UIデザインガイドライン（ PC 版）

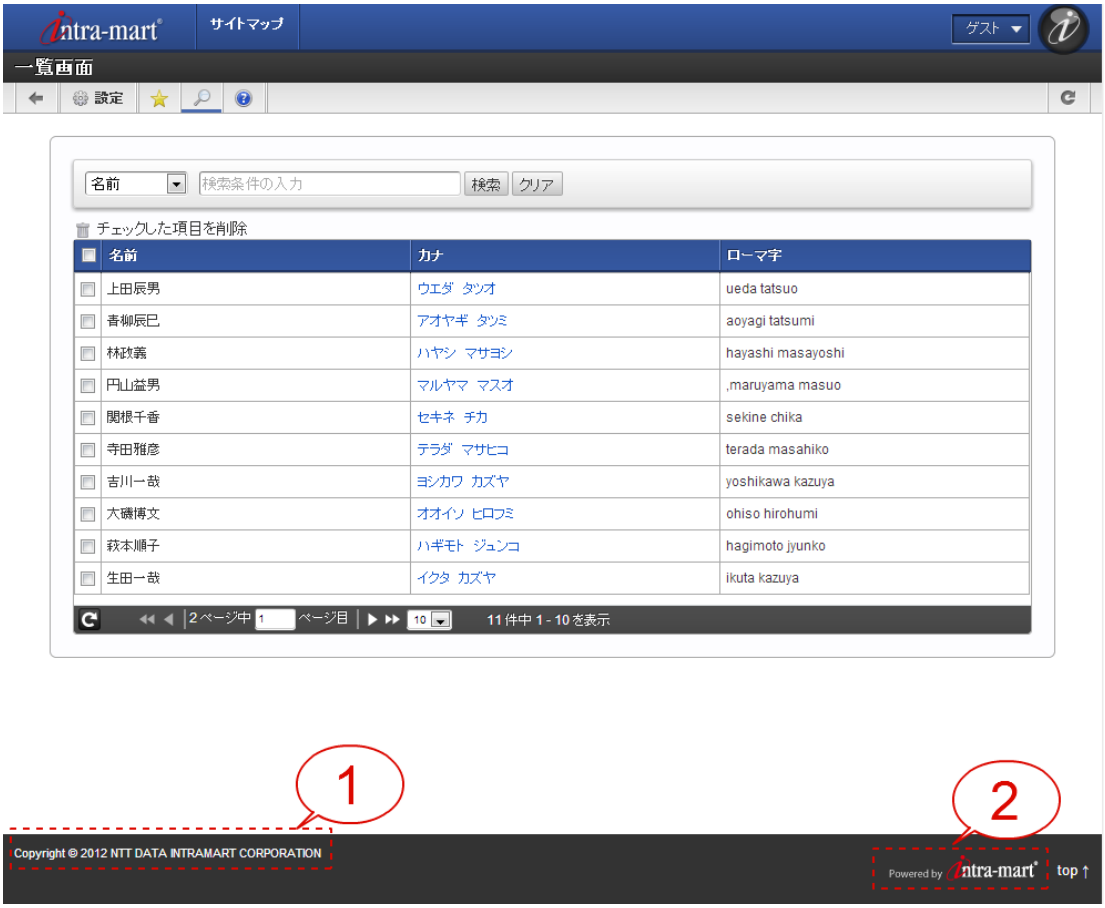
改訂情報

変更年月日	変更内容
2014-01-01	初版
2014-04-01	第２版 ■ 「*ツールバー*」にツールバーに配置可能な「文字列」の説明を追加 ■ 「*ツールバー*」の「*4 タブアイコン*」の注意を修正 ■ 「*画面タイトル*」の画面キャプチャを貼り換え
2014-08-01	第３版 ■ 「*テーブル*」の「セル内の位置／文字寄せ（align）」を修正
2014-12-01	第４版 ■ 「*バリデーション*」の「JSSP Validation と連携する場合」にコラムを追加
2015-04-01	第５版 ■ TERASOLUNA Global FrameworkをTERASOLUNA Server Framework for Java (5.x)に変更

禁止事項

intra-mart Accel Platform は、使用許諾によって以下を禁止しています。

1. Copyright を変更・削除しないでください。
2. Powered by intra-mart の画像を変更・削除しないでください。



はじめに

項目

本書の目的

本書では、intra-mart Accel Platform における、UIデザインガイドラインを説明します。 開発者は、本書を利用することで、intra-mart Accel Platform と統一された画面開発を行うことができます。

intra-mart Accel Platform では、「UI基本方針」を定め、それに則した「UIデザイン」を定めました。 「UIデザイン」では、 intra-mart Accel Platform で提供するインタフェースを使用した画面開発とルールを説明します。

intra-mart Accel Platform で提供するインタフェースは、「UIモジュール」と「テーマ」です。 「UIモジュール」は、画面を構成する部品群です。開発者は「UIモジュール」を利用し、intra-mart Accel Platform と同じデザインの画面開発を行えます。 「テーマ」は、画面レイアウトを切り替えるための仕組みです。 開発者はこの仕組みを利用することにより、簡単に画面レイアウトを切り替えられ、コーディングの負担が減ります。

これらのインタフェースを利用し、「UIデザイン」に則り、「基本的な画面の作り方」を参考に画面開発を行ってください。 旧バージョンに比べて、より早く、より簡単に intra-mart Accel Platform と同じユーザインタフェースの画面開発を行えるでしょう。

> **コラム**
>
> 本書の画面キャプチャは、全てGoogle Chromeにて表示しています。
> Web ブラウザの違いやバージョンによって、グラデーションや影、丸み等の見え方が異なります。
> これは、Web ブラウザにより表示の解釈が異なるためと、ブラウザによりCSS3の対応状況が異なるためです。

対象読者

次の開発者を対象としてます。

- intra-mart Accel Platform のUIデザインガイドラインを確認したい方
- intra-mart Accel Platform で初めてプログラミングを行う開発者

本書の構成

本書の構成は以下の 3 部構成です。

- *UI基本方針*
  intra-mart Accel Platform のアプリケーションを作成するときの考え方を説明します。

- *UIデザイン*
  UIデザインの流れに沿って、部品の配置から画面遷移のルールなど説明します。

- *基本的な画面の作り方*
  実際の画面のサンプルを使用して説明します。

> **注意**
>
> *UIデザイン* は、スクリプト開発を用いて説明します。
> 外部ドキュメントへのリンクは全てスクリプト開発用となりますので、SAStruts+S2JDBC で開発している場合は、適宜読み替えてください。

UI基本方針

intra-mart Accel Platform におけるUI基本方針は、以下の通りです。

- 使用するユーザの目線を忘れないこと。
- 操作方法が分かりやすく、迷わないこと。
- 同じ動きをするものは、操作方法が統一されていること。
- 画面デザインが全体で統一されていること。

コラム
ユーザにとって、「使いやすさ」や「見た目」と「動き」がアプリケーションへの評価となります。

対応環境条件

項目

クライアント要件

対応するOS、ブラウザは、intra-mart Accel Platform リリースノートの クライアント要件 を参照してください。 最新の情報が確認できます。

注意

上記リリースノートは、最新のバージョンです。
古いバージョンのクライアント要件は、 以下を手順で参照してください。

1. Products Information Site を表示します。
2. 「intra-mart Accel Platform 旧アップデート版 リリースノート / 公開ソースダウンロード」の該当バージョンをクリックします。
3. 「リリースノート」をクリックします。
4. 「システム要件」をクリックします。
5. 「クライアント要件」を確認します。

コラム

本書の画面キャプチャは、全てGoogle Chromeにて表示しています。
Web ブラウザの違いやバージョンによって、グラデーションや影、丸み等の見え方が異なります。
これは、Web ブラウザにより表示の解釈が異なるためと、ブラウザによりCSS3の対応状況が異なるためです。

画面解像度

対応する画面の解像度は、以下の通りです。

- intra-mart Accel Platform は 1024 x 768 を基準とします。

Webブラウザ設定

- IEは、文字サイズを「中」とします。
- 表示倍率（ズーム）は「100%」とします。

UIデザイン

本章よりUIデザインの流れにそった画面作成を説明します。 目次は以下のとおりです。

UIデザインの流れ

UIデザインの流れを簡単に説明します。

> **注意**
> スクリプト開発にて説明します。
> 外部ドキュメントへのリンクは全てスクリプト開発用となりますので、SAStruts+S2JDBC で開発している場合は、適宜読み替えてください。

