した場合は、入力フィールドへの入力が不可になります。  
(HTML の<INPUT>タグの readonly 属性同様)  
属性 readonly を未指定にした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="50"
value	初期値		—	value="User00011"
readonly	入力不可の指定		—	readonly

〔記述例 ①〕：属性 readonly を未指定にして、入力可能にした場合

```
<IMART type="imInputTd" style="text"
name="userID" size="30" value="User00011"></IMART>
```

User00011

〔記述例 ②〕：属性 readonly を指定して、入力不可にした場合

```
<IMART type="imInputTd" style="text"
name="userID" size="30" value="User00011" readonly></IMART>
```

User00011

〔記述例 ③〕：属性 class に "default\_right" を指定した場合

```
<IMART type="imInputTd" style="text" class="default_right"
name="sort_no" size="10" value="200"></IMART>
```

200

【style="password" の属性】・・・パスワード入力フィールド

- 属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)
- 属性 value には、入力フィールドの「初期値」を指定する。(HTML の<INPUT>タグの value 属性同様)
- 属性属性 readonly を指定した場合は、入力フィールドへの入力が不可になります。  
(HTML の<INPUT>タグの readonly 属性同様)  
属性 readonly を未指定にした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="40"
value	初期値		—	value="Password00001"
readonly	入力不可の指定		—	readonly

【記述例 ①】：属性 readonly を未指定にして、入力可能にした場合

```
<IMART type="imInputTd" style="password"
name="pass" size="40" value="Pass00011"></IMART>
```

【記述例 ②】：属性 readonly を指定して、入力不可にした場合

```
<IMART type="imInputTd" style="password"
name="pass" size="40" value="Pass00011" readonly></IMART>
```

【style="file" の属性】・・・ファイル入力フィールド

- 属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="50"

【記述例】：属性 size="50" を指定して表示

```
<IMART type="imInputTd"
style="file" name="forward_file" size="50"></IMART>
```

## 【style="checkbox" の属性】・・・チェックボックス

- 属性 caption には、チェックボックス横に表示する「名称」を設定する。
- 属性 checked を指定した場合は、チェックボックスがチェックされた状態で表示される。  
(HTML の<INPUT>タグの checked 属性同様)  
属性 checked を未指定にした場合は、チェックがされていない状態で表示される。

属性	説明	必須	デフォルト値	書式
caption	名称		—	caption="チェック 1"
checked	チェック状態の指定		—	checked

【記述例 ①】：属性 caption="チェック 1" と指定して表示

```
<IMART type="imInputTd" style="checkbox"
name="check1" caption="チェック 1"></IMART>
```

☐ チェック1

【記述例 ②】：属性 checked を指定して、あらかじめチェックされた状態にした場合

```
<IMART type="imInputTd" style="checkbox"
name="check1" caption="チェック 1" checked></IMART>
```

☒ チェック1

## 【style="radio" の属性】・・・ラジオボタン

- 属性 caption には、ラジオボタン横に表示する「名称」を設定する。
- 属性 checked を指定した場合は、ラジオボタンがチェックされた状態で表示される。  
(HTML の<INPUT>タグの checked 属性同様)  
属性 checked を未指定にした場合は、チェックがされていない状態で表示される。

属性	説明	必須	デフォルト値	書式
caption	名称		—	caption="ラジオ 1"
checked	チェック状態の指定		—	checked

【記述例 ①】：属性 caption="ラジオ 1" と指定して表示

```
<IMART type="imInputTd" style="radio"
name="radio1" caption="ラジオ 1"></IMART>
```

☐ ラジオ1

【記述例 ②】：属性 checked を指定して、あらかじめチェックされた状態にした場合

```
<IMART type="imInputTd" style="radio"
name="radio1" caption="ラジオ 1" checked></IMART>
```

☒ ラジオ1

## 【style="textarea" の属性】・・・テキストエリア表示

- 属性 value には、表示する「文字列」を指定する。
- 属性 cols には、テキストエリアの「横幅」を指定する。  
(HTML の<TEXTAREA>タグの cols 属性同様)
- 属性 rows には、テキストエリアの「縦幅(行数)」を指定する。  
(HTML の<TEXTAREA>タグの rows 属性同様)
- 属性属性 readonly を指定した場合は、入力フィールドへの入力が不可になります。  
(HTML の<TEXTAREA>タグの readonly 属性同様)  
属性 readonly を未指定にした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
value	表示する文字列		—	value="ここに長い・・・"
cols	テキストエリアの横幅 (<TEXTAREA>の cols 属性)		—	cols="40"
rows	テキストエリアの縦幅(行数) (<TEXTAREA>の rows 属性)		—	rows="5"
readonly	入力不可の指定		—	readonly

〔記述例〕：属性 cols="40"、属性 rows="5" を指定して表示

```
<IMART type="imInputTd" style="textarea" name="comment"
cols="40" rows="5" value="ここに長い文章が書けます。"></IMART>
```

〔記述例〕：属性 cols="40"、属性 rows="5" を指定して表示

```
<IMART type="imInputTd" style="textarea" name="comment"
cols="40" rows="5" value="読み出し専用の入力フォーム" readonly></IMART>
```

### 3.3.4.5.3 選択ボックス

#### 3.3.4.5.3.1 詳細

- `<IMART type="imSelectTd">` により、入力項目の選択ボックスを表示する。
- 登録系画面や、更新・削除系画面で、入力項目の選択ボックスを記述する際に使用する。
- 既存の画面デザイン共通モジュール `<IMART type="select">` とほぼ同じ機能であり、画面デザインを統一するためにデザインが変更されている。

#### 3.3.4.5.3.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。

HTML の`<TD>`タグとして作成されるため、`<TABLE><TR>~</TR></TABLE>` 内に記述してください。

また、デザインを統一するために本タグを使用する際には、必ず「3.3.4.4 使用例」に示す`<TABLE><TR>~</TR></TABLE>`の中に記述してください。

#### 3.3.4.5.3.3 属性

- `<IMART type="imSelectTd">` に指定する属性を、以下に示します。
- 属性 `type` には IMART タグ名「`imSelectTd`」を指定する。
- 属性 `list` には「表示リスト」をオブジェクトで指定する。  
属性 `list` に指定されたオブジェクトの「プロパティ名」が コンボボックス内の各要素(`<OPTION>`タグ)の `value` 値になり、「該当するプロパティに格納されている値」が コンボボックス内の要素の画面上での表示名になる。(オブジェクトに格納されている値が文字列以外の場合の動作は保証外である。)  
属性 `list` に指定したオブジェクトに格納されているすべての値が、コンボボックス内の表示要素になる。

【属性 `list` に指定するオブジェクトの例】

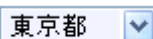
listData["tokyo"] = "東京都";	listData.tokyo = "東京都";
↓            ↓	↓            ↓
value 値    表示名	value 値    表示名

- 属性 `selected` には、コンボボックスの初期選択(初期表示)値を指定する。  
初期選択状態にする値のキー(属性 `list` に指定したオブジェクトに該当するプロパティ名)を指定することで、初期選択状態にすることができる。  
属性 `selected` に指定した値に該当するキーが、属性 `list` に存在しない場合は、どの値も選択状態にはならない。  
また、初期選択状態とするキーを同時に複数個指定する場合は、属性 `selected` に対して配列を指定することで可能となる。  
その際、配列内の各要素の値として、初期選択状態にするキー(属性 `list` に格納したオブジェクトの該当するプロパティ名)を格納すること。
- 属性 `blank` には、コンボボックス内に空データを表示させる「位置」を指定する。  
属性 `blank` に`"top"`を指定した場合は、ボックス内の先頭位置に空データが表示される。  
属性 `blank` に`"bottom"`を指定した場合は、ボックス内の最後尾位置に空データが表示される。
- 属性 `size` には、一度に表示する選択肢の行数を指定する。(HTML の`<SELECT>`タグの `size` 属性同様)  
属性 `size` を未指定にした場合は、デフォルト値「1」が設定される。
- 属性 `multiple` を指定した場合は、複数行選択が可能になる。(HTML の`<SELECT>`タグの `multiple` 属性が付加される)  
属性 `multiple` を未指定にした場合は、複数行選択が不可になる。
- その他、`<TD>` タグに設定する各属性値が指定できる。  
指定する値は、HTML の `<TD>`タグ属性に従う。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imSelectTd"
list	表示リスト	○	—	list=listData
selected	初期選択値の設定		—	selected =selecteData
blank	空データを表示する位置 ・top : 先頭位置に空データ ・bottom: 最後尾位置に空データ		—	blank="top"
name	名前		—	name="hometown"
size	表示する行数		1	size="5"
multiple	複数行選択の可否		—	multiple
その他	<TD>タグに設定する各属性値			width="150" colspan="2"

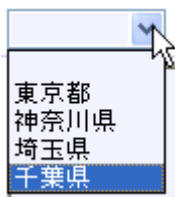
〔記述例 ①〕 ※ listDataの設定方法については、「3.3.4.4 使用例」を参照

```
<IMART type="imSelectTd" list=listData name="hometown"></IMART>
```



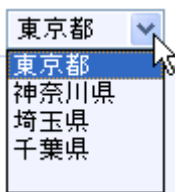
〔記述例 ②〕 : 属性 blank="top" に指定した場合 → 先頭に空データが挿入

```
<IMART type="imSelectTd"
list=listData name="hometown" blank="bottom"></IMART>
```



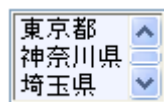
〔記述例 ③〕 : 属性 blank="bottom" に指定した場合 → 最後尾に空データが挿入

```
<IMART type="imSelectTd"
list=listData name="hometown" blank="bottom"></IMART>
```



〔記述例 ④〕 : 属性 size="3" に指定した場合

```
<IMART type="imSelectTd" list=listData name="hometown" size="3"></IMART>
```













### 3.3.5 リストコントロール

#### 3.3.5.1 機能詳細

- <IMART type="imListControl"> により、各種一覧に対する ソート切り替えコントロールと、ページ切り替えコントロールを表示する。
- ソート および、ページ切り替えコントロールはリンク表示となっている。  
各コントロールのリンクがクリックされると、JavaScript関数 が呼び出されるため、本タグを使用した場合は、各コントロールに対してJavaScript関数の処理を記述する必要がある。  
(詳細は「3.3.5.5 デフォルトの必須JavaScript関数」を参照)
- ソート切り替えコントロールには、昇順ソート(有効および無効)、降順ソート(有効および無効)の矢印リンクが用意されている。

ソート対象	昇順アイコン	降順アイコン
昇順	 (有効)	 (無効)
降順	 (無効)	 (有効)

- ソート切り替えコントロールは、属性の指定により表示、非表示を設定することができる。
- ページ切り替えコントロールには、 (前のページ)、 (次のページ)、 (最初のページ)、 (最後のページ) の矢印リンクが用意されている。
- 一覧で表示しているページによって、矢印アイコンの表示および、非表示を切り替える。

表示ページ	最初のページ アイコン	前のページ アイコン	次のページ アイコン	最後のページ アイコン
最初のページ	非表示	非表示		
最後のページ			非表示	非表示

- 一覧の表示件数が [全件数<1ページの表示数] の場合は、ページ切り替えコントロールは表示されない。(ページ件数の表示のみ)



#### 3.3.5.2 制約

本タグは、HTML の<TABLE>タグとして作成されます。

本タグを配置する場所は、HTML の<BODY> タグ内です。

### 3.3.5.3 属性

<IMART type="imListControl"> に指定する属性を、以下に示す。

- 属性 type には IMART タグ名「imListControl」を指定する。
- 属性 maxRecord には一覧に表示するデータの「全件数」を指定する。
- 属性 currentPage には「表示するページ番号」を指定する。  
属性 currentPage を未指定にした場合は、1 ページ目が表示される。
- 属性 pageLine には「1 ページの表示数」を指定することができる。  
属性 pageLine を未指定にした場合は、1 ページに表示される件数は 10 件となる。
- 「昇順／降順」切り替えコントロールは 表示、または 非表示を指定することができる。  
属性 sortDisplay に "true" を指定した場合は表示、"false" を指定した場合は非表示になる。  
属性 sortDisplay が未指定の場合は、「昇順／降順」切り替えコントロールが表示される。
- 属性 sortOrder には「ソート基準方向」を指定する。  
現在のソート方向が昇順の場合は"asc"、降順の場合は"desc"を指定する。  
属性 sortOrder を未指定にした場合は、デフォルト値の "asc"（昇順）となる。
- 属性 prevPageFunc には、前ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 currentPage]-1')" が設定される。
- 属性 nextPageFunc には、次ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 currentPage]+1')" が設定される。
- 属性 firstPageFunc には、最初ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('1')" が設定される。
- 属性 lastPageFunc には、最終ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 maxRecord]/[属性 pageLine]')" が設定される。
- 属性 ascSortFunc には、昇順ソートリンク  (有効) および  (無効) が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onSortLinkFunc('asc')" が設定される。
- 属性 descSortFunc には、降順ソートリンク  (有効) および  (無効) が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onSortLinkFunc('desc')" が設定される。