- *画面レイアウト* を指定します。
  - 基本的な画面レイアウトの方法を確認します。
  - 「テーマ」を利用し、画面構成のパーツを組み合わせた画面レイアウトを行います。
- *body 部分* を作成します。
  - まずは、*UIモジュール* にどんな部品があるかを確認します。
  - 初めに *画面タイトル* を作成します。
  - 戻るやイベントを配置する *ツールバー* を作成します。
  - *コンテンツエリア* を作成します。
    - *見出し* にて、画面をセクションごとに分けます。
    - *テーブル* にて、表のルールを確認し、配置します。
    - *入力フォーム* にて、部品ごとのルールを確認し、配置します。
    - *ボタン* にて、種類や画面ごとのルールを確認し、配置します。
    - *処理リンク／処理アイコン* にて、の種類やルールを確認し、配置します。
    - *アラートとコンファーム* の表示方法やルールを確認、表示します。
- 作成した画面を *画面遷移* の規約を確認し、実装します。
- *エラー処理* を確認します。
- 入力項目の国際化を行う場合は、 *国際化情報入力項目* を確認します。

画面レイアウト

本章では、画面レイアウトの方法について説明します。
intra-mart Accel Platform では、「テーマ」を利用して、画面レイアウトの大枠（ヘッダ、フッタの有無など）を作成します。
「テーマ」を利用すると、開発者が作成するのは、 HTML の要素の body 部分だけです。 head 、 header 、 body 、 footer をそれぞれコーディングする必要はありません。

> **注意**
> 本章は、「テーマ」を利用して、画面レイアウトを行う方法を説明します。 body 部分にコンテンツを配置する説明は、 *body 部分* にて説明します。

> **コラム**
> 本書の画面キャプチャは、「標準テーマ（青）」を用いています。

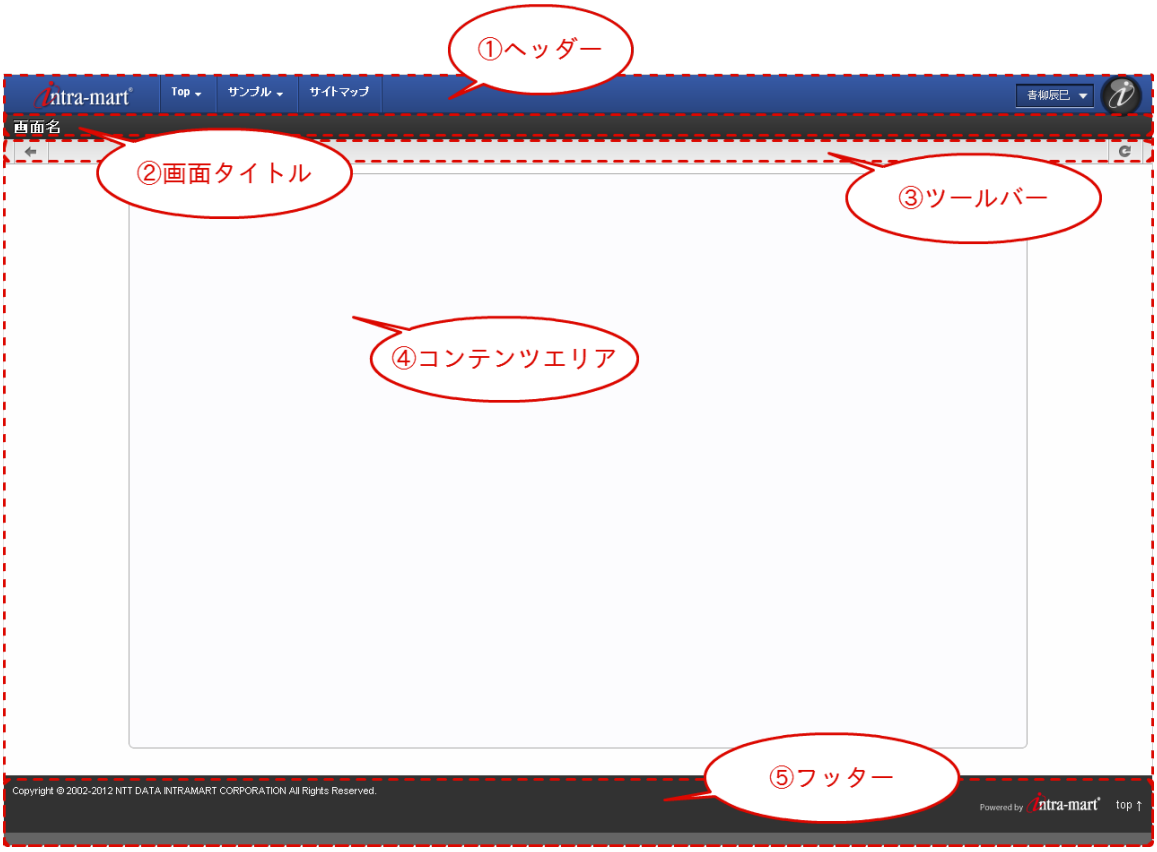
画面レイアウトの目次は以下の通りです。

基本的な画面レイアウト

本章では、全ての構成要素を含んだ画面レイアウトを用いて、画面構成を説明します。 構成要素の組み合わせによる画面レイアウトの方法は、*テーマを使って画面レイアウトを行う* で説明します。

基本的な画面構成

基本的な画面は、 6 つの構成要素が含まれます。



6 つの構成要素、4 つのパーツ名

6 つの構成要素は、本書では 4 つのパーツに分類します。 パーツ名の詳細は、 *テーマからみた画面の 4 つのパーツ* にて行います。

No	場所	HTML 要素	パーツ名	配置内容
①	ヘッダー	<body>配下の<header>	head	メニュー、My Menu
②	画面タイトル	<body>	body	画面タイトル
③	ツールバー	<body>	body	処理アイコン、処理リンク（アイコンのみ、アイコン＋リンク、リンクのみ）
④	コンテンツエリア	<body>	body	入力フォーム、一覧、ボタン等
⑤	フッター	<body>配下の<footer>	footer	コピーライト
⑥	ヘッド情報	<head>	head	文書のヘッダ情報です。画面上に表示されません。

HTML 構造は以下の通り出力されます。

```
<html>
 <head>⑥ヘッド情報</head>
 <body>
  <header>①ヘッダー</header>
  <div>
   ③ツールバー
   ④コンテンツエリア
  </div>
  <footer>⑤フッタ</footer>
 </body>
</html>
```

> **注意**
> 基本的な構成要素の実装例で触れますが、開発者は、 ①のヘッダー、⑤のフッターをコーディングする必要はありません。 html タグや body タグ等のタグも不要です。 *テーマを使って画面レイアウトを行う* で詳しく説明します。

基本的な構成要素の実装例

実際に intra-mart Accel Platform では、以下のようにコーディングします。 ①のヘッダー、⑤のフッター、⑥のヘッダ情報、html タグや body タグは、「テーマ」が自動的に生成します。

> **注意**
> 「テーマ」の機能を使用して、ヘッダーやフッターの有無をレイアウト指定できます。 *テーマを使って画面レイアウトを行う* で詳しく説明します。

- HTML 実装例

```
<!-- ⑥head タグ-->
<imart type="head">
 <title>画面名</title>
 <script>
  function hoge(){
   doSomething();
  }
 </script>
</imart>
<!-- ②画面タイトル -->
<div class="imui-title">
 <h1>画面タイトル</h1>
</div>

<!-- ③ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
  <ul class="imui-list-toolbar">
   <!-- 戻る -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
  <!-- ツールバー右側 -->
  <ul class="imui-list-toolbar-utility">
   <li><a href="#" class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span>
</a></li>
  </ul>
 </div>
</div>

<!-- ④コンテンツエリア -->
<div class="imui-form-container">

</div>
```

- JavaScript 実装例

```
function init(request) {
 doSomething();
}
```

- コーディング内容

No	場所	配置内容
①	ヘッダー	テーマが自動的に生成します。明示的にコーディングする必要はありません。
②	画面タイトル	<h1>に、画面タイトルを入力します。
③	ツールバー	*ツールバー* を参照してください。
④	コンテンツエリア	*body 部分* を参照してください。*UIモジュール* や独自に用意した表などのコンポーネントを配置します。
⑤	フッター	テーマが自動的に生成します。明示的にコーディングする必要はありません。
⑥	ヘッド情報	テーマが自動的に生成します。画面上表示されていませんが、head タグが出力されています。 ▪ title タグに画面名を記述してください。 ▪ 必要に応じて、script 、 link タグなどを記述してください。

コラム

コンテンツエリアは、 div 要素内に記述します。div に設定すべき class 属性は、imui-form-containerを含め 3 つ用意しています。
次の項 *固定レイアウト（1カラム）* にて説明します。

基本的な画面レイアウト別実装例

以下に基本的な画面レイアウト別の実装例を示します。

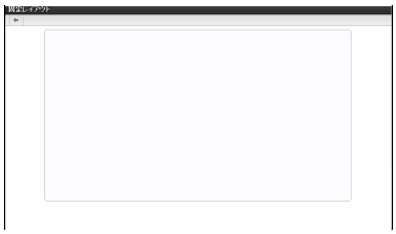
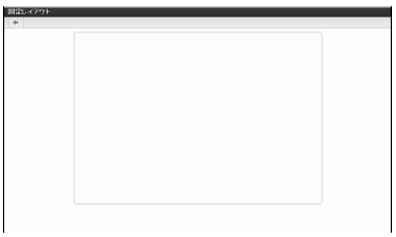
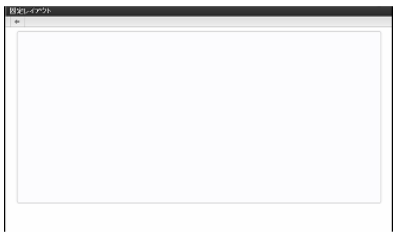
固定レイアウト（1カラム）

コンテンツエリアの一番外側の div の class 属性の説明です。

- 一番外側の div の class 属性に **imui-form-container** 、 **imui-form-container-narrow** 、 **imui-form-container-wide** のいずれかのコンテナクラスを指定します。

```
<div class="imui-form-container">
  ...
</div>
```

- 指定した class により枠の幅 (%) が変わります。（サンプル画像では style=height:300px を指定）

imui-form-container	imui-form-container-narrow	imui-form-container-wide
width:75%	width:60%	width:90%
		

コラム

imui-form-container 、 imui-form-container-narrow 、 imui-form-container-wide は、*UIモジュール* の 1 つです。
別ドキュメントの CSS Module List の「コンテナ」で説明しています。

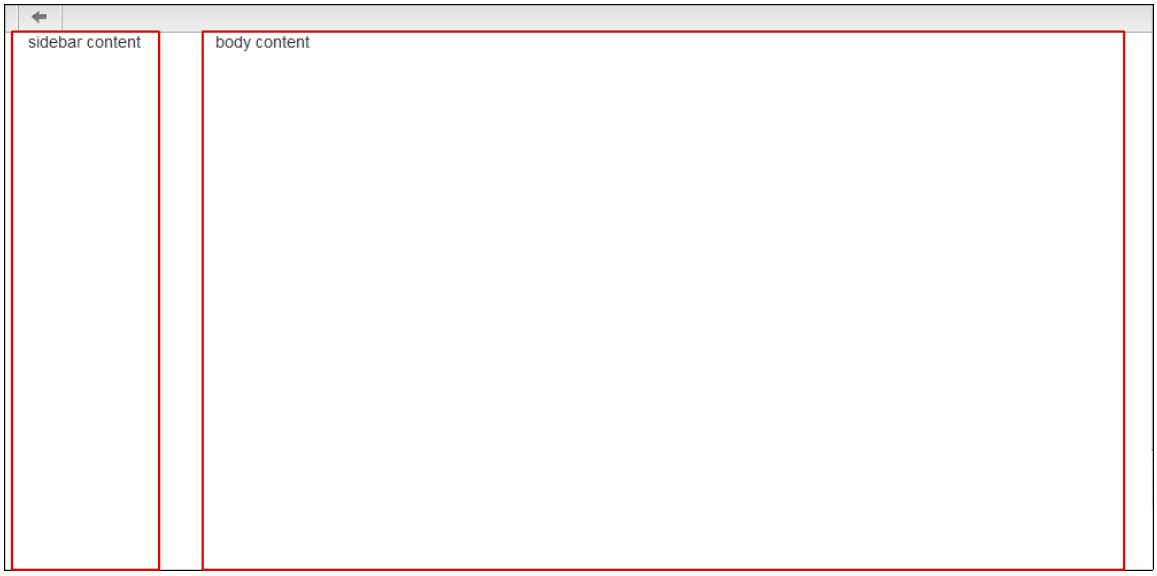
左右に分割するレイアウト（2カラム）

コンテンツエリアを左右に分割したい場合の実装例です。

- Twitter の Bootstrap を利用します。
- 実装例

```
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span2" style="height:400px;">
      <div>sidebar content</div>
    </div>
    <div class="span10" style="height:400px;">
      <div>body content</div>
    </div>
  </div>
</div>
```

- 画面



テーマを使って画面レイアウトを行う

基本的な画面レイアウト で触れたとおり、開発者はヘッダー 、フッター等のコーディングする必要がありません。
intra-mart Accel Platform が提供するインタフェース「テーマ」を利用して作成します。
「テーマ」を利用すると、開発者が作成するのは、body 部分だけです。
開発者は、「テーマ」を利用することで、共通部分に修正が入った時の手間や、記述ミスによる不具合を避けることができます。
本章では、テーマを使って画面レイアウトを行う方法を説明します。
さらに、構成要素の組み合わせ方についても説明します。

項目

コラム

「標準テーマ（青）」を用いて説明します。

テーマとは

「テーマ」とは、以下を指します。

- 画面レイアウトやスタイルを切り替える仕組み
- その構成ファイル群

本書では「テーマ」を利用して **画面レイアウトを指定する方法** を説明をします。

注意

本ドキュメントでは、「テーマ」を利用した画面レイアウトの方法を説明します。
テーマの仕組みを知りたい、テーマの色を変更したい、ロゴを変更したい場合は、以下の別ドキュメントを参照してください。

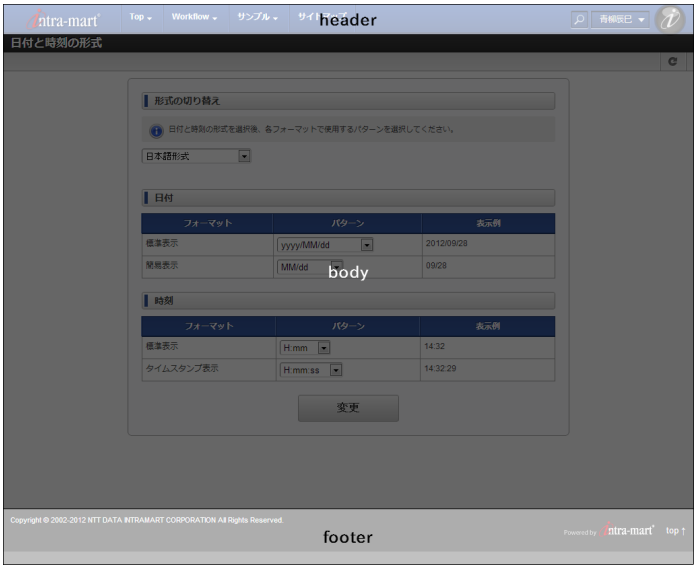
- 標準テーマカスタマイズ セットアップガイド
- 標準テーマカスタマイズ 操作ガイド

テーマの仕組みの詳細は、以下の別ドキュメントを参照してください。

- テーマ仕様書

テーマからみた画面の 4 つのパーツ

まず、テーマからみた画面の考え方を説明します。
基本的な画面レイアウト では 6 つの構成要素、 4 つのパーツとして説明しました。
この 4 つのパーツ（ head 、 header 、 body 、 footer ）は、「テーマ」の仕組みから、分類しています。



No	場所	パーツ名
①	ヘッダー	header
②	画面タイトル	body
③	ツールバー	body
④	コンテンツエリア	body
⑤	フッター	footer
⑥	ヘッド情報	head