属性	説明	必須	デフォルト値	書式
type	IMART タグ名	○	—	type="imListControl"
maxRecord	全件数	○	—	maxRecord="50"
currentPage	表示するページ番号		1	currentPage="1"
pageLine	1 ページの表示数		10	pageLine="20"
sortDisplay	「昇順／降順」の有無 (true:表示/false:非表示)		true	sortDisplay="true"
sortOrder	ソート基準方向 (asc:昇順／desc:降順)		asc	sortOrder="desc"
prevPageFunc	前ページ選択時の js 関数		上記参照	prevPageFunc= "javascript:prevPage();"
nextPageFunc	次ページ選択時の js 関数		上記参照	nextPageFunc= "javascript:nextPage();"
firstPageFunc	最初ページ選択時の js 関数		上記参照	firstPageFunc= "javascript:firstPage();"
lastPageFunc	最終ページ選択時の js 関数		上記参照	lastPageFunc= "javascript:lastPage();"
ascSortFunc	昇順ソート選択時の js 関数		上記参照	ascSortFunc= "javascript:ascSort();"
descSortFunc	降順ソート選択時の js 関数		上記参照	descSortFunc= "javascript:descSort();"

### 3.3.5.4 使用例

リストコントロール(ソート切り替え、ページ切り替え)を表示するためのサンプルを、以下に示します。

HTML のサンプル

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
  <SCRIPT LANGUAGE="JavaScript">

    // ページ切り替え<>がクリックされた時の処理
    function onPageLinkFunc(num)
    {
      :
      // ...ここには各自で処理を記述する...
      :
    }

    // 昇順／降順(ソート条件)をクリックした時の処理
    function onSortLinkFunc(mode)
    {
      :
      // ...ここには各自で処理を記述する...
      :
    }
  </SCRIPT>
</HEAD>
<BODY>
  <TABLE border="0" width="100%" cellpadding="0" cellspacing="2">
    <TR>
      <TD>
        <!-- リストコントロール -->
        <IMART type="imListControl"
              maxRecord="20"
              currentPage="1"
              pageLine="5"
              sortOrder="asc">
        </IMART>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

表示結果



### 3.3.5.5 デフォルトの必須 JavaScript 関数

ソート切り替えコントロール、およびページ切り替えコントロールのリンクがクリックされた際に呼び出される、デフォルトの JavaScript 関数の詳細を以下に示します。

本タグを使用する際に、以下の属性を未指定にした場合は、各自で必ずデフォルトの JavaScript 関数の処理を記述してください。

- 属性 prevPageFunc
- 属性 nextPageFunc
- 属性 firstPageFunc
- 属性 lastPageFunc
- 属性 ascSortFunc
- 属性 descSortFunc

※ 必須の JavaScript 関数を記述しない場合の動作については、保証外とします。

※ コーディング方法などの詳細については、「3.3.5.4 使用例」を参照してください。

※ 上記の属性を指定した場合は、デフォルトの JavaScript 関数の処理を記述する必要はありません。

### 3.3.5.5.1 ソート切り替え

ソート切り替えコントロールで、「昇順」および「降順」のリンクがクリックされた場合、JavaScript 関数の `onSortLinkFunc()` が呼び出されます。

以下に、その詳細を示します。

- 関数名

`onSortLinkFunc( mode )`

- 概要

ソート切り替えコントロール（昇順or降順）がクリックされた際に、呼び出される関数です。

引数 `mode` の値により「昇順」、「降順」のどちらが選択されたかの情報を、取得することができます。

- 引数

変数	内容	値	条件
mode	ソート基準方向	asc	「昇順」リンクが選択された場合
		desc	「降順」リンクが選択された場合

### 3.3.5.5.2 ページ切り替え

ページ切り替えコントロールで、各矢印のリンクがクリックされた場合、JavaScript 関数の `onPageLinkFunc()` が呼び出されます。

以下に、その詳細を示します。

- 関数名

`onPageLinkFunc( num )`

- 概要

ページ切り替えコントロールの、各矢印リンクがクリックされた際に、呼び出される関数です。

引数 `num` の値により、次に表示するページ番号の情報を、取得することができます。

- 引数

変数	内容	値	条件
num	次のページ番号	1～n	

### 3.3.6 リストヘッダ

#### 3.3.6.1 機能詳細

- `<IMART type="imListHeader">` により、各種一覧に対する ヘッダを表示する。
- 本タグで生成された一覧のヘッダ項目名は、リンク表示となっている。  
各項目のリンクがクリックされると、JavaScript関数が呼び出されるため、本タグを使用した場合は、各項目リンクに対してJavaScript関数の処理を記述する必要がある。  
(詳細は「3.3.6.5 必須のJavaScript関数」を参照)
- ヘッダの項目名は、属性の指定によりリンク表示と、通常表示(ただの文字列)を設定することができる。
- ソートキーになっている項目は黄色表示、通常の項目は水色表示になる。
- ヘッダ情報(項目名など)は、ファンクション・コンテナ(.js ファイル)で設定する。

#### 3.3.6.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。  
また、HTML の`<TR>`タグとして作成されるため、`<TABLE>~</TABLE>` 内に記述してください。  
その際、`<TABLE>`タグには `class="list_border_bg"` または `class="table_border_bg"` を指定してください。

表全体の枠 (横幅 100% 固定)	<code>&lt;TABLE class="list_border_bg"&gt;</code>
表全体の枠 (横幅任意指定)	<code>&lt;TABLE class="table_border_bg"&gt;</code>

#### 3.3.6.3 属性

`<IMART type="imListHeader">` に指定する属性を、以下に示します。

- 属性 `type` には IMART タグ名「`imListHeader`」を指定する。
- 属性 `headerData` には、「ヘッダ情報オブジェクト」の配列を指定する。  
※ ヘッダ情報オブジェクトの構成については、「3.3.6.3.1 ヘッダ情報オブジェクト」を参照してください。

属性	説明	必須	デフォルト値	書式
<code>type</code>	IMART タグ名	○	—	<code>type="imListHeader"</code>
<code>headerData</code>	ヘッダ情報オブジェクト	○	—	<code>headerData=aHeader</code>

## 3.3.6.3.1 ヘッダ情報オブジェクト

<IMART type="imListHeader"> を使用してリストヘッダを作成する際に、属性 headerData に指定するオブジェクトです。

ヘッダ情報オブジェクトの構成を、以下の「表 3-5 ヘッダ情報オブジェクトの構成」に示します。

表 3-5 ヘッダ情報オブジェクトの構成

プロパティ	設定内容	説明
name	項目名	ヘッダに表示する項目名を指定する。 未指定の場合は、空のヘッダが表示される。 また、項目 ID="checkbox" を指定した場合は、 <INPUT type="checkbox"> タグの name 属性となる。
key	項目 ID	項目 ID を指定する。 ここに指定した値は、プロパティ href を未指定にした場合の、デフォルトの js 関数へ渡される引数となる。 現在のソートキー項目を判定するための ID となるので、他項目との重複は不可とする。 項目 ID に "checkbox" を指定すると、ヘッダ内に「チェックボックス」が表示される。
sortEnable	ソート表示状態 true : リンク表示 false: 通常表示	項目名の表示状態を指定する。 未指定の場合は、デフォルト値「true」が設定される。 また、項目 ID="checkbox" を指定した場合は、ソート対象外となるため、通常表示「false」となる。
href	項目名選択時の js 関数	項目名がリンク表示のとき、リンクが選択された時の js 関数を指定する。 未指定の場合は、デフォルト値の「onSortHeadClick(項目 ID);」が設定される。 また、項目 ID="checkbox" を指定した場合は、「onClock イベントハンドラ」に設定する js 関数を指定すること。
status	現在のソートキー true : ソートキー false: ソートキーでない	現在のソートキーを指定する。 未設定の場合は、デフォルト値「false」が設定される。 また、項目 ID="checkbox" を指定した場合は、設定が無視され、自動的に「false」となる。
align	項目名の表示位置 center: 中央表示 left : 左寄せ right : 右寄せ	項目名の表示位置を指定する。 未指定の場合は、デフォルト値「center」が設定される。
その他	上記プロパティ以外の任意属性の設定値	上記プロパティ以外の属性を追加可能。各項目の<TD>タグに対する属性となる。



## 3.3.6.4 使用例

リストヘッダを表示するためのサンプルを、以下に示します。

## HTML のサンプル

```
<HTML>
<HEAD>
  <IMART type="imDesignCss"></IMART>
  <SCRIPT LANGUAGE="JavaScript">
    // 一覧の項目名リンクがクリックされた時の処理
    function onSortHeadClick(key)
    {
      // ...ここには各自で処理を記述する...
      :
    }

    // (1) で指定した JavaScript 関数の処理
    function onCheckBox()
    {
      // ...ここには各自で処理を記述する...
    }
  </SCRIPT>
</HEAD>
<BODY>
  <TABLE class="list_border_bg">
    <!-- リストヘッダ -->
    <IMART type="imListHeader" headerData=aHeader></IMART>
  </TABLE>
</BODY>
</HTML>
```

## js のサンプル

```
// バインド変数宣言
var aHeader; // ヘッダ項目情報

// ページの初期化関数
function init() {
  aHeader = new Array();

  aHeader[0] = new Object();
  aHeader[0].name = "checkCtrl";
  aHeader[0].key = "checkbox";
  aHeader[0].href = "onCheckBox()";
  ..... (1)

  aHeader[1] = new Object();
  aHeader[1].name = "ユーザコード";
  aHeader[1].key = "user_cd";
  aHeader[1].status = true;
  aHeader[1].href = "func1()";
  ..... (2)

  aHeader[2] = new Object();
  aHeader[2].name = "ユーザ名";
  aHeader[2].key = "name kana";
  ..... (3)

  aHeader[3] = new Object();
  aHeader[3].name = "会社／組織";
  aHeader[3].key = "division";
  aHeader[3].sortEnable = false;
  aHeader[3].align = "left";
  aHeader[3].width = "200";
  ..... (4)
}
```

表示結果

<input type="checkbox"/>	ユーザコード	ユーザ名	会社／組織
:	:	:	:
:	:	:	:
(1)	(2)	(3)	(4)

3.3.6.5 必須のJavaScript関数

リストヘッダの項目名リンクがクリックされた際に Call される、JavaScript 関数の詳細を以下に示す。  
本タグを使用する場合は、各自で必ず JavaScript 関数の処理を記述すること。

- ※ 必須の JavaScript 関数を記述しない場合の動作については、保証外とする。
- ※ コーディング方法などの詳細については、「3.3.6.4 使用例」を参照。

3.3.6.5.1 ソートの項目切り替え

リストヘッダの項目名リンクがクリックされた場合、JavaScript 関数の onSortHeadClick() が呼び出されます。  
以下に、その詳細を示す。

- 関数名  
onSortHeadClick( key )
- 概要  
リストヘッダの項目名リンクがクリックされた際に、呼び出される関数です。  
引数 key の値により、どのヘッダ項目が選択されたかの情報を、取得することができます。

■ 引数			
変数	内容	値	条件
key	ヘッダの項目 ID	(例) user_cd	リストのヘッダ項目リンクが選択された

※ 引数 key の値は、ヘッダ情報オブジェクトの key プロパティで指定した「ヘッダの項目 ID」。

## 3.4 JavaEE開発モデル

このドキュメント内では、prefix 属性を "imarttag" として説明を行います。  
記述例を以下に示します。

[記述例]

```
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
```

### 3.4.1 デザインスタイルシート

#### 3.4.1.1 機能詳細

- <imarttag:imartDesignCss>により、現在指定されているテーマ(色)のスタイルシートを設定する。
- ServerManager 配下の conf/system.xml ファイルの system/color-config/color-pattern タグ内で設定したCSSファイルが利用可能になります。  
system/color-config/color-pattern に複数のCSSファイルを設定することも可能です。

#### 3.4.1.2 制約

本タグを配置する場所は、HTML の<HEAD> タグ内です。

#### 3.4.1.3 属性

<imarttag:imartDesignCss> に指定する属性は、ありません。

#### 3.4.1.4 使用例

デザインスタイルシートを設定するためのサンプルを、以下に示します。

JSP のサンプル

```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
:
<HTML>
  <HEAD>
    <imarttag:imartDesignCss />
  </HEAD>
  <BODY>
    :
    :
  </BODY>
</HTML>
```