> **コラム**
> テーマを使った画面レイアウトの組み合わせ方法は、パーツ名で説明します。

画面レイアウトの指定方法

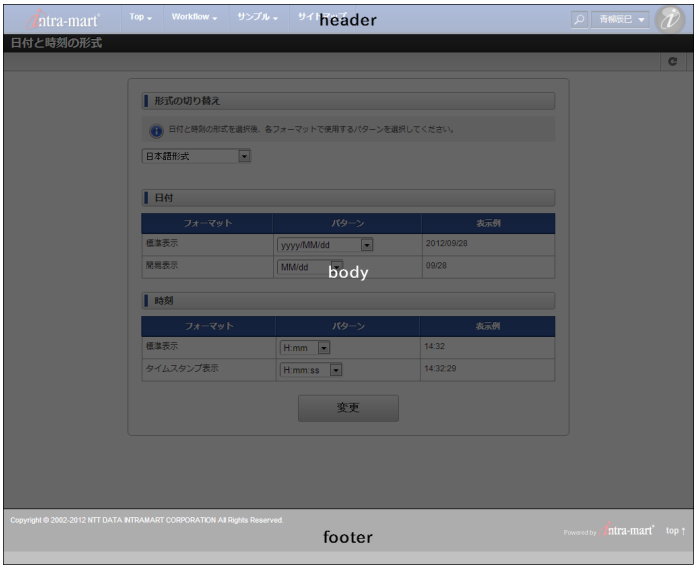
本項では、「テーマ」を利用した画面レイアウトの方法を説明します。
「テーマ」を利用すると、画面によって header を表示しない、作成した画面だけを表示したい、など組み合わせることが可能です。

> **コラム**
> テーマの画面レイアウトやスタイルを切り替える仕組みの中の PageBuilder を使い、画面レイアウトの指定します。
> 詳細は、 テーマ仕様書 の PageBuilder を参照してください。

項目

画面を構成する 4 つのパーツ

テーマからみた画面の 4 つのパーツ に示したとおり、intra-mart Accel Platform の画面レイアウトは、テーマの観点から head 、 header 、 body 、 footer の 4 つの パーツに分類されます。



この 4 つのパーツは、標準テーマ、標準（シンプル）テーマにおいては、以下の通り HTML 要素として出力します。

パーツ名	HTML 要素	備考
head	head タグ	画面上には表示されません。
header	body タグ内の header タグ	標準テーマにおいては、グローバルナビ等が表示されます。
body	body タグ内の div.imui-container タグ	開発者が作成した画面タイトル、ツールバーやコンテンツが表示されます。
footer	body タグ内の footer タグ	標準テーマにおいては、コピーライト等が表示されます。

HTML ソースは以下の通り出力されます。

```
<html>
 <head>ヘッド情報</head>
 <body>
  <header>ヘッダー</header>
  <div class="imui-container">
   画面タイトル
   ツールバー
   コンテンツエリア
  </div>
  <footer>フッタ</footer>
 </body>
</html>
```

パーツ組合せの種類

4 つのパーツの組合せの種類を説明します。
組合せの種類は以下 5 つです。

- head 、 header 、 body 、 footer ヘッダー、フッタ表示あり。
- head 、 body 、 footer ヘッダー表示なし。フッタ表示あり。
- head 、 body ヘッダー、フッタ表示なし。
- body ヘッダー、フッタ表示なし。
- テーマ適用無し ヘッダー、フッタ表示なし。

これらのパーツの組合せは、 PageBuilder にて実装を行います。

PageBuilder で実装可能な画面レイアウトの種類

PageBuilder で実装可能な画面レイアウトの 6 種類を説明します。
（パーツの組み合わせ方のうち、 head 、 body の組み合わせが 2 パターンあります。）

1. HeadWithFooterThemeBuilder

- head 、 body 、 footer を含んだ HTML を生成します。
- header （メニューや、ユーティリティ）を表示したくないが、 footer は表示したい場合に使用します。
- body は、 で囲まれて出力されます。

2. HeadWithContainerThemeBuilder

- head 、 body を含んだ HTML を生成します。
- header （メニューや、ユーティリティ）、 footer を表示したくないが、 CSS やクライアントサイド JavaScript は使用したい場合に使用します。
- 主に intra-mart Accel Platform 向けに作成した画面を表示するために使用することを想定しています。
- body は、 で囲まれて出力されます。

3. HeadOnlyThemeBuilder

- head 、 body を含んだ HTML を生成します。
- header （メニューや、ユーティリティ）、 footer を表示したくないが、 CSS やクライアントサイド JavaScript は使用したい場合に使用します。
- 主に iWP7.2 以前のシステム向けに作成した画面を表示するために使用することを想定しています。
- body は、指定された URL の HTML そのものが出力されます。

4. BodyOnlyThemeBuilder

- DOCTYPE、htmlタグ、body を含んだ HTML を生成します。
- header （メニューや、ユーティリティ）、 footer を表示せず、 CSS やクライアントサイド JavaScript も使用しない場合に使用します。
- body は、指定された URL の HTML そのものが出力されます。

5. NoThemeBuilder

- 指定された URL の HTML をそのまま返します。
- テーマを一切使用せず、自分で作成した HTML をそのまま出力したい場合に使用します。
- body は、指定された URL の HTML そのものが出力されます。

6. FullThemeBuilder

- head 、 header 、 body 、 footer のすべてを含んだ HTML を生成します。
- body は、 で囲まれて出力されます。
- 基本はこれを使用します。

注意

パーツの組み合わせ方のうち、 head 、 body を含んだ HTML を生成する実装は、 HeadWithContainerThemeBuilder と HeadOnlyThemeBuilder の 2 つあります。
2 つの違いは、 HeadWithContainerThemeBuilder は、 **bodyを で囲んで出力します** が、 HeadOnlyThemeBuilder は、**出力されません**。
intra-mart Accel Platform の *UIモジュール* を利用する場合は、 HeadWithContainerThemeBuilder を指定してください。

コラム

FullThemeBuilder は、 *基本的な画面レイアウト* の説明で使用した全ての構成を含むレイアウトです。

PageBuilderを利用した画面レイアウトの 3 つの指定方法

PageBuilderを利用した画面レイアウトの指定方法は 3 つあります。

- *設定ファイルに指定する方法*
  パーツの組み合わせ方を静的に決定する場合
- *リクエストへ属性／パラメータで指定する方法*
  パーツの組み合わせ方を動的に決定する場合や、forward する場合
  以下 2つの指定方法があります。
  - リクエストに属性として指定する方法
  - リクエストパラメータを指定する方法

適用順位

画面レイアウトの 3 つの指定方法には適用順位があります。
設定ファイル、パラメータ、属性の適用は、以下の順に検索し、最初に合致した PageBuilder を使用します。
設定ファイルに記述したものより、属性に指定したものの方が優先されます。

1. 属性
2. パラメータ
3. 設定ファイル

設定ファイルに指定する方法

パーツの組み合わせ方を静的に決定する場合、設定ファイルに記述します。
手順は以下の通りです。

1. 設定ファイルを配置する。
2. 設定ファイルの内容を指定する。

設定ファイルは、WEB-INF/conf 配下の PageBuilder の実装毎のフォルダに配置します。
ファイル名は任意です。

適用したい **PageBuilder**	指定する値
HeadWithFooterThemeBuilder	WEB-INF/conf/theme-head-with-footer-path-config
HeadWithContainerThemeBuilder	WEB-INF/conf/theme-head-with-container-path-config
HeadOnlyThemeBuilder	WEB-INF/conf/theme-head-only-path-config
BodyOnlyThemeBuilder	WEB-INF/conf/theme-body-only-path-config
NoThemeBuilder	WEB-INF/conf/theme-no-theme-path-config
FullThemeBuilder	なし

注意

それぞれの設定ファイルは異なる XML Schema で定義されています。
いずれかの設定ファイルを別のフォルダにコピーしても動作しないので注意してください。

設定ファイルに指定する例

設定ファイルで指定する方法の例として、*http://hostname/iap/sample/page* へのリクエストを head 、 body 、 footer を含んだ HTML としたい場合を取り上げます。
この場合、使用する PageBuilder は、 HeadWithFooterThemeBuilder になります。

HeadWithFooterThemeBuilder の設定ファイルは以下のようになります。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <theme-head-with-footer-path-config xmlns="http://www.intra-mart.jp/theme/theme-head-with-footer-path-config">
3    <path>/sample/page</path>
4 </theme-head-only-path-config>
```

path の中に、コンテキストパス以下のパスを、”/” から記述します。

別の例として、*http://hostname/iap/example/{parameter1}*、*http://hostname/iap/example/{parameter1}/{parameter2}* へのリクエストを異なるビルダーモジュールで表示する場合を取り上げます。この場合、正規表現を利用して path を表現します。path 要素に regex 属性を true として追加することで正規表現として扱われます。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <theme-head-with-container-path-config xmlns="http://www.intra-mart.jp/theme/theme-head-with-container-path-config">
3    <path regex="true">/example/[^/]+?</path>
4 </theme-head-only-path-config>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <theme-head-with-footer-path-config xmlns="http://www.intra-mart.jp/theme/theme-head-with-footer-path-config">
3    <path regex="true">/example/[^/]+?/[^/]+?</path>
4 </theme-head-only-path-config>
```

リクエスト属性／パラメータで指定する方法

パーツの組み合わせ方を動的に決定する場合や、forward する場合、リクエスト属性またはパラメータを指定します。
forward を行うと、PageBuilder が処理対象とする URL は forward 前の URL となります。
forward 後のページに対して forward 前の PageBuilder とは別の PageBuilder を指定したい場合、リクエストにパラメータを指定することで PageBuilder を切り替えることができます。

指定するキー　imui-theme-builder-module

適用したい **PageBuilder**	指定する値
HeadWithFooterThemeBuilder	headwithfooter
HeadWithContainerThemeBuilder	headwithcontainer
HeadOnlyThemeBuilder	headonly
BodyOnlyThemeBuilder	bodyonly
NoThemeBuilder	notheme

上記の値をリクエストのパラメータ、または属性として指定することで PageBuilder が切り替わります。

リクエストに属性として指定する例

```
function init(request) {
   request.setAttribute("imui-theme-builder-module", "headwithfooter");
   forward("somewhere");
}
```

リクエストパラメータとして指定する例

```
<form name="form" action="sample/page">
  <input type="hidden" name="imui-theme-builder-module" value="headwithfooter">
  <input type="submit" value="submit"/>
</form>
```

画面レイアウトの作成ルール

項目

作成ルール

1. html 、 body 、 head を書いてはいけません。
2. title を書きましょう。
3. <imart type=”head”> を使って script 、 link を書きましょう。
4. <body onload=”func”> は $(document).ready(func) に置き換えましょう。

テーマで指定してあるので変更できないもの

以下は、テーマで指定してあるため、変更できません。

- DOCTYPE
  <!DOCTYPE html>
- charset
  <meta charset=”UTF-8”>
- base
  <base href=”*http://hostname:portnumber/iap*” target=”_self”/> ホスト名などはシステムによって異なります。
- favicon
  <link rel=”icon” href=”favicon.ico” type=”image/x-icon” /> <link rel=”Shortcut Icon” type=”img/x-icon” href=”favicon.ico” />
- 必須の CSS
  - テーマ固有の CSS

theme.css
    - 後述の UI コンポーネント固有の CSS
      imui.css
    - Twitter Bootstrap の CSS
      bootstrap.css
- 必須の JavaScript ライブラリ
    - jQuery
    - jQueryUI
    - jQuery 、 jQueryUI Plugin
    - 後述の UI コンポーネントが定義してある JavaScript
      imui.js
    - 画面遷移のユーティリティが定義してある JavaScript
      imui-form-util.js
    - im_json.js 、im_window.js
      メニュー制御で使用します。
    - html5.js
      IE8 で intra-mart にアクセスした際に読み込まれます。このライブラリを読み込むことによって、HTML5 で追加されたタグを利用することができるようになります。

指定しなくてはならないもの

タイトルタグ <title></title>

すべての画面に説明的なタイトルを記述しなければなりません。 タイトルに指定する内容は、アプリケーション名、画面名、操作対象などから構築します。 これらを詳細なものから広範なものへと並べ、- （ハイフン） でつなげることを推奨します。

スケジュールの参照画面を例にとると、

リソース名	例
アプリケーション名	スケジューラ
画面名	予定の参照
操作対象名	対象の予定のタイトル

のようになり、title タグは以下のようになります。

```
1 <title>打ち合わせ - 予定の参照 - スケジューラ</title>
```

titleタグに指定した文字列は、以下のような用途に使われます。

- ブラウザのお気に入り （ブックマーク） へ登録する際の名前の初期値
- ブラウザウィンドウ、またはタブの名前

ユーザがお気に入りに登録することを考慮すると、ユーザごとに一意なタイトルを記述することを推奨します。 タイトルが一意になることで、お気に入りの中で重複しなくなるためです。

任意で指定するもの

独自で追加したい CSS や JavaScript

- 独自 CSS
    - <link rel=”stylesheet” type=”text/css” href=”somewhere/cssfile”>
    - ページ独自の css ファイルへの参照を記述する
- 独自 JavaScript
    - <script src=”somewhere/csjs.js”>
    - ページ独自のクライアント JavaScript ファイルへの参照を記述する

ただし、これらのタグを head タグに直接記述することはできません。 これらのタグを記述したい場合は <imart type=”head”> タグの中に追記したいタグを記述します。 <imart type=”head”> タグの中に記述した内容が head タグの必須のスクリプトの後に追加されます。

```
<imart type="head">
 <link rel="stylesheet" href="sample.css" type="text/css" />
 <style type="text/css">
  .sample {
    background-color: black;
  }
 </style>
 <script type="text/javascript" src="sample.js"></script>
 <script type="text/javascript">
  function doSomething() {
    var foo='foo';
    someFunction(foo);
  }
 </script>
</imart>
```

コラム

テーマおよびUIコンポーネントは、jQueryを利用しています。
prototype.js を利用したい場合、<imart type=”head”>タグの中で prototype.js を読み込んでください。
UIコンポーネントは、(function($){...})(jQuery) のように無名関数として実装しているので、$関数の競合は発生しません。

head タグにスクリプトを記述したい

head タグに画面固有のスクリプトやCSSを記述したい場合は、<imart type=”head”></imart> タグの中に記述します。 下記の HTML 実装例を参考にしてください。

```
<!-- head タグ-->
<imart type="head">
 <title>画面名</title>
 <script>
  function hoge(){
   doSomething();
  }
 </script>
</imart>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>画面タイトル</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
  <ul class="imui-list-toolbar">
   <!-- 戻る -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
  <!-- ツールバー右側 -->
  <ul class="imui-list-toolbar-utility">
   <li><a href="#" class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a>
</li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container">
 <!--コンテンツエリア-->
</div>
```

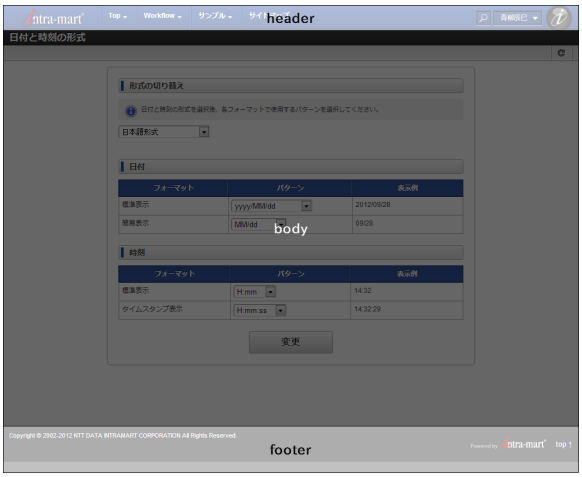
body タグの onload 属性にJavaScriptを記述したい

body タグの onload 属性に JavaScript を記述したい場合、jQuery の機能を利用して実行するようにしてください。

```
jQuery(document).ready(function() {
 doSomething();
});
```

body 部分

本章では、 body 部分の作成について説明します。



画面を構成する 4 つのパーツ で説明したとおり、 body 部分には以下を配置します。

- 画面タイトル
- ツールバー
- コンテンツエリア

これらは、 intra-mart Accel Platform で提供する*UIモジュール* を使って配置してください。 従来よりも効率よく画面開発を行えます。

本章では、全体で利用する UI モジュールを説明のあと、画面に配置する順番に説明します。

UIモジュール

UIモジュールは、PC 向けの画面を効率的に作成するためのモジュールです。 開発者はUIモジュールを利用することで、効率的にかつ統一したデザインの PC 向け画面開発を行うことができます。 intra-mart Accel Platform におけるUIモジュールは、以下を示します。