#### 3.4.1.5 注意点

各画面(ページ)を作成する JSP ファイルの <HEAD> タグ内には、  
必ず デザインスタイルシート<imarttag:imartDesignCss> を記述してください。

## 3.4.2 タイトルバー

### 3.4.2.1 機能詳細

- `<imarttag:imartTitleBar>` により、タイトルバーを表示する。
- タイトルバーには、「アイコン」と「ページタイトル名」を指定して表示することができる。

### 3.4.2.2 制約

- 本タグを配置する場所は、HTML の`<BODY>` タグ内です。

### 3.4.2.3 属性

`<imarttag:imartTitleBar>` に指定する属性を、以下に示します。

- 属性 `title` には「ページタイトル名」を指定する。
- 属性 `icon` には「アイコン」のファイル名を、JSPからの相対パス、またはコンテキストルートからの相対パスで指定する。

属性 `icon` を未指定にした場合は、アイコン  (`/images/standard/title.gif`) が表示される。

※ 属性 `icon` に指定する相対パスの記述例

JSPからの相対パス : `images/standard/title.gif`

コンテキストルートからの相対パス : `/images/standard/title.gif`

属性	型	必須	デフォルト値	説明
<code>title</code>	<code>java.lang.String</code>	○	—	ページタイトル名
<code>icon</code>	<code>java.lang.String</code>		—	アイコンのファイル名 ※JSPからの相対パス または コンテキストルートからの相対パス

### 3.4.2.4 使用例

タイトルバーを表示するためのサンプルを、以下に示します。

#### JSP のサンプル

```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
:
:
<HTML>
<HEAD>
<imarttag:imartDesignCss />
</HEAD>
<BODY>
<!-- タイトルバー表示 -->
<imarttag:imartTitleBar title="新規登録" icon="/images/standard/title.gif" />
</BODY>
</HTML>
```

#### 表示結果



新規登録

### 3.4.3 ツールバー

#### 3.4.3.1 機能詳細

- `<imarttag:imartToolbarFrame>` タグにより、ツールバーを表示する。  
(タグの詳細は「3.4.3.5.1 `<imarttag:imartToolbarFrame>`」を参照)
- ツールバーの中には、複数の「処理リンク」を配置することができる。
- `<imarttag:imartToolbarLeft>` タグに挟まれた「処理リンク」は、ツールバーの左側に配置される。  
(タグの詳細は「3.4.3.5.2 `<imarttag:imartToolbarLeft>`」を参照)
- `<imarttag:imartToolbarRight>` タグに挟まれた「処理リンク」は、ツールバーの右側に配置される。  
(タグの詳細は「3.4.3.5.3 `<IMART type="imToolbarRight">`」を参照)
- `<imarttag:imartToolbarFrame>` タグの内部に、`<imarttag:imartToolbarRight>` タグおよび `<imarttag:imartToolbarLeft>` タグがどちらも存在しない場合の動作については未定義とする。
- `<imarttag:imartIcon>` タグにより、「アイコン＋処理名」の処理リンクを表示する。  
(タグの詳細は「3.4.3.5.4 `<imarttag:imartIcon>`」を参照)

#### 3.4.3.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。

#### 3.4.3.3 手順

ここでは、ツールバーを作成するための手順として、JSP ファイルに記述する内容を簡単に説明します。

※ ソースコードのサンプルについては、「3.4.3.4 使用例」を参照してください。

- (1) `<imarttag:imartDesignCss>` を記述して、スタイルシートの宣言を行います。  
(記述方法については「3.4.1 デザインスタイルシート」を参照)
- (2) `<imarttag:imartToolbarFrame>` タグを記述して、ツールバーの外枠を表示します。
- (3) `<imarttag:imartToolbarFrame>` タグに挟まれた範囲に、`<imarttag:imartToolbarLeft>` タグを記述します。  
(ツールバーの左側に配置する「処理リンク」がない場合は、省略可能。)
- (4) `<imarttag:imartToolbarFrame>` タグに挟まれた範囲に、`<imarttag:imartToolbarRight>` タグを記述します。  
(ツールバーの右側に配置する「処理リンク」がない場合は、省略可能。)
- (5) `<imarttag:imartIcon>` タグを、`<imarttag:imartToolbarLeft>` タグ または、`<imarttag:imartToolbarRight>` タグに挟まれた範囲に記述して、「アイコン＋処理名」の処理リンクを表示します。
- (6) 「処理リンク」をツールバーの左側に表示するときは、`<imarttag:imartToolbarLeft>` タグに挟まれた範囲に `<imarttag:imartIcon>` タグを記述してください。
- (7) 「処理リンク」をツールバーの右側に表示するときは、`<imarttag:imartToolbarRight>` タグに挟まれた範囲に `<imarttag:imartIcon>` タグを記述してください。

## 3.4.3.4 使用例

ツールバーを表示するためのサンプルを、以下に示す。

## JSP のサンプル

```

<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
<%
:
:
%>
<HTML>
  <HEAD>
    <imarttag:imartDesignCss />
  </HEAD>
  <BODY>
    <!-- タイトルバー表示 -->
    :
    (省略)
    :
    <!-- ツールバー表示 -->
    <imarttag:imartToolbarFrame>
      <imarttag:imartToolbarLeft>
        <imarttag:imartIcon
          name = "新規登録"
          icon = "/images/standard/new.gif"
          href = "javascript:onAddAction();" />
        <imarttag:imartIcon
          name = "検索条件"
          icon = "/images/standard/search.gif"
          href = "javascript:onSearchAction();" />
      </imarttag:imartToolbarLeft>
      <imarttag:imartToolbarRight>
        <imarttag:imartIcon
          name = "戻る"
          icon = "/images/standard/arrow_left.gif"
          href = "javascript:onBackAction();" />
        <imarttag:imartIcon
          name = "最新情報"
          icon = "/images/standard/refresh.gif"
          href = "javascript:onReloadAction();" />
      </imarttag:imartToolbarRight>
    </imarttag:imartToolbarFrame>
  </BODY>
</HTML>

```

手順(1)

手順(5) (6)

手順(5) (6)

手順(5) (7)

手順(5) (7)

手順(3)

手順(2)

手順(4)

## 表示結果



### 3.4.3.5 ツールバーを作成するために必要なタグライブラリ

この章では、ツールバーを作成するために必要な、タグライブラリの詳細を示します。

ツールバーは、以下の「表 3-6 ツールバーに必要な画面デザイン共通モジュール」に示す、タグライブラリで構成されています。

表 3-6 ツールバーに必要な画面デザイン共通モジュール

参照		ページ
3.4.3.5.1	<imarttag:imartToolbarFrame>	66
3.4.3.5.2	<imarttag:imartToolbarLeft>	67
3.4.3.5.3	<imarttag:imartToolbarRight>	67
3.4.3.5.4	<imarttag:imartIcon>	68

#### 3.4.3.5.1 <imarttag:imartToolbarFrame>

##### 3.4.3.5.1.1 詳細

- ツールバーの外枠を表示する。
- ツールバーの表示色は、テーマによって自動に設定される。

##### 3.4.3.5.1.2 属性

- <imarttag:ImartToolbarFrame> に指定する属性は、ありません。



### 3.4.3.5.2 <imarttag:imartToolbarLeft>

#### 3.4.3.5.2.1 詳細

- ツールバー内の 左側 に「処理リンク」を表示させたいときに使用する。  
ツールバーの左側に表示する「処理リンク」がないときは、省略可能である。
- 必ず <imarttag:imartToolbarFrame> タグの内部に記述すること。
- <imarttag:imartToolbarLeft> タグに挟まれた範囲に記述された「処理リンク」は、ツールバー内で左寄せに表示される。
- 本タグに挟まれた範囲には、複数の「処理リンク」を記述することができる。
- 「処理リンク」には、<imarttag:imartIcon> タグ（詳細は「3.4.3.5.4 <imarttag:imartIcon>」を参照）だけでなく、通常のHTMLタグ<A href="XXXX"></A> なども記述することが可能である。

#### 3.4.3.5.2.2 制約

- 本タグを単独で利用することはできない。  
<imarttag:imartToolbarFrame> タグの内部タグとして利用すること。
- 本タグは<imarttag:imartToolbarRight> タグの前に記述すること。

#### 3.4.3.5.2.3 属性

<imarttag:imartToolbarLeft> タグに指定する属性は、ありません。

### 3.4.3.5.3 <imarttag:imartToolbarRight>

#### 3.4.3.5.3.1 詳細

- ツールバー内で「処理リンク」を 右側 に表示させたいときに使用するタグである。  
ツールバーの右側に表示する「処理リンク」がないときは、省略可能である。
- 必ず <imarttag:imartToolbarFrame> タグの内部に記述すること。
- <imarttag:imartToolbarRight> タグに挟まれた範囲に記述された「処理リンク」は、ツールバー内で右寄せに表示される。
- 本タグに挟まれた範囲には、複数の「処理リンク」を記述することができる。
- 「処理リンク」には、<imarttag:imartIcon> タグ（詳細は「3.4.3.5.4 <imarttag:imartIcon>」を参照）だけでなく、通常のHTMLタグ<A href="XXXX"></A> なども記述することが可能である。

#### 3.4.3.5.3.2 制約

- 本タグを単独で利用することはできない。  
<imarttag:imartToolbarFrame> タグの内部タグとして利用すること。
- 本タグは<imarttag:imartToolbarLeft> タグの後に記述すること。

#### 3.4.3.5.3.3 属性

<imarttag:imartToolbarRight> タグに指定する属性は、ありません。

### 3.4.3.5.4 <imarttag:imartIcon>

#### 3.4.3.5.4.1 詳細

ツールバー内に「処理リンク」を表示する。

- この <imarttag:imartIcon> タグにより、「アイコン＋処理名」の処理リンクが表示される。
- 「アイコン」のみ、または「処理名」のみの表示も可能である。
- 表示するアイコンは、属性 icon により指定できる。
- 表示する処理名は、属性 name により指定できる。
- 処理を選択されたときのリンク先として、js の関数名や URL を指定することができる。  
また、リンク先の表示ウィンドウを指定することができる。
- 本タグを <imarttag:imartToolBarLeft> タグに挟まれた範囲に記述すると、ツールバー内で左寄せに表示される。
- 本タグを <imarttag:imartToolBarRight> タグに挟まれた範囲に記述すると、ツールバー内で右寄せに表示される。
- 本タグを記述した順番に、ツールバーの左から「処理リンク」が配置される。

#### 3.4.3.5.4.2 制約

- 属性 href を指定した場合、<A>タグとして作成されます。
- 属性 attr (任意の属性)を指定する場合、属性 href が設定されている必要があります。

#### 3.4.3.5.4.3 属性

<imarttag:imartIcon> に指定する属性を、以下に示す。

- 属性 name には「処理名」を指定する。  
属性 name を未指定にした場合は、「アイコン」だけが表示される。
- 属性 icon には「アイコン」のファイル名を、JSPからの相対パス、またはコンテキストルートからの相対パスで指定する。  
属性 icon を未指定にした場合は、「処理名」だけが表示される。  
※ 属性 icon に指定する相対パスの記述例  
JSP からの相対パス                   : images/standard/search.gif  
コンテキストルートからの相対パス   : /images/standard/search.gif
- 属性 name、および 属性 icon がどちらも未指定の場合は、処理リンクの配置を無視する。
- 属性 href には処理を選択された際のリンク先を指定する。  
指定する値は、js の関数名 (javascript:XXXXX();) や、URL などが可能である。  
属性 href を未指定にした場合は、リンク表示ではなく、ただのテキスト表示になる。  
js の関数名       : href="javascript:onAddAction()"  
URL               : href="http://www.intra-mart.co.jp"
- 属性 target にはリンク先の表示ウィンドウを指定する。  
指定する値は、HTML の target 属性に従う。  
属性 target を未指定にした場合は、同ウィンドウに表示される。
- 属性 alt には画像の変わりに表示される文字列を指定する。
- 属性 attr には<A>タグに指定する、「その他の属性値」を指定する。  
属性 attr に指定する値は、一度「変数」に代入してから attr="<%= attr %>"で指定すること。  
属性 attr に指定した値は、指定された文字列のまま<A>タグの属性として出力される。

(例)

```

<%
    String attr="tabindex =¥"5¥" id=¥"reloadInfo¥"";
%>
:
<imarttag:imartIcon name="最新情報" attr="<%= attr %>" />