- jQuery UI
  jQuery UI は、jQuery のプラグインとして稼働する PC 用ライブラリです。
  jQuery UI を利用すると、開発者はインタフェースを意識しなくても、PC 用に最適化された Web 画面を作ることが可能になります。
  UI モジュールをインストールすると jQuery UI を利用する環境が整いますので、開発者は改めて jQuery UI をインストールする必要はありません。
  jQuery UI の詳細については下記サイトを参照してください。
  http://jqueryui.com/
  http://jqueryui.com/demos/
- UIコンポーネント
  jQuery UI を拡張したプラグインを利用した入力部品です。
  スクリプト開発向けタグライブラリのPC版UIコンポーネント を参照してください。
- CSSモジュール
  intra-mart Accel Platform 全体で統一されたデザインの CSS を提供します。
  詳細は、CSS Module List を参照してください。
  ツールバーや、表、見出しなどを用意しています。
- CSS Sprite
  アイコンを CSS Sprite として提供します。
  詳細は、CSS Sprite Image List の PC 向け を参照してください。

画面タイトル

画面タイトルを指定しましょう。 H1 タグを使用します。

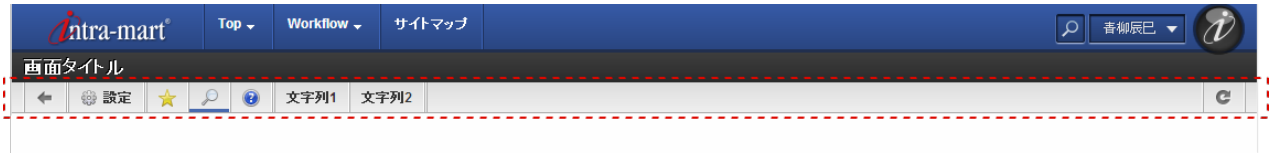
画面タイトルの HTML ソース

```
<div class="imui-title">
  <h1>画面タイトル</h1>
</div>
```

> **コラム**
> CSS Module List の見出しでも HTML ソースを中心に情報公開を行っています。 h1～h6の見出しを用意しています。

ツールバー

ツールバーは、画面を操作する処理リンクを配置します。 本章では、画面タイトル下に配置するツールバーについて説明します。



項目

処理リンクや表示切替など表示画面に対して操作を行う場合、ツールバーを利用しましょう。

以下 2 点を説明します。

1. ツールバーの構成
2. 処理リンクの配置方法

> **注意**
> 全機能でツールバーを必須ではありません。配置するアイコンが1つもない場合は、不要です。

> **注意**
> intra-mart Accel Platform では、アイコンを CSS Sprite としてを用意しています。 CSS Spriteは、*UIモジュール* の 1 つです。 処理リンクや処理アイコンとして、利用してください。 アイコンリストは、 CSS Sprite Image List の PC 向け を参照してください。

> **コラム**
> CSS Module List のツールバーでも HTML ソースを中心に情報公開を行っています。 記述レベルは本書が詳細にわたっています。

ツールバーの構成

ツールバーは、複数の div で構成されます。 処理リンク、処理アイコンは、ツールバー左右いずれかに配置します。

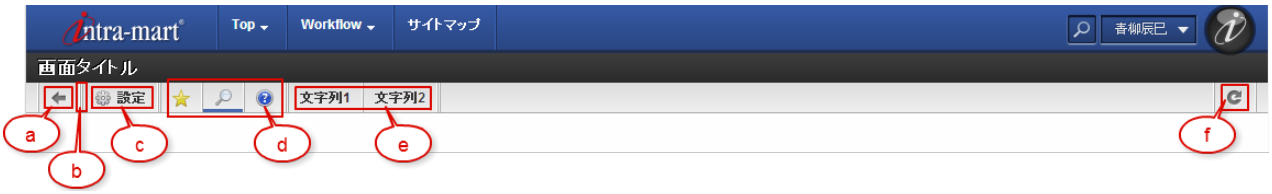
ツールバーの基本構成は、以下のとおりです。

- HTML ソース

```
<div class="imui-toolbar-wrap">[1]
 <div class="imui-toolbar-inner">[2]
  <ul class="imui-list-toolbar">[3]
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a>
</li>[4-1]
  </ul>
  <ul class="imui-list-toolbar-utility">[5]
   <li><a href="#" class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span>
</a></li>[4-2]
  </ul>
 </div>
</div>
```

	Class名	配置先	備考
1	imui-toolbar-wrap	外枠	–
2	imui-toolbar-inner	内枠	[1] の中に配置
3	imui-list-toolbar	ツールバー左側	[2] の中に配置
4	imui-list-toolbar-icon	処理リンクまたは処理アイコン	[3] 、 [5]の中に配置。[4-1] [4-2] に CSS Sprite のクラスを指定し、アイコン表示。
5	imui-list-toolbar-utility	ツールバー右側	[2] の中に配置

ツールバー実装例



配置内容

	名称	場所	構成	用途
a	戻る	左側の最左	アイコン＋ツールチップ	前のページへ戻る
b	区切り線	任意	区切り線	区切り線として利用
c	処理リンク	左側の中	アイコン＋文字リンク	設定の処理リンクに利用
d	タブアイコン	左側の中	アイコン＋ツールチップ	タブ切替に利用
e	文字列	左側の中	文字列	文字列表示に利用
f	最新表示	右側の最右	アイコン＋ツールチップ（最新表示）	再表示

HTMLコーディング

上記を HTML コーディングすると以下のようになります。

```
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
  <ul class="imui-list-toolbar">
   <!-- 戻るアイコン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- 設定リンク -->
   <li><a href="#" class="imui-toolbar-icon"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- お気に入りタブアイコン -->
   <li><a href="#" class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
   <!-- 検索タブアイコン -->
   <li><a href="#" class="imui-toolbar-tab"><span class="im-ui-icon-common-16-search"></span></a></li>
   <!-- ヘルプタブアイコン -->
   <li><a href="#" class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- 文字列 -->
   <li class="imui-toolbar-text-only">文字列1</li>
   <!-- 文字列 -->
   <li><span class="imui-toolbar-text-only">文字列2</span></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
  </ul>
  <!-- ツールバー右側 -->
  <ul class="imui-list-toolbar-utility">
   <!-- 最新表示アイコン -->
   <li><a href="#" class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a>
</li>
  </ul>
 </div>
</div>
```

指定内容まとめ

上記の指定内容をまとめると以下のようになります。

	名称	の class	<a>の class	<a>の title	の class	表示されるアイコン
a	戻る	–	imui-toolbar-icon	戻る	im-ui-icon-common-16-back	
b	区切り線	icon-split	–	–	–	–
c	処理リンク	–	imui-toolbar-icon	–	im-ui-icon-common-16-settings mr-5 [1]	
b	区切り線	icon-split	–	–	–	–
d	タブアイコン	–	imui-toolbar-tab	各機能で指定	im-ui-icon-common-16-star im-ui-icon-common-16-search im-ui-icon-common-16-question	
b	区切り線	icon-split	–	–	–	–
f	文字列	imui-toolbar-text-only [2]	–	–	imui-toolbar-text-only [3]	–
b	区切り線	icon-split	–	–	–	–
e	最新表示	–	imui-toolbar-icon	最新表示	im-ui-icon-common-16-refresh	

[1] アイコン＋文字リンクの処理リンクは、アイコンと文字リンクの間隔が狭くなってしまうため mr-5 クラスを指定します。

[2] liタグ、spanタグどちらかに imui-toolbar-text-only クラスを指定します。

[3] liタグ、spanタグどちらかに imui-toolbar-text-only クラスを指定します。

処理リンク／処理アイコン配置方法まとめ

処理リンク／処理アイコンの配置の表現は以下の 6 通りあります。

- *1 アイコン*
- *2 アイコン+文字リンク*
- *3 文字リンク*
- *4 タブアイコン*
- *5 文字列*
- *6 区切り線*

> **注意**
> - ツールバーは、解像度 1024 x 768 で表示した際に 1 行で収まるようにしてください。
> - ダイアログの場合指定したウィンドウサイズで 1 行で収まるようにしてください。
> - 表示したい処理リンクが 1 行で収まらない場合は、以下の対応をしてください。
>
> 1. 文字数を再考慮し、収まるようにしてください。
> 2. アイコンのみの表示とし、文字はツールチップで表示してください。