```

属性	型	必須	デフォルト値	説明
name	java.lang.String	○	—	処理名
icon	java.lang.String		—	アイコンのファイル名 ※JSP からの相対パス または コンテキストルートからの相対パス
href	java.lang.String		—	処理選択時のジャンプ先 ※js 実行関数名や、URL など
Target	java.lang.String			表示ウィンドウ
alt	java.lang.String		—	画像の変わりに表示される文字列を指定
attr	java.lang.String		—	<A>タグに指定する属性値

### 3.4.4 入力項目

ここでは、登録系画面や、更新・削除系画面などで頻繁に使用される「項目名称＋入力フィールド」をまとめて、「入力項目」として説明しています。

#### 3.4.4.1 機能詳細

- 登録系画面や、更新・削除系画面で、入力項目を作成する際に使用するタグである。
- `<imarttag:imartItemNameTd>` により、項目名称を表示する。  
(タグの詳細は「3.4.4.5.1 項目名称」を参照)
- `<imarttag:imartInputTd>` により、入力フィールドを表示する。  
(タグの詳細は「3.4.4.5.2 入力フィールド」を参照)
- `<imarttag:imartSelectTd>` により、選択部品(セレクトボックス)を表示する。  
(タグの詳細は「3.4.4.5.3 選択ボックス」を参照)

#### 3.4.4.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。

HTML の`<TD>`タグとして作成されるため、`<TABLE><TR>~</TR></TABLE>` 内に記述してください。

また、デザインを統一するために本タグを使用する際には、必ず「3.4.4.4 使用例」に示す

`<TABLE><TR>~</TR></TABLE>`の中に記述してください。

#### 3.4.4.3 手順

ここでは、入力項目を作成するための手順として、JSP ファイルに記述する内容を簡単に説明します。

※ ソースコードのサンプルについては、「3.4.4.4 使用例」を参照してください。

- (1) `<imarttag:imartDesignCss>` を記述して、スタイルシートの宣言を行います。  
(記述方法については「3.4.1 デザインスタイルシート」を参照)
- (2) デザインを統一するため、`<TABLE><TR>~</TR></TABLE>` を記述します。
- (3) `<imarttag:imartItemNameTd>` タグを (2) に挟まれた範囲に記述して、項目名称を表示します。
- (4) `<imarttag:imartInputTd>` タグを (2) に挟まれた範囲に記述して、入力フィールドを表示します。
- (5) 選択ボックスに設定する、表示データを構築します。
- (6) `<imarttag:imartSelectTd>` タグを (2) に挟まれた範囲に記述して、選択ボックスを表示します。

## 3.4.4.4 使用例

入力項目を表示するためのサンプルを、以下に示します。

JSP のサンプル

```

<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
<%
    Vector from = new Vector();
    HashMap hashTemp = new HashMap();

    // <imartSelectTd> の list 属性に指定するデータを生成
    String[] fromList = new String[]{"東京都", "神奈川県", "埼玉県", "千葉県"};
    String[] fromValue = new String[]{"Tokyo", "Kanagawa", "Saitama", "Chiba"};

    for(int i = 0 ; i < fromList.length ; i++)
    {
        hashTemp = new HashMap();
        hashTemp.put("value", fromValue[i]);
        hashTemp.put("from", fromList[i]);
        from.add(hashTemp);
    }
    :
    :
%>
<HTML>
<HEAD>
<imarttag:imartDesignCss />
</HEAD>
<BODY>
<!-- 入力項目表示 -->
<TABLE width="100%" border="0" cellpadding="0" cellspacing="0">
<TR>
<TD align="center">
<TABLE class="edit">
<TR>
<imarttag:imartItemNameTd name="ユーザ ID" />
<imarttag:imartInputTd
type = "text"
name = "userId"
size = "30"
value = "User00011"
readonly. />
</TR>
<TR>
<imarttag:imartItemNameTd name="ユーザ名" require />
<imarttag:imartInputTd
type = "text"
name = "userName"
size = "50"
value = "ユーザ 00011" />
</TR>
<TR>
<imarttag:imartItemNameTd name="パスワード" require />
<imarttag:imartInputTd
type = "password"
name = "passWord"
size = "40" />
</TR>
<TR>
<imarttag:imartItemNameTd name="出身地" />
<imarttag:imartSelectTd
list = "<%= from %>"
name = "hometown"
optionValue = "value"

```

手順(5)

手順(1)

手順(3)

手順(4)

手順(3)

手順(4)

手順(2)

手順(3)

手順(4)

手順(3)

手順(6)

```
optionText = "from"
size = "1" />
</TR>
<TR>
<imarttag:imartItemNameTd name="転送ファイル" />
<imarttag:imartInputTd
type = "file"
name = "forwarFile"
size = "50" />
</TR>
</TABLE>
</TD>
</TR>
</TABLE>
:
:
</BODY>
</HTML>
```

手順(3)

手順(4)

手順(2)

## 表示結果

ユーザID	<input type="text" value="User00011"/>
ユーザ名(必須)	<input type="text"/>
パスワード(必須)	<input type="password"/>
出身地	<input type="text" value="東京都"/> ▼
転送ファイル	<input type="text"/> 参照...

### 3.4.4.5 入力項目を作成するために使用するタグライブラリ

#### 3.4.4.5.1 項目名称

##### 3.4.4.5.1.1 詳細

- `<imarttag:imartItemNameTd>` により、入力項目の「項目名称」を表示する。
- 登録系画面や、更新・削除系画面で、入力項目の項目名称を記述する際に使用する。
- 属性 `name` に指定した値が、項目名称として表示される。
- 属性 `attr` に指定した値は、`<TD>`タグの属性として出力される。
- 属性 `require` に指定した値によって、「必須項目」または「通常項目」を切り替えることができる。

##### 3.4.4.5.1.2 制約

HTML の`<TD>`タグとして作成されるため、`<TABLE><TR>~</TR></TABLE>` 内に記述してください。

また、デザインを統一するために本タグを使用する際には、必ず「3.4.4.4 使用例」に示す

`<TABLE><TR>~</TR></TABLE>`の中に記述してください。

##### 3.4.4.5.1.3 属性

`<imarttag:imartItemNameTd>` に指定する属性を、以下に示します。

- 属性 `name` には「項目名称」を指定する。
- 属性 `attr` には `<TD>`タグに指定する、「その他の属性値」を指定する。  
属性 `attr` に指定する値は、一度「変数」に代入してから、`attr="<%= attr %>"`で指定すること。  
属性 `attr` に指定した値は、指定された文字列のまま `<TD>`タグの属性として出力される。  
`<TD>`タグ属性を複数指定する場合は、`[属性名]=[値]` をスペース区切りで繋げた文字列とする。  
指定する値は、HTML の `<TD>`タグ属性に従う。

(例)

```
<%
String attr="width=¥150¥ height=¥30¥ colspan=¥2¥";
%>
:
<imarttag:imartItemNameTd name="ユーザ ID" attr="<%= attr %>" />
```

- 属性 `require` には、「必須項目」または「通常項目」の種別を指定する。  
属性 `require` に `"true"` を指定すると、「必須項目」として表示される。  
属性 `require` を未指定または、`"false"` を指定した場合は、「通常項目」として表示される。

属性	型	必須	デフォルト値	説明
name	java.lang.String	○	—	項目名称
attr	java.lang.String		—	<code>&lt;TD&gt;</code> タグに指定する属性値
require	boolean		—	必須項目の指定

<imarttag: imartItemNameTd>の記述例を示します。

〔記述例 ①〕：属性 `require` を指定して、必須項目にした場合

```
<IMART type="imItemName" name="ユーザ名" require="true"></IMART>
```

ユーザ名(必須)

〔記述例 ②〕：属性 `require` を未指定にして、通常項目にした場合

```
<IMART type="imItemName" name="ユーザ名"></IMART>
```

ユーザ名

〔記述例 ③〕：属性 `attr` を未指定にして、<TD>タグに属性を追加した場合

```
<%  
  String attr="width=¥"150¥" height=¥"100¥"";  
%>  
:  
<imarttag:imartItemNameTd name="ユーザ ID" attr="<%= attr %>" />
```

ユーザID



#### 3.4.4.5.2.1 詳細

#### 3.4.4.5.2.1 詳細

- |       |           |
|-------|-----------|
| ユーザID | User00011 |
|-------|-----------|

- パスワード

- 転送ファイル  [参照...](#)

- 希望日 ☐ 6月1日 ☐ 6月2日 ☐ 6月3日

- 性別 ☐ 男性 ☐ 女性

- |      |             |
|------|-------------|
| コメント | <div></div> |
|------|-------------|

また、デザインを統一するために本タグを使用する際には、必ず「3.4.4.4 使用例」に示す  
 <TABLE><TR>～</TR></TABLE>の中に記述してください。

## 3.4.4.5.2.3 属性

<imarttag:imartInputTd> に指定する属性を、以下に示します。

## 【共通属性】

- 属性 type には、表示する部品の「種類」を指定する。

属性 type に指定できる値は、以下の通りである。

text : テキスト入力フィールド  
password : パスワード入力フィールド  
file : ファイル入力フィールド  
checkbox : チェックボックス  
radio : ラジオボタン  
textarea : テキストエリア入力フィールド

(属性 type="textarea" 以外は、HTML の<INPUT>タグの type 属性同様)

- 属性 name には「名前」を指定する。

(HTML の<INPUT> および <TEXTAREA>タグの name 属性同様)

- 属性 attr には <TD>タグに指定する、「その他の属性値」を指定する。

属性 attr に指定する値は、一度「変数」に代入してから、attr="<%= attr %>"で指定すること。

属性 attr に指定した値は、指定された文字列のまま <TD>タグの属性として出力される。

<TD>タグ属性を複数指定する場合は、[属性名]=[値]" をスペース区切りで繋げた文字列とする。

指定する値は、HTML の <TD>タグ属性に従う。

(例)

```
<%
String attr="width=¥150¥ height=¥30¥ colspan=¥2¥";
%>
:
<imarttag:imartInputTd name="userId" attr="<%= attr %>" />
```

- 属性 inputAttr には <INPUT>タグに指定する、「その他の属性値」を指定する。

属性 inputAttr に指定する値は、一度「変数」に代入してから、inputAttr="<%= inputAttr %>"で指定すること。

属性 inputAttr に指定した値は、指定された文字列のまま <INPUT>タグの属性として出力される。

複数指定する場合は、[属性名]=[値]" をスペース区切りで繋げた文字列とする。

指定する値は、HTML の <INPUT>タグ属性に従う。

(例)

```
<%
String inputAttr = "maxlength=¥256¥";
%>
:
<imarttag:imartInputTd name="userId" inputAttr="<%= inputAttr %>" />
```

属性	型	必須	デフォルト値	説明
type	java.lang.String	○	—	部品の種類 ・text: テキスト入力フィールド ・password: パスワード入力フィールド ・file: ファイル入力フィールド ・checkbox: チェックボックス ・radio: ラジオボタン ・textarea: テキストエリア入力フィールド
name	java.lang.String		—	名前
attr	java.lang.String		—	<TD>タグに指定する属性値
inputAttr	java.lang.String		—	<INPUT>タグに指定する属性値

## 【type="text" の属性】・・・テキスト入力フィールド

属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)

属性 value には、入力フィールドの「初期値」を指定する。(HTML の<INPUT>タグの value 属性同様)

属性 readonly には、入力の可否を指定する。

属性 readonly に "true" を指定した場合は、入力フィールドへの入力が不可になる。


(HTML の<INPUT>タグの readonly 属性同様)

属性 readonly を未指定または、に "false" をした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="50"
value	初期値		—	value="User00011"
readonly	入力不可の指定		—	readonly="true"

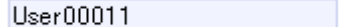
【記述例 ①】：属性 readonly を未指定して、入力可能にした場合

```
<imarttag:imartInputTd
type="text" name="userID" size="30" value="User00011" />
```



【記述例 ②】：属性 readonly を指定して、入力不可にした場合

```
<imarttag:imartInputTd type="text"
name="userID" size="30" value="User00011" readonly="true" />
```



## 【type="password" の属性】・・・パスワード入力フィールド

■ 属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)

■ 属性 value には、入力フィールドの「初期値」を指定する。(HTML の<INPUT>タグの value 属性同様)

■ 属性 readonly には、入力の可否を指定する。

属性 readonly に "true" を指定した場合は、入力フィールドへの入力が不可になる。

(HTML の<INPUT>タグの readonly 属性同様)

属性 readonly を未指定または、に "false" をした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="40"
value	初期値		—	value="Password00001"
readonly	入力不可の指定		—	readonly="true"

【記述例 ①】：属性 readonly を未指定して、入力可能にした場合

```
<imarttag:imartInputTd
type="password" name="pass" size="40" value="Password00011" />
```



【記述例 ②】：属性 readonly を指定して、入力不可にした場合

```
<imarttag:imartInputTd type="password"
name="pass" size="40" value="Password00011" readonly="true" />
```



【type="file" の属性】・・・ファイル入力フィールド

- 属性 size には、入力フィールドの「大きさ」を指定する。(HTML の<INPUT>タグの size 属性同様)

属性	説明	必須	デフォルト値	書式
size	大きさ		—	size="50"

〔記述例〕：属性 size="50" を指定して表示

```
<imarttag:imartInputTd type="file" name="forward_file" size="50" />
```



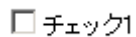
【type="checkbox" の属性】・・・チェックボックス

- 属性 caption には、チェックボックス横に表示する「名称」を設定する。
- 属性 checked を指定した場合は、チェックボックスがチェックされた状態で表示される。  
(HTML の<INPUT>タグの checked 属性同様)

属性	説明	必須	デフォルト値	書式
caption	名称		—	caption="チェック 1"
checked	チェック状態の指定		—	checked

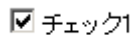
〔記述例 ①〕：属性 caption="チェック 1" と指定して表示

```
<imarttag:imartInputTd type="checkbox" name="check1" caption="チェック 1" />
```



〔記述例 ②〕：属性 checked を指定して、あらかじめチェックされた状態にした場合

```
<imarttag:imartInputTd type="checkbox" name="check1" caption="チェック 1" checked />
```



【type="radio" の属性】・・・ラジオボタン

- 属性 caption には、ラジオボタン横に表示する「名称」を設定する。
- 属性 checked を指定した場合は、ラジオボタンがチェックされた状態で表示される。  
(HTML の<INPUT>タグの checked 属性同様)

属性	説明	必須	デフォルト値	書式
caption	名称		—	caption="ラジオ 1"
checked	チェック状態の指定		—	checked

【記述例 ①】：属性 caption="ラジオ 1" と指定して表示

```
<imarttag:imartInputTd type="radio"
name="radio1" caption="ラジオ 1" />
```

☐ ラジオ1

【記述例 ②】：属性 checked を指定して、あらかじめチェックされた状態にした場合

```
<imarttag:imartInputTd type="radio"
name="radio1" caption="ラジオ 1" checked />
```

☒ ラジオ1

【type="textarea" の属性】・・・テキストエリア表示

- 属性 value には、表示する「文字列」を指定する。
  - 属性 cols には、テキストエリアの「横幅」を指定する。(HTML の<TEXTAREA>タグの cols 属性同様)
  - 属性 rows には、テキストエリアの「縦幅(行数)」を指定する。  
(HTML の<TEXTAREA>タグの rows 属性同様)
  - 属性 readonly には、入力の可否を指定する。  
属性 readonly に "true" を指定した場合は、入力フィールドへの入力が不可になる。  
(HTML の<TEXTAREA>タグの readonly 属性同様)
- 属性 readonly を未指定または、に "false" をした場合は、デフォルト値「入力可能」が設定される。

属性	説明	必須	デフォルト値	書式
value	表示する文字列		—	value="ここに長い・・・"
cols	テキストエリアの横幅 (<TEXTAREA>の cols 属性)		—	cols="40"
rows	テキストエリアの縦幅(行数) (<TEXTAREA>の rows 属性)		—	rows="5"
readonly	入力不可の指定		—	readonly="true"

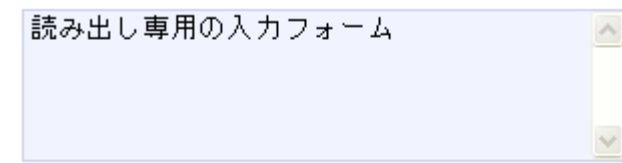
【記述例】：属性 cols="40"、属性 rows="5" を指定して表示

```
<imarttag:imartInputTd type="textarea" name="comment"
cols="40" rows="5" value="ここに長い文章が書けます。" />
```

ここに長い文章が書けます。

〔記述例〕：属性 cols="40"、属性 rows="5" を指定して表示

```
< imarttag:imartInputTd type="textarea" name="comment"
cols="40" rows="5" value="読み出し専用の入力フォーム" readonly />
```



### 3.4.4.5.3 選択ボックス

#### 3.4.4.5.3.1 詳細

- `<imarttag:imartSelectTd>` により、入力項目の選択ボックスを表示する。
- 登録系画面や、更新・削除系画面で、入力項目の選択ボックスを記述する際に使用する。
- 既存の画面デザイン共通モジュール `<imarttag:Select>` と同じ機能であり、画面デザインを統一するためにデザインが変更されている。

#### 3.4.4.5.3.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。  
HTML の`<TD>`タグとして作成されるため、`<TABLE><TR>~</TR></TABLE>` 内に記述してください。  
また、デザインを統一するために本タグを使用する際には、必ず「3.4.4.4 使用例」に示す  
`<TABLE><TR>~</TR></TABLE>`の中に記述してください。

#### 3.4.4.5.3.3 属性

`<imarttag:imartSelectTd>` に指定する属性を、以下に示します。

- 属性 `list` には「表示リスト」を指定する。  
属性 `list` には、List インターフェースを実装したクラスのインスタンスに Map インターフェースを実装したクラスのインスタンスを格納して指定する。  
属性 `list` に指定したインスタンスに格納されているすべての Map インターフェースを実装したクラスのインスタンスの値が、コンボボックス内の表示要素になる。
- 属性 `list` に指定されたインスタンスに格納されている Map インターフェースを実装したクラスの `optionValue` 属性で指定されたキー名に格納されている値が、コンボボックス内の各要素(`<OPTION>`タグ)の `value` 値になり、`optionText` 属性で指定されたキー名に格納されている値がコンボボックス内の要素の画面上での表示名になる。  
(ただし、Map インターフェースを実装したクラスのインスタンスに格納されているキーおよび値が String 以外の場合の動作は保証外である。)
- 属性 `optionValue` を未指定にした場合は、デフォルト値の「`option_value`」が設定される。
- 属性 `optionText` を未指定にした場合は、デフォルト値の「`option_text`」が設定される。
- 属性 `selected` には、コンボボックスの初期選択 (初期表示) 値を指定する。  
初期選択状態にする値のキー (属性 `list` に指定した Map インターフェースを実装したクラスのインスタンスの該当するキー名称) と一致する文字列を含む List インターフェースを実装したクラスのインスタンスを指定することで、初期選択状態にすることが出来る。  
属性 `selected` に指定した値に該当するキーが、属性 `list` に指定した Map インターフェースを実装したクラスのインスタンスのキー名称として存在しない場合は、どの値も選択状態にはならない。  
(ただし、コンボボックスの初期選択指定が存在しない場合は、ブラウザの仕様で自動的にコンボボックス内の先頭の要素が初期選択状態として表示される。)  
また、初期選択状態とするキーを同時に複数個指定する場合は、属性 `selected` に対して複数のキー文字列を含む List インターフェースを実装したクラスのインスタンスを指定する事で可能になる。  
その際、配列内の各要素の値として、初期選択状態にするキー (属性 `list` に格納した Map インターフェースを実装したクラスのインスタンスのキー名称) を格納すること。
- 属性 `blank` には、コンボボックス内に空データを表示させる「位置」を指定する。  
属性 `blank` に`"top"`を指定した場合は、ボックス内の先頭位置に空データが表示される。  
属性 `blank` に`"bottom"`を指定した場合は、ボックス内の最後尾位置に空データが表示される。

- 属性 `size` には、一度に表示する選択肢の行数を指定する。(HTML の<SELECT>タグの `size` 属性同様)  
属性 `size` を未指定にした場合は、デフォルト値「1」が設定される。
- 属性 `multiple` を指定した場合は、複数行選択が可能になる。(HTML の<SELECT>タグの `multiple` 属性が付加される)  
属性 `multiple` を未指定にした場合は、複数行選択が不可になる。
- 属性 `attr` には <TD>タグに指定する、「その他の属性値」を指定する。  
属性 `attr` に指定する値は、一度「変数」に代入してから、`attr="<%= attr %>"`で指定すること。  
属性 `attr` に指定した値は、指定された文字列のまま <TD>タグの属性として出力される。  
<TD>タグ属性を複数指定する場合は、[属性名]=[値] をスペース区切りで繋げた文字列とする。  
指定する値は、HTML の <TD>タグ属性に従う。

(例)

```
<%
String attr="width=¥150¥ height=¥30¥ colspan=¥2¥";
%>
:
<imarttag:imartSelectTd name="sel" attr="<%= attr %>" />
```

- 属性 `selectAttr` には <INPUT>タグに指定する、「その他の属性値」を指定する。  
属性 `selectAttr` に指定する値は、一度「変数」に代入してから、`selectAttr="<%= selectAttr %>"`で指定すること。  
属性 `selectAttr` に指定した値は、指定された文字列のまま <SELECT>タグの属性として出力される。  
複数指定する場合は、[属性名]=[値] をスペース区切りで繋げた文字列とする。  
指定する値は、HTML の <SELECT>タグ属性に従う。

(例)

```
<%
String selectAttr="title=¥ユーザID¥ disabled";
%>
:
<imarttag:imartSelectTd name="sel" selectAttr="<%= selectAttr %>" />
```

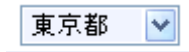
属性	型	必須	デフォルト値	説明
list	java.util.List	○	—	表示リスト
selected	java.util.List		—	初期選択値の設定
blank	java.util.String		—	空データを表示する位置 ・top : 先頭位置に空データ ・bottom: 最後尾位置に空データ
name	java.util.String			名前
size	java.util.String または double		1	表示する行数
multiple				複数行選択の可否
optionValue	java.util.String		option_value	<OPTION> タグの value 値に対応する キー名
optionText	java.util.String		option_text	<OPTION> タグの text 値に対応する キー名 (任意)
attr	java.lang.String		—	<TD>タグに指定する属性値
selectAttr	java.lang.String		—	<SELECT>タグに指定する属性値



〔記述例 ①〕：属性 list に指定した値を、選択ボックスで表示

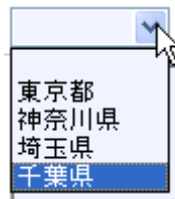
※ <%= from %> の設定方法については、「3.4.4.4 使用例」を参照

```
<imarttag:imartSelectTd list="<%= from %>"
name="hometown" optionValue="value" optionText="from" />
```



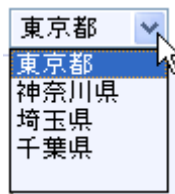
〔記述例 ②〕：属性 blank="top" に指定した場合 → 先頭に空データが挿入

```
<imarttag:imartSelectTd list="<%= from %>"
name="hometown" optionValue="value" optionText="from" blank="top" />
```



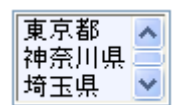
〔記述例 ③〕：属性 blank="bottom" に指定した場合 → 最後尾に空データが挿入

```
<imarttag:imartSelectTd list="<%= from %>"
name="hometown" optionValue="value" optionText="from" blank="bottom" />
```



〔記述例 ④〕：属性 size="3" に指定した場合

```
<imarttag:imartSelectTd list="<%= from %>"
name="hometown" optionValue="value" optionText="from" size="3" />
```



〔記述例 ⑤〕：属性 selectAttr="disabled" を指定し、<SELECT>タグに属性を設定

```
<%
String selectAttr ="disabled";
%>
:
<imarttag:imartSelectTd list="<%= from %>" name="hometown"
optionValue="value" optionText="from" selectAttr="<%= selectAttr %>" />
```











### 3.4.5 リストコントロール

#### 3.4.5.1 機能詳細

- <imarttag:imartListControl> タグにより、各種一覧に対する ソート切り替えコントロールと、ページ切り替えコントロールを表示する。
- ソート および、ページ切り替えコントロールはリンク表示となっている。  
各コントロールのリンクがクリックされると、JavaScript関数 が呼び出されるため、本タグを使用した場合は、各コントロールに対してJavaScript関数の処理を記述する必要がある。  
(詳細は「3.4.5.5 デフォルトの必須JavaScript関数」を参照)
- ソート切り替えコントロールには、昇順ソート(有効および無効)、降順ソート(有効および無効)の矢印リンクが用意されている。

ソート対象	昇順アイコン	降順アイコン
昇順	 (有効)	 (無効)
降順	 (無効)	 (有効)

- ソート切り替えコントロールは、属性の指定により表示、非表示を設定することができる。
- ページ切り替えコントロールには、 (前のページ)、 (次のページ)、 (最初のページ)、 (最後のページ) の矢印リンクが用意されている。
- 一覧で表示しているページによって、矢印アイコンの表示および、非表示を切り替える。