> **注意**
> 本章では、ツールバーに配置する場合の説明です。 汎用的な使用方法は、*処理リンク／処理アイコン* を参照してください。

1 アイコン

以下のとおり HTML コーディングを行います。

```
1  <li><a href="AA" class="imui-toolbar-icon" title="BB"><span class="CC"></span></a></li>
```

AA	遷移先を指定
BB	ツールチップの表示内容
CC	CSS Sprite の class 指定

> **注意**
> アイコンのみの表示の場合、ツールチップを指定しましょう。

2 アイコン+文字リンク

以下のとおり HTML コーディングを行います。

```
1  <li><a href="AA" class="imui-toolbar-icon"><span class="BB mr-5"></span>CC</a></li>
```

AA	遷移先を指定
BB	CSS Sprite の class 指定
CC	文字リンクの表示内容

3 文字リンク

以下のとおり HTML コーディングを行います。

```
1  <li><a href="AA" class="imui-toolbar-icon">BB</a></li>
```

AA	遷移先を指定
BB	文字リンクの表示内容

4 タブアイコン

以下のとおり HTML コーディングを行います。

```
<li><a href="AA" class="imui-toolbar-tab" title="BB"><span class="CC"></span></a></li>
```

AA	遷移先を指定
BB	ツールチップの表示内容
CC	CSS Sprite の class 指定

注意

タブアイコンは、以下2つの class を用意しています。

- 未選択時用 .imui-toolbar-tab
  マウスホバーすると、アイコン下に薄いテーマカラーの下線が入ります。
- active時用 .imui-list-toolbar-tab-selected
  アイコン下に濃いテーマカラーの下線が入ります。
- 使用方法
  intra-mart Accel Platform では、 class の切り替えを行いません。
  開発者にて、クリックしたアイコンの a タグの class 属性に .imui-list-toolbar-tab-selected が addClass されるようプログラミングを行ってください。

5 文字列

以下のとおり HTML コーディングを行います。 記述方法は2通りあります。

```
<li class="imui-toolbar-text-only">AA</li>
```

```
<li><span class="imui-toolbar-text-only">AA</span></li>
```

AA	文字列を指定

6 区切り線

以下のとおり HTML コーディングを行います。

```
<li class="icon-split"></li>
```

コンテンツエリア

コンテンツエリアには、入力フォームや一覧表などのコンテンツを配置します。

基本ルールは以下のとおりです。

- コンテンツエリア内で、コンテンツの塊ごとに div で囲みます。
- コンテンツの内容に応じて、見出しを配置します。
- コンテンツの内容に応じて、構造化を意識して要素を配置します。
- コンテンツ配置に *UI モジュール* を利用します。

コンテンツエリアは、以下の順番に説明します。

見出し

見出し <h1>は、画面タイトルやダイアログウィンドウのタイトルで使用しています。
コンテンツエリアでは、<h2>～<h6>を使って、見出しをつけましょう。
intra-mart Accel Platform で提供する以下の見出し<h2>～<h6>は、CSS Module List の「見出し」を参照してください。

1. <h2>見出しレベル 2 </h2>
2. <h3>見出しレベル 3 </h3>
3. <h4>見出しレベル 4 </h4>
4. <h5>見出しレベル 5 </h5>
5. <h6>見出しレベル 6 </h6>

テーブル

intra-mart Accel Platform では、UI モジュールとして、テーブルを用意しています。 下記以外の表を含め、詳細は、別ドキュメントのCSS Module List の「テーブル」を参照してください。

入力フォームのテーブル

入力や選択項目がある場合は、入力フォーム用テーブル「table.imui-form」を使用します。
詳細は、別ドキュメントの CSS Module List の入力フォーム用テーブル「table.imui-form」 を参照してください。


```
<table class="imui-form">
  <tr>
    <th><label>項目1</label></th>
    <td><input type="text" name="item1" /></td>
  </tr>
  <tr>
    <th><label>項目2</label></th>
    <td><input type="text" name="item2" /></td>
  </tr>
</table>
```

項目名となる th の中には直接文字列を指定せず label の中に文字列を指定します。

一覧テーブル

一覧画面などで使用するテーブルは、imuiListTable を使います。
imuiListTable は、ページャーのや表示件数を表示、カラムのソートなどの機能を備えています。
詳細は、別ドキュメントの PC版UIコンポーネント imuiListTable を参照してください。

検索条件／詳細検索

検索条件の指定を行う場合は、検索条件用テーブル「table.imui-form-search-condition」を使用します。
詳細は、別ドキュメントの CSS Module Listの検索条件用テーブル「table.imui-form-search-condition」 を参照してください。

項目1	
項目2	

```
<table class="imui-form-search-condition">
  <tr>
    <th><label>項目1</label></th>
    <td><input type="text" name="item1" /></td>
  </tr>
  <tr>
    <th><label>項目2</label></th>
    <td><input type="text" name="item2" /></td>
  </tr>
</table>
```

項目名となる th の中には直接文字列を指定せず label の中に文字列を指定します。

セル内の位置／文字寄せ（align）

横並びの表

横並びの表は、 1 行目：項目、 2 行目以降：データ、入力フォーム部品、処理アイコン、処理リンクとなります。
文字の寄せは、以下のとおりとします。

種類	文字寄せ
項目（行）	左寄せ
データ：数値	右寄せ
データ：文字列	左寄せ
データ：処理アイコン	中央寄せ
データ：処理リンク	中央寄せ
データ：日付／日時	左寄せ
データ：区分／コード	中央寄せ
チェックボックス	中央寄せ
ラジオボタン	中央寄せ
アイコン（状態表示など）	中央寄せ

縦並びの表

縦並びの表は、 1 列目：項目、 2 列目：データ、入力フォーム部品、処理アイコン、処理リンクとなります。
文字の寄せは、以下のとおりとします。

種類	文字寄せ
項目（列）	左寄せ
データ：数値	右寄せ [1]
データ：文字列	左寄せ
データ：処理アイコン	左寄せ
データ：処理リンク	左寄せ
データ：日付／日時	左寄せ
データ：区分／コード	左寄せ
チェックボックス	左寄せ
ラジオボタン	左寄せ

種類	文字寄せ
アイコン（状態表示など）	左寄せ

文字寄せの CSS クラス名

文字寄せを指定するには、以下の CSS クラスを指定してください。

文字寄せ	class
左寄せ	不要 [2]
中央寄せ	align-C
右寄せ	align-R

[1] ただし、数値とその他の入力フォーム・ラベルなどが左右に極端に離れてしまう場合、数値の入力フォーム幅を小さくするなどし、視線の移動が少なくなるように注意してください。

[2] CSSの継承により、左寄せにならない場合、align-L を指定してください。

コラム

align-C 、 align-R 、 align-L は、それぞれ UIコンポーネントのCSSモジュールです。
詳細は、以下を参照してください。

- CSS Module List の文字寄せ（左）「.align-L」
- CSS Module List の文字寄せ（中央）「.align-C」
- CSS Module List の文字寄せ（右）「.align-R」

サイズ指定方法

テーブルのサイズ指定の方法は以下とします。

HTML で table を記述する場合（スクロール無）

1. table
    a. width 指定不要です。（スタイルシートに指定があるため）
    b. imui-form-container で囲みます。
    または、Bootstrap を使用します。
2. th
    a. width はスタイルシートで準備されたクラスを使用します。wd-15、wd-20、wd-225px、wd-335px があります。
    ※wd-15 は、width:15% !important;、wd-20は、width:20% !important;が指定されます。
    または、Bootstrap を使用します。
3. td
    a. 基本的にサイズを指定しません。

コラム

Bootstrap について
Bootstrap の詳細は、別サイト「 Bootstrap 」を参照してください。
テーブル全体で Bootstrap を使用する場合は、該当のクラスを指定します。

HTML で table を記述する場合（スクロール有）

2 つのテーブルを上下または左右に配置してスクロールを表示させるため、td の width の幅を指定する必要があります。
px を指定しないと線のずれが発生するため、style=”width: 0px” を指定してください。