表示ページ	最初のページ アイコン	前のページ アイコン	次のページ アイコン	最後のページ アイコン
最初のページ	非表示	非表示		
最後のページ			非表示	非表示

- 一覧の表示件数が [全件数<1ページの表示数] の場合は、ページ切り替えコントロールの矢印部分は表示されない。(ページ件数の表示のみ)

#### 3.4.5.2 制約

本タグは、HTML の<TABLE>タグとして作成されます。  
本タグを配置する場所は、HTML の<BODY> タグ内です。

## 3.4.5.3 属性

<imarttag:imartListControl> に指定する属性を、以下に示す。

- 属性 maxRecord には一覧に表示するデータの「全件数」を指定する。
- 属性 currentPage には「表示するページ番号」を指定する。  
属性 currentPage を未指定にした場合は、『1 ページ目』が表示される。
- 属性 pageLine には「1ページの表示数」を指定することができる。  
属性 pageLine を未指定にした場合は、1 ページに表示される件数は『10 件』となる。
- 「昇順／降順」切り替えコントロールは 表示、または 非表示を指定することができる。  
属性 sortDisplay に "true" を指定した場合は表示、"false" を指定した場合は非表示になる。  
属性 sortDisplay が未指定の場合は、「昇順／降順」切り替えコントロールが表示される。
- 属性 orderBy には「ソート基準方向」を指定する。  
現在のソート方向が昇順の場合は"asc"、降順の場合は"desc"を指定する。  
属性 orderBy を未指定にした場合は、デフォルト値の "asc"（昇順）となる。
- 属性 prevPageFunc には、前ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 currentPage]-1')" が設定される。
- 属性 nextPageFunc には、次ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 currentPage]+1')" が設定される。
- 属性 firstPageFunc には、最初ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('1')" が設定される。
- 属性 lastPageFunc には、最終ページリンク  が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onPageLinkFunc('[属性 maxRecord]/[属性 pageLine]')" が設定される。
- 属性 ascSortFunc には、昇順ソートリンク  (有効)および  (無効)が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onSortLinkFunc('asc')" が設定される。
- 属性 descSortFunc には、降順ソートリンク  (有効)および  (無効)が選択時の js 関数を指定する。  
未指定の場合は、"javascript:onSortLinkFunc('desc')" が設定される。

属性	型	必須	デフォルト値	説明
maxRecord	java.util.String または double	○	—	全件数
currentPage	java.util.String または double		1	表示するページ番号
pageLine	java.util.String または double		10	1 ページの表示件数
sortDisplay	boolean		true	「昇順／降順」の有無 (true: 表示 / false: 非表示)
sortOrder	java.lang.String		asc	ソート基準方向 (asc: 昇順 / desc: 降順)
prevPageFunc	java.lang.String		上記参照	前ページ選択時の js 関数
nextPageFunc	java.lang.String		上記参照	次ページ選択時の js 関数
firstPageFunc	java.lang.String		上記参照	最初ページ選択時の js 関数
lastPageFunc	java.lang.String		上記参照	最終ページ選択時の js 関数
ascSortFunc	java.lang.String		上記参照	昇順ソート選択時の js 関数
descSortFunc	java.lang.String		上記参照	降順ソート選択時の js 関数

### 3.4.5.4 使用例

リストコントロール(ソート切り替え、ページ切り替え)を表示するためのサンプルを、以下に示します。

#### JSP のサンプル




```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
:
:
<HTML>
<HEAD>
<imarttag:imartDesignCss />
<SCRIPT LANGUAGE="JavaScript">

// ページ切り替え<>がクリックされた時の処理
function onPageLinkFunc(num)
{
:
// ...ここには各自で処理を記述する...
:
}

// 昇順／降順(ソート条件)をクリックした時の処理
function onSortLinkFunc(mode)
{
:
// ...ここには各自で処理を記述する...
:
}
</SCRIPT>
</HEAD>
<BODY>
<TABLE border="0" width="100%" cellpadding="0" cellspacing="2">
<TR>
<TD>
<!-- リストコントロール -->
<imarttag:imartListControl
maxRecord="150"
currentPage="1"
pageLine="10"
sortOrder="asc" />
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

表示結果

 昇順  降順

  11-20/150 表示中  

#### 3.4.5.5 デフォルトの必須JavaScript関数

ソート切り替えコントロール、およびページ切り替えコントロールのリンクがクリックされた際に呼び出される、デフォルトの JavaScript 関数の詳細を以下に示します。

本タグを使用する際に、以下の属性を未指定にした場合は、各自で必ずデフォルトの JavaScript 関数の処理を記述してください。

- 属性 prevPageFunc
- 属性 nextPageFunc
- 属性 firstPageFunc
- 属性 lastPageFunc
- 属性 ascSortFunc
- 属性 descSortFunc

※ 必須の JavaScript 関数を記述しない場合の動作については、保証外とします。

※ コーディング方法などの詳細については、「3.4.5.4 使用例」を参照してください。

※ 上記の属性を指定した場合は、デフォルトの JavaScript 関数の処理を記述する必要はありません。

### 3.4.5.1 ソート切り替え

<imarttag:imartListControl>で、属性 ascSortFunc および、属性 descSortFunc のいずれかを未指定にした場合、ソート切り替えコントロールで「昇順」および「降順」のリンクがクリックされると、JavaScript 関数の onSortLinkFunc() が呼び出されます。

以下に、その詳細を示します。

#### ■ 関数名

onSortLinkFunc( mode )

#### ■ 概要

ソート切り替えコントロール(昇順or降順) がクリックされた際に、呼び出される関数です。

引数 mode の値により「昇順」、「降順」のどちらが選択されたかの情報を、取得することができます。

#### ■ 引数

変数	内容	値	条件
mode	ソート基準方向	asc	「昇順」リンクが選択された場合
		desc	「降順」リンクが選択された場合

### 3.4.5.2 ページ切り替え

<imarttag:imartListControl>で、属性 prevPageFunc、nextPageFunc、firstPageFunc、lastPageFunc のいずれかを未指定にした場合、ページ切り替えコントロールで、各矢印のリンクがクリックされた場合、JavaScript 関数の onPageLinkFunc() が呼び出されます。

以下に、その詳細を示します。

#### ■ 関数名




onPageLinkFunc( num )

#### ■ 概要

ページ切り替えコントロールの、各矢印リンクがクリックされた際に、呼び出される関数です。

引数 num の値により、次に表示するページ番号の情報を、取得することができます。

#### ■ 引数

変数	内容	値	条件
num	ページ番号	1	「最初ページ」リンク  が選択された場合
		([属性 maxRecord] / [属性 pageLine]) の小数点繰上げ	「最後ページ」リンク  が選択された場合
		[属性 currentPage] - 1	「前ページ」リンク  が選択された場合
		[属性 currentPage] + 1	「次ページ」リンク  が選択された場合

## 3.4.6 リストヘッダ

### 3.4.6.1 機能詳細

- `<imarttag:imartListHeader>` により、各種一覧(リスト)に対する ヘッダを表示する。
- 属性 `headerData` には、「一覧ヘッダ情報」を指定する。  
属性 `headerData` には、`List` インターフェースを実装したクラスのインスタンスに、一覧のヘッダ情報クラス (`ListHeaderObject`) のインスタンスを格納して指定する。  
属性 `headerData` に指定したインスタンスに格納されているすべての `ListHeaderObject` クラスのインスタンスの値が、ヘッダ項目の表示要素になる。  
(一覧のヘッダ情報クラス (`ListHeaderObject`) の詳細は「3.4.6.5 クラス `ListHeaderObject`」を参照)
- 属性 `headerData` に指定されたインスタンスに格納されている、`ListHeaderObject` クラスのインスタンスを生成するときに指定した「項目 ID」が、属性 `sortKey` の値と一致する場合に、該当の項目がソートキー(現在、ソートの基準になっている項目のこと)となる。
- 属性 `sortKey` には、ヘッダ項目の初期表示時にソートキーとする「項目 ID」を指定する。  
属性 `sortKey` に指定した値が、属性 `headerData` に指定した `ListHeaderObject` クラスのインスタンスを生成するときに指定した「項目 ID」として存在しない場合は、どの項目もソートキーにはならない。

### 3.4.6.2 制約

本タグを配置する場所は、HTML の`<BODY>` タグ内です。  
また、HTML の`<TR>`タグとして作成されるため、`<TABLE>~</TABLE>` 内に記述してください。  
その際、`<TABLE>`タグには `class="list_border_bg"` または `class="table_border_bg"` を指定してください。

表全体の枠 (横幅 100% 固定)	<code>&lt;TABLE class="list_border_bg"&gt;</code>
表全体の枠 (横幅任意指定)	<code>&lt;TABLE class="table_border_bg"&gt;</code>

### 3.4.6.3 属性

`<imarttag:imartListHeader>` に指定する属性を、以下に示します。

- 属性 `headerData` には、`List` インターフェースを実装したクラスのインスタンスに、`ListHeaderObject` クラスのインスタンスを格納して指定する。  
(`ListHeaderObject` クラス の詳細は「3.4.6.5 クラス `ListHeaderObject`」を参照)
- 属性 `sortKey` には、ソートキーとする項目 ID を指定する。

属性	型	必須	デフォルト値	説明
<code>headerData</code>	<code>java.util.List</code>	○	—	一覧ヘッダ情報
<code>sortKey</code>	<code>java.util.String</code>	○	—	ソートキーとする項目 ID

## 3.4.6.4 使用例

リストヘッダを表示するためのサンプルを、以下に示します。

## JSP のサンプル

```
<%@ page contentType="text/html; charset=Windows-31J" pageEncoding="Shift_JIS" %>
<%@ taglib prefix="imarttag" uri="http://www.intra-mart.co.jp/taglib/foundation/imarttag" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="jp.co.intra_mart.foundation.core.taglib.ListHeaderObject" %>

<%
    // ヘッダ項目情報を設定
    ListHeaderObject objData = null;
    ArrayList data = new ArrayList();

    objData = new ListHeaderObject("checkbox", "checkCtrl", "onCheckBox()"); ..... (1)
    data.add(objData);

    objData = new ListHeaderObject("userId", "ユーザコード"); ..... (2)
    data.add(objData);

    objData = new ListHeaderObject("userName", "ユーザ名"); ..... (3)
    data.add(objData);

    objData = new ListHeaderObject("comp", "会社／組織", false); ..... (4)
    objData.setAlign("left");
    objData.setAttr("width=¥"150¥" tabindex=¥"1¥");
    data.add(objData);

%>
:
:
<HTML>
<HEAD>
    <imarttag:imartDesignCss />
    <SCRIPT LANGUAGE="JavaScript">

        // 一覧の項目名リンクがクリックされた時の処理
        function onSortHeadClick( key )
        {
            // ...ここには各自で処理を記述する...
            :
        }

        // (1) で指定した JavaScript 関数の処理
        function onCheckBox()
        {
            // ...ここには各自で処理を記述する...
        }
    }
</SCRIPT>
</HEAD>
<BODY>
    <TABLE class="list_border_bg">
        <!-- リストヘッダー -->
        <imarttag:imartListHeader
            headerData="<%= data %>"
            sortKey="userId" />
    </TABLE>
</BODY>
</HTML>
```



表示結果			
<input type="checkbox"/>	ユーザコード	ユーザ名	会社／組織
:	:	:	:
:	:	:	:
(1)	(2)	(3)	(4)

### 3.4.6.5 クラスListHeaderObject

List インターフェースを実装したクラスのインスタンスに、このヘッダ情報クラス ListHeaderObject のインスタンスを格納して、属性 headerData に指定します。

jp.co.intra\_mart.foundation.core.taglib.ListHeaderObject

ListHeaderObjectクラスをインスタンス化するときに、使用するコンストラクタによって異なるヘッダ情報を生成することができます。※ コンストラクタの詳細は「3.4.6.5.2 コンストラクタ」を参照してください。

ヘッダ情報クラス ListHeaderObject に設定するヘッダ情報項目の内容について、以下の「表 3-8 ListHeaderObjectクラスの設定メソッド」に示します。

表 3-7 ListHeaderObject クラスに設定するヘッダ情報

設定内容	型	説明
項目 ID	java.lang.String	項目 ID の値を、属性 sortKey に指定すると、ソートキーになる。 項目 ID に "checkbox" を指定すると、ヘッダ内に「チェックボックス」が表示される。
項目名	java.lang.String	ヘッダに表示する項目名を指定する。 未指定の場合は、空のヘッダが表示される。 また、項目 ID="checkbox" を指定した場合は、 <INPUT type="checkbox">タグの name 属性となる。
ソート表示状態	boolean true : リンク表示 false: 通常表示	項目名の表示状態を指定する。 未指定の場合は、デフォルト値「true」が設定される ただし、項目 ID="checkbox" を指定した場合は、ソート対象外となるため、通常表示「false」が設定される。
項目名選択時の js 関数	java.lang.String	項目名がリンク表示のとき、リンクが選択された時の js 関数を指定する。 未指定の場合は、デフォルト値の「onSortHeadClick(項目 ID);」が設定される。 また、項目 ID="checkbox" を指定した場合は、「onClock イベントハンドラ」に設定する js 関数を指定する。
項目名の表示位置	java.lang.String center: 中央表示 left : 左寄せ right : 右寄せ	項目名の表示位置を指定する。 未指定の場合は、デフォルト値「center」が設定される。
項目に設定する任意の属性	java.lang.String	項目に設定する任意の属性を指定する。 (実際には各項目の<TD>タグの属性となる。) 複数の任意の属性を指定することも可能。 (指定方法は、3.4.6.4 使用例を参照してください。)

#### 3.4.6.5.1 注意事項

ListHeaderObject クラスをインスタンス化した際に、ソート表示状態が「リンク表示」に設定された場合は、項目名はリンク表示となります。

項目名のリンクがクリックされると、JavaScript 関数が呼び出されるため、本タグを使用した場合は、各項目名リンクに対しての JavaScript 関数の処理を記述する必要があります。

「項目名選択時の js 関数」を指定した場合は、その JavaScript 関数を記述してください。未指定にした場合は、デフォルトの JavaScript 関数の「onSortHeadClick」を必ず記述してください。

(詳細は「3.4.6.6 デフォルトの必須JavaScript関数」を参照)

## 3.4.6.5.2 コンストラクタ

ListHeaderObject クラスをインスタンス化するときに、使用するコンストラクタによって異なるヘッダ情報を生成することができます。

- public ListHeaderObject( )  
空のヘッダ情報オブジェクトを作成します。

- ◆ 引数なし。
- ◆ 空のヘッダ項目を作成する際に使用する。

- public ListHeaderObject( String id, String name )  
項目 ID と項目名を指定して、ヘッダ情報オブジェクトを作成します。

引数	型	説明
id	java.lang.String	項目 ID
name	java.lang.String	項目名

- ◆ 引数 id に「checkbox」を指定すると、ヘッダ内にチェックボックスが表示される。  
チェックボックスをクリックした際の onClick イベントは無し。
- ◆ ソート表示状態は、デフォルト値の「リンク表示(true)」になる。  
ただし、引数 id="checkbox" の場合は、「通常表示(false)」になる。
- ◆ 項目名選択時の js 関数は、デフォルト値の「onSortHeadClick('id');」になる。

- public ListHeaderObject( String id, String name, boolean sortLink )  
項目 ID と項目名とソート表示状態を指定して、ヘッダ情報オブジェクトを作成します。

引数	型	説明
id	java.lang.String	項目 ID
name	java.lang.String	項目名
sortLink	boolean	ソート表示状態 true : リンク表示 false : 通常表示

- ◆ 引数 id に「checkbox」を指定すると、ヘッダ内にチェックボックスが表示される。  
この場合、引数 sortLink の指定は無視される。  
チェックボックスをクリックした際の onClick イベントは無し。
- ◆ 項目名選択時の js 関数は、デフォルト値の「onSortHeadClick('id');」になる。

- public ListHeaderObject( String id, String name, String href )  
項目 ID と項目名と項目名選択時の js 関数を指定して、ヘッダ情報オブジェクトを作成します。

引数	型	説明
id	java.lang.String	項目 ID
name	java.lang.String	項目名
href	java.lang.String	項目名選択時の js 関数

- ◆ 引数 id に「checkbox」を指定すると、ヘッダ内にチェックボックスが表示される。
- ◆ ソート表示状態は、デフォルト値の「リンク表示(true)」になる。  
ただし、引数 id="checkbox" の場合は、「通常表示(false)」になる。
- ◆ ソート表示状態は、デフォルト値の「リンク表示(true)」になる。  
ただし、引数 id="checkbox" の場合は、「通常表示(false)」になる。
- ◆ 引数 id に「checkbox」を指定した場合、引数 href で指定した js 関数は、チェックボックスの onClick イベントハンドラに記述される。

## 3.4.6.5.3 メソッド

ヘッダ情報を設定するためのメソッドについて、以下の「表 3-8 ListHeaderObjectクラスの設定メソッド」に示します。

表 3-8 ListHeaderObject クラスの設定メソッド

メソッド	パラメータ	説明
setSortLink(boolean sortLink)	ソート表示状態 true : リンク表示 false : 通常表示	項目名の表示状態を指定する。 未指定の場合は、 デフォルト値「true」が設定される
setHref(String href)	項目名選択時の js 関数	項目名がリンク表示の場合、リンクが選択された時の js 関数を指定する。 未指定の場合は、デフォルト値の「onSortHeadClick('id');」が設定される。 また、項目 ID="checkbox" を指定した場合は、「onClock イベントハンドラ」に記述される js 関数を指定すること。
setAlign(String align)	項目名の表示位置 center : 中央表示 left : 左寄せ right : 右寄せ	項目名の表示位置を指定する。 未指定の場合は、 デフォルト値「center」が設定される。
setAttr(String attr)	項目の<TD>タグに設定する任意の属性の値	各項目の<TD>タグに指定する属性値を指定する。

3.4.6.6 デフォルトの必須JavaScript関数

リストヘッダの項目名リンクがクリックされた際に呼び出される、デフォルトの JavaScript 関数の詳細を以下に示します。

本タグを使用する際に、属性 `headerData` に指定する `ListHeaderObject` クラスをインスタンス化するときに、ソート表示状態が「リンク表示」で、「項目名選択時のjs関数」を指定しないでデフォルト値のまま設定した場合は、各自でデフォルトの JavaScript 関数の処理を記述する必要があります。

- ※ 必須の JavaScript 関数を記述しない場合の動作については、保証外とします。
- ※ コーディング方法などの詳細については、「3.4.6.4 使用例」を参照してください。
- ※ `ListHeaderObject` クラスをインスタンスするときに、「項目名選択時のjs関数」を指定した場合は、デフォルトの JavaScript 関数の処理を記述する必要はありません。

3.4.6.6.1 ソートの項目切り替え

属性 `headerData` に指定する `ListHeaderObject` クラスをインスタンスするときに、「項目名選択時のjs関数」を未指定にした場合、リストヘッダの項目名リンクがクリックされると、JavaScript 関数の `onSortHeadClick()` が呼び出されます。

以下に、その詳細を示します。

- 関数名  
`onSortHeadClick( key )`
- 概要  
リストヘッダの項目名リンクがクリックされた際に、呼び出される関数です。  
引数 `key` の値により、どのヘッダ項目が選択されたかの情報を取得することができます。

■ 引数

変数	内容	値	条件
key	ヘッダの項目 ID	(例) <code>userName</code>	リストヘッダの項目名リンクが選択された場合

※ 引数 `key` の値は、`ListHeaderObject` クラスをインスタンスするときに指定した「項目 ID」です。

## 4 画面デザインサンプル

共通でよく使用されると思われる、画面デザインのイメージサンプルを定義します。

このドキュメント内で示す画面全体図はあくまでイメージであり、「こんな用途の場合は、このような感じの画面にする」ということを規定しています。

規定されていない画面仕様が発生した場合は、以下に示すデザイン定義を基準として、各仕様に応じて作成してください。

### 4.1 検索系画面

検索画面についてのイメージを定義します。

検索条件入力画面と、検索結果一覧画面は、フレーム分けで一画面に表示するのではなく、別画面に遷移して表示してください。

#### 4.1.1 検索条件入力画面

- (1) 検索に必要な、検索条件項目を表示する。
- (2) [決定] ボタンは、検索条件項目の下部中央に配置する。

全体図のイメージについては、以下「図 4-1 検索条件入力画面の全体図」を参照してください。

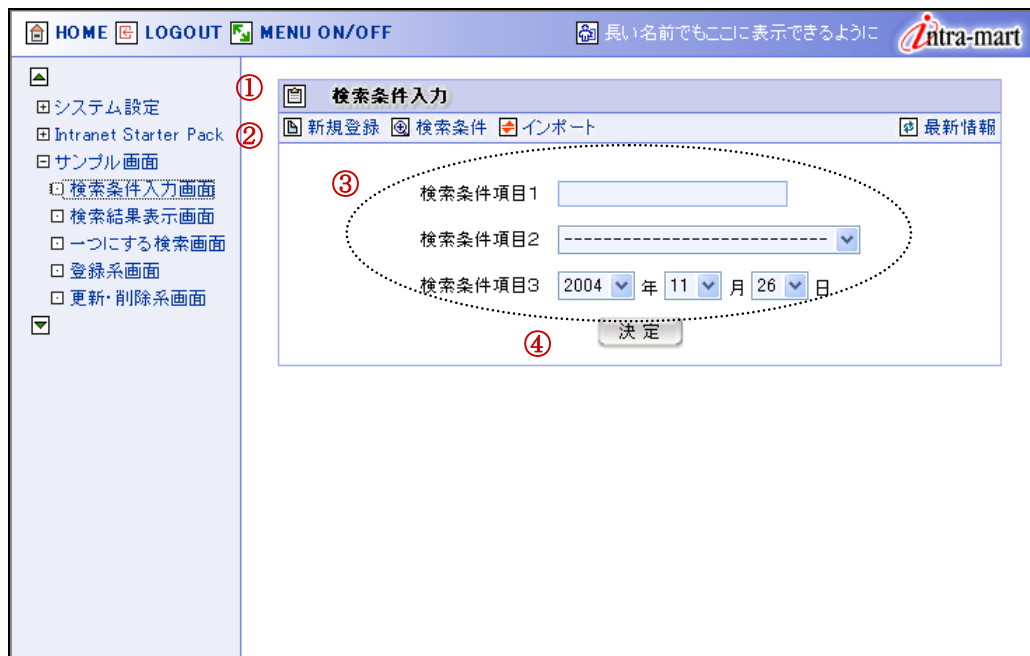


図 4-1 検索条件入力画面の全体図

検索条件入力画面の必須コントロールを、以下「表 4-1 検索条件入力画面の必須コントロール」に示します。

表 4-1 検索条件入力画面の必須コントロール

	コントロール	動作	備考(条件)
①	タイトルバー		
②	ツールバー		
③	検索条件項目	必要な検索条件を表示する	
④	[決定] ボタン	検索結果あり: 「4.1.2 検索結果一覧画面」 検索結果なし: 「検索結果なし画面」へ遷移する	

### 4.1.2 検索結果一覧画面

- (1) 検索条件入力画面で入力された、検索条件に一致する結果一覧を表示する。
- (2) 検索結果が存在しない場合は、「検索結果なし画面」へ遷移する。

全体図のイメージについては、以下「図 4-2 検索結果一覧画面の全体図」を参照してください。

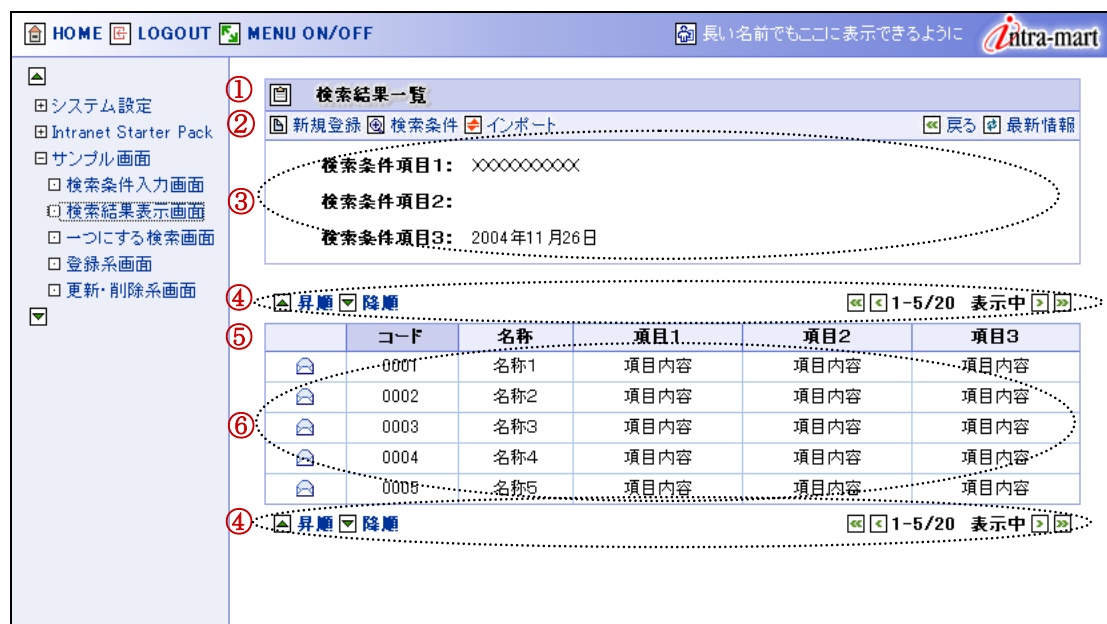


図 4-2 検索結果一覧画面の全体図

検索条件入力画面の必須コントロールを、以下「表 4-2 検索結果一覧画面の必須コントロール」に示します。

表 4-2 検索結果一覧画面の必須コントロール

	名称	コントロール	備考(条件)
①	タイトルバー	<IMART type="imTitleBar"> または <imarttag:imartTitleBar>	
②	ツールバー	<IMART type="imToolbarFrame"> または <imarttag:imartToolbarFrame>	
③	現在の検索条件	指定された検索条件を表示する	
④	リストコントロール ・[昇順/降順]ボタン ・[ページ切り替え]ボタン	昇順/降順で並べ替える ページを切り替える (詳細は「3.3.5 リストコントロール」または、 「3.4.5 リストコントロール」を参照)	一覧の上下に配置する
⑤	リストヘッダ ・一覧 [項目名]	項目名をクリックすると、ソートキーが切り替わる (詳細は「3.3.6 リストヘッダ」または、 「3.4.6 リストヘッダ」を参照)	ソートキー 項目名と通常項目 名は色分けする
⑥	一覧 [データ部]		1ページ10件まで

## 4.2 登録系画面

新規登録画面についてのイメージを定義します。

- (1) 登録情報の項目は、必須入力項目と通常入力項目を区別できるように色分けする。
- (2) [登録] ボタンは、登録情報項目の下部中央に配置する。

全体図のイメージについては「図 4-3 登録画面の全体図」を参照してください。

図 4-3 登録画面の全体図

登録画面の必須コントロールを、以下「表 4-4 登録画面の必須コントロール」に示します。

表 4-3 登録画面の必須コントロール

	コントロール	動作	備考(条件)
①	タイトルバー		
②	ツールバー		
③	登録情報項目	登録に必要な項目を表示する	必須項目には「(必須)」と記述する
④	[登録] ボタン	「更新・削除画面」へ遷移する	



### 4.3 更新・削除系画面

更新・削除画面についてのイメージを定義します。

- (1) 更新情報の項目は、必須入力項目と通常入力項目を区別できるように色分けする。
- (2) 更新情報項目の内容には、既存の登録情報を表示させる。
- (3) [更新]ボタン、[削除]ボタンは、登録情報項目の下部中央に配置する。
- (4) [削除]ボタンをクリックした後は、「削除確認」のダイアログメッセージを表示させること。

全体図のイメージについては、以下「図 4-4 更新・削除画面の全体図」を参照してください。

図 4-4 更新・削除画面の全体図

更新・削除画面の必須コントロールを、以下「表 4-4 登録画面の必須コントロール」に示します。

表 4-4 登録画面の必須コントロール

	コントロール	動作	備考(条件)
①	タイトルバー		
②	ツールバー		
③	更新情報項目	既存の登録情報を表示する	必須項目は色分けする
④	[更新] ボタン	仕様により遷移先が変わる	
⑤	[削除] ボタン	ボタン押下時に「削除確認」ダイアログメッセージを表示	仕様により動作の変更可能

## 4.4 一覧系画面

一覧系画面についてのイメージを定義します。

ユーザー一覧画面および会社・組織ツリー画面は、フレーム分けをしないで、画面遷移して表示してください。

### 4.4.1 ユーザー一覧

ユーザ検索画面(単一選択時)についてのイメージを定義します。

- (1) 検索条件入力画面で入力された、検索条件に一致する結果一覧を表示する。
- (2) 検索結果が存在しない場合は、別途検索結果なし画面へ遷移する。



図 4-5 ユーザー一覧画面の全体図(ポップアップ)

ユーザー一覧画面の必須コントロールを、以下「表 4-5 ユーザー一覧画面の必須コントロール(ポップアップ)」に示します。

表 4-5 ユーザー一覧画面の必須コントロール(ポップアップ)

	コントロール	動作	備考(条件)
①	ウィンドウタイトル	表示する文字は、リクエスト引数により変更可能	request.wnd_title
②	タイトルバー	表示する文字は、リクエスト引数により変更可能	request.message
③	ツールバー		
③-1	[検索条件]	「4.1.1 検索条件入力画面」へ遷移する(戻る)。戻り先の「検索条件入力」画面には、前回入力された検索条件を再表示する。	history.back()は使用不可
③-2	[閉じる]	ポップアップウィンドウを閉じる	
④	現在の検索条件	指定された検索条件を表示する	
⑤	[決定] ボタン	複数選択:メイン画面一覧に選択内容を追加 単一選択:選択を確定し、ウィンドウを閉じる	
⑥	リストコントロール ・[昇順/降順]ボタン ・[ページ切り替え]ボタン	昇順/降順で並べ替える ページを切り替える (詳細は「3.3.5 リストコントロール」または、「3.4.5 リストコントロール」を参照)	一覧の上下に配置する
⑦	リストヘッダ ・一覧 [項目名]	項目名をクリックすると、ソートキーが切り替わる (詳細は「3.3.6 リストヘッダ」または、「3.4.6 リストヘッダ」を参照)	ソートキー項目と通常項目は色分けする
⑧	一覧 [データ部]		1ページ10件まで
⑧-1	[ラジオボタン]or [チェックボックス]	[ラジオボタン]: 1件のみ選択できる [チェックボックス]: 複数選択できる	複数選択、単一選択により表示内容を変更する。

## 4.4.2 ロール一覧

ロール検索画面(複数選択時)についてのイメージを定義します。

- (1) 検索条件入力画面で入力された、検索条件に一致する結果一覧を表示する。
- (2) 検索結果が存在しない場合は、別途検索結果なし画面へ遷移する。

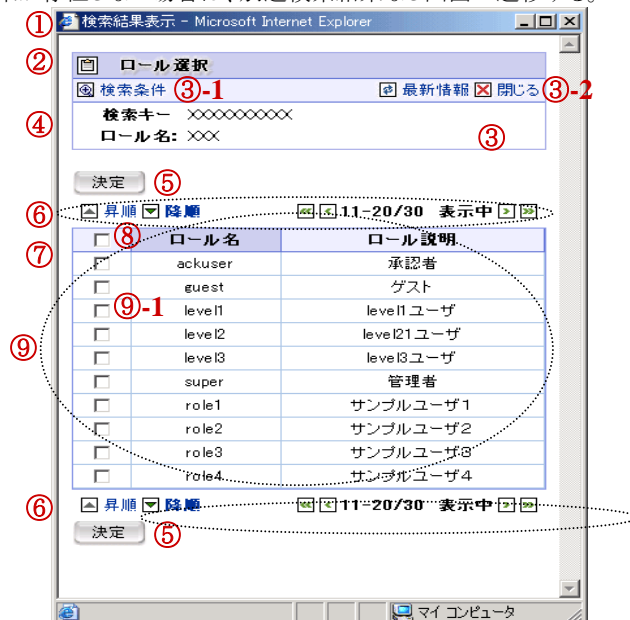


図 4-6 ロール一覧画面の全体図(ポップアップ)

ロール一覧画面の必須コントロールを、「表 4-6 ロール一覧画面の必須コントロール(ポップアップ)」に示します。


表 4-6 ロール一覧画面の必須コントロール(ポップアップ)

	コントロール	動作	備考(条件)
①	ウィンドウタイトル	表示する文字は、リクエスト引数により変更可能	request.wnd_title
②	タイトルバー	表示する文字は、リクエスト引数により変更可能	request.message
③	ツールバー		
③-1	[検索条件]	「4.1.1検索条件入力画面」へ遷移する(戻る)	history.back()は使用不可
③-2	[閉じる]	ポップアップウィンドウを閉じる	
④	現在の検索条件	指定された検索条件を表示する	
⑤	[決定] ボタン	複数選択:メイン画面一覧に選択内容を追加 単一選択:選択を確定し、ウィンドウを閉じる	一覧の上下に配置する
⑥	リストコントロール ・[昇順/降順]ボタン ・[ページ切り替え]ボタン	昇順/降順で並べ替える ページを切り替える (詳細は「3.3.5リストコントロール」または、「3.4.5リストコントロール」を参照)	一覧の上下に配置する
⑦	リストヘッダ ・一覧 [項目名]	項目名をクリックすると、ソートキーが切り替わる (詳細は「3.3.6リストヘッダ」または、「3.4.6リストヘッダ」を参照)	ソートキー項目と通常項目は色分けする
⑧	項目欄[チェックボックス]	チェックボックスの全選択/全解除ができる	複数選択の場合のみ表示
⑨	一覧 [データ部]		1ページ10件まで
⑨-1	[ラジオボタン]or [チェックボックス]	[ラジオボタン]: 1件のみ選択できる [チェックボックス]: 複数選択できる	複数選択、単一選択により表示内容を変更する。

## 5 共通アイコン

ツールバー、および画面内で主に使用される、共通のアイコンを以下に示します。  
アプリケーションで新たに必要なアイコンが発生した場合は、その都度作成し使用してください。





### 5.1 タイトルバー

アイコン	ファイルパス	内容
	images/standard/title.gif	タイトルバーのデフォルトアイコン (属性 icon 未指定時)





### 5.2 処理系

アイコン	ファイルパス	内容
	images/standard/refresh.gif	最新情報
	images/standard/help.gif	ヘルプ
	images/standard/close.gif	閉じる
	images/standard/arrow_left.gif	戻る
	images/standard/new.gif	新規登録・追加 など
	images/standard/delete.gif	削除・クリア など
	images/standard/edit.gif	編集
	images/standard/print.gif	印刷
	images/standard/search.gif	検索
	images/standard/copy.gif	コピー
	images/standard/inoutput.gif	情報の取り込み・取り出し
	images/standard/leader.gif	極秘
	images/standard/user_setting.gif	個人設定
	images/standard/list.gif	表示
	images/standard/info.gif	情報





### 5.3 検索種別

アイコン	ファイルパス	内容
	images/standard/company_group.gif	会社／組織
	images/standard/offical_group.gif	パブリックグループ
	images/standard/private_group.gif	プライベートグループ
	images/standard/role.gif	ロール





## 5.4 ページ切替

アイコン	ファイルパス	内容
	images/standard/previous.gif	前のページ
	images/standard/arrow_l2.gif	最初のページ
	images/standard/arrow_r1.gif	次のページ
	images/standard/arrow_r2.gif	最後のページ







## 5.5 昇順／降順切替

アイコン	ファイルパス	内容
	images/standard/up.gif	昇順ソート(有効時)
	images/standard/up_none.gif	昇順ソート(無効時)
	images/standard/down.gif	降順ソート(有効時)
	images/standard/down_none.gif	降順ソート(無効時)



## 5.6 選択リストボックス

アイコン	ファイルパス	内容
	images/standard/arrow_up.gif	上に移動
	images/standard/arrow_down.gif	下に移動
	images/standard/arrow_left.gif	右のリストボックスに移動
	images/standard/arrow_right.gif	左のリストボックスに移動

## 5.7 ツリー表示

アイコン	ファイルパス	内容
	images/standard/home_icon.gif	ルート
	images/standard/folder_close.gif	下階層を閉じたとき
	images/standard/folder_open.gif	下階層を開いたとき
	images/standard/item.gif	メニュー項目
	images/standard/up.gif	すべて閉じる
	images/standard/down.gif	すべて開く

## 5.8 ボタン・タブ

アイコン	ファイルパス	内容
	左枠: images/standard/button_left.gif 文字表示部: images/standard/button_middle.gif 右枠: images/standard/button_right.gif	実行ボタン 文字表示部は文字列幅に合わせ伸縮
	左枠: images/standard/portal_tab_left.gif 文字表示部: images/standard/portal_tab_middle.gif 右枠: images/standard/portal_tab_right.gif	タブ 文字表示部は文字列幅に合わせ伸縮







intra-mart WebPlatform/AppFramework Ver. 7.0  
画面デザインガイドライン

2012/03/26 第3版

Copyright 2000-2012 株式会社NTTデータ イントラマート  
All rights Reserved.

TEL: 03-5549-2821

FAX: 03-5549-2816

E-MAIL: [info@intra-mart.jp](mailto:info@intra-mart.jp)

URL: <http://www.intra-mart.jp/>