コラム

HTML5 の廃止タグ

- th 要素は、HTML5 では abbr 属性、align 属性、axis 属性、char 属性、charoff 属性、height 属性、nowrap 属性、valign 属性、width 属性 が廃止されています。
- td 要素は、HTML5 では abbr 属性、align 属性、axis 属性、char 属性、charoff 属性、height 属性、nowrap 属性、scope 属性、valign 属性、width 属性が廃止されています。

imuiListTable を使わない場合のソート順

テーブルのソートについて説明します。
以下は、基本ルールとなります。画面のユーザビリティが低下する場合は、各画面にて仕様を決定してください。
また、画面仕様により、機能単位で決定する場合は、画面間の差異が発生しないよう注意しましょう。

- 一覧テーブルに、複数列存在する場合は、ソートキーを第２ソートキーまで指定しましょう。
- 初期表示のソートキーは、画面表示の基準となります。従って、以下を実行時に初期表示と同じソートとなります。
  - 検索ボタンを押下した場合
  - 最新表示アイコンを押下した場合
- ソートの順番は、昇順から降順になります。
  - 未ソート例を押下した場合、昇順になります。

コラム

imuiListTableのソートについては、別ドキュメントの PC版UIコンポーネント imuiListTable を参照してください。

ソートキー指定例

例１：ソートを実行した場合、直前の第１ソートキーが第２ソートキーになります。

1	初期表示	第１ソートキー：ユーザ名昇順	第２ソートキー：ユーザコード昇順
2	住所をクリック	第１ソートキー：住所昇順	第２ソートキー：ユーザ名昇順
3	電話番号をクリック	第１ソートキー：電話番号昇順	第２ソートキー：住所昇順

例２：第Xソートキーを固定キーにします。（以下更新日時は、一覧表には非表示）

1	初期表示	第１ソートキー：ユーザ名昇順	第２ソートキー：更新日時降順
2	住所をクリック	第１ソートキー：住所昇順	第２ソートキー：更新日時降順
3	電話番号をクリック	第１ソートキー：電話番号昇順	第２ソートキー：更新日時降順

入力フォーム

本章では、入力フォームの部品（テキストボックスやセレクトボックスなど）について説明します。

項目

入力フォーム全体

以下を推奨します。

登録／更新／参照画面／一覧画面は、以下のコンテナいずれかで囲みます。

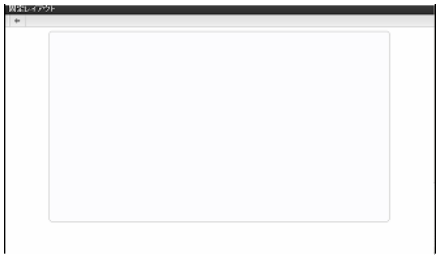
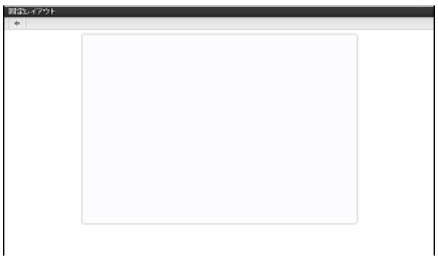
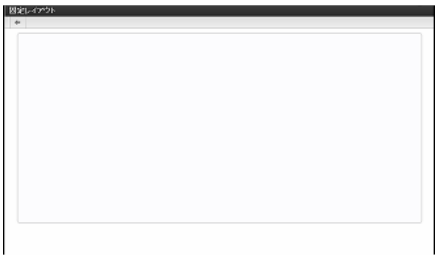
- imui-form-container
- imui-form-container-narrow
- imui-form-container-wide

HTML コーディング 実装例

```
<div class="imui-form-container">
  ここに表や入力フォームなどが配置されます
</div>
```

コンテナの違い

- 指定した class により枠の幅 (%) が変わります。（サンプル画像では style=height:300px を指定）

imui-form-container	imui-form-container-narrow	imui-form-container-wide
width:75%	width:60%	width:90%
		

> **コラム**
>
> imui-form-container 、 imui-form-container-narrow 、 imui-form-container-wide は、*UIモジュール* の 1 つです。
> 別ドキュメントの CSS Module List の「コンテナ」で説明しています。

入力フォーム部品

本章で説明する部品一覧は以下の通りです。

名称	imart タグ	生成される HTML タグ	placeholder 属性 [1]
テキストボックス	imuiTextbox	<input type=”text” />	○
パスワード	imuiPassword	<input type=”password” />	○
テキストエリア	imuiTextArea	<textarea></textarea>	○
チェックボックス	imuiCheckbox	<input type=”checkbox” />	−
ラジオボタン	imuiRadio	<input type=”radio” />	−
セレクトボックス	imuiSelect	<select></select>	− [2]

[1] placeholder は、*入力ヒント（ placeholder ）の表示* を参照してください。

[2] list の1番目に、入力ヒント（ placeholder ）を体言止めで記述します。（例：ロケールを選択）

> **注意**
>
> 上記以外に、imuiMultiDragbox や imuiRichtextbox などの入力フォーム部品も用意しています。
> スクリプト開発向けタグライブラリのPC版UIコンポーネント を参照してください。

入力フォーム部品の基本ルール

入力フォーム部品の基本的な記述方法と基本ルールを説明します。 詳細は、スクリプト開発向けタグライブラリのPC版UIコンポーネント を参照してください。

テキストボックス（imuiTextbox）、パスワード（imuiPassword）

- 外観と基本的な HTML ソース
  - imuiTextbox

テキストボックス

```
<imart type="imuiTextbox" value="テキストボックス" />
```

- imuiPassword

••••••••

```
<imart type="imuiPassword" value="password" />
```

> **注意**
> 詳細は、スクリプト開発向けタグライブラリのPC版UIコンポーネント の imuiTextbox 、 imuiPassword を参照してください。

- 共通ルール
  - autofocus 属性は、任意指定です。
    画面表示時に1番最初に入力してほしい部品に指定します。
    例：ログイン画面で、ユーザコードのテキストボックスに autofocus を指定します。
    　　検索画面で、検索文字列のテキストボックスに autofocus を指定します。
  - 横幅の指定方法は以下の通りです。
    style=”width:000px;” で指定します。（000 は該当のサイズを指定してください）
    size 属性は使用しません。
    ※size 属性は、ブラウザによる表示の差異が発生します。（表示フォントにも依存します）
  - maxlength 属性を指定します。
    （一般ユーザのユーザビリティ向上の為に指定します）
  - class を指定せずに、自動で角丸デザインが適用されます。
  - 入力不可の場合（１）
    readonly 属性を指定します。
    例：フローティングカレンダーからテキストボックスに入力します。（直接編集は不可です）
  - 入力不可の場合（２）
    readonly 属性を指定し、class=”imui-text-readonly” を指定します。
    （class=”imui-text-readonly” は線なし、背景なしになります）
    例：登録画面で入力可能だった項目を参照画面、編集画面で表示します。
  - 入力不可の場合（３）
    disabled 属性を指定します。
    例：ラジオボタンの選択値により、入力制御をします。

> **コラム**
> 上記の「入力不可」は、 imuiTextbox 、 imuiPassword 、 input type=”text” 、 input type=”password” をそのまま利用する方法です。
> これ以外に、label タグと input type=”hidden” を組み合わせて入力不可にする方法もあります。
> どちらの方法を採用しても問題ありませんが、画面単位での統一を図るようにしてください。

テキストエリア（imuiTextArea）

- 外観と基本的な HTML ソース

テキストエリア

```
<imart type="imuiTextArea" value="テキストエリア" />
```

> **注意**
> 詳細は、スクリプト開発向けタグライブラリのPC版UIコンポーネント の imuiTextArea を参照してください。

- 共通ルール
  - 縦幅、横幅の指定方法
    style=”width: 000px; height: 000px;” で指定。（000 は該当のサイズを指定してください）

cols 属性、rows 属性は使用しません。（テキストボックスと合わせます）

- class を指定せずに、自動で角丸デザインが適用されます。
- 入力不可の場合
  テキストボックスに準拠します。
  （class=”imui-text-readonly” は線なし、背景なしになります）

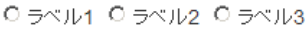
チェックボックス（imuiCheckbox）、ラジオボタン（imuiRadio）

- 外観と基本的な HTML ソース
  - imuiCheckbox

    ```
    1  <imart type="imuiCheckbox" label="ラベル" />
    ```

  - imuiRadio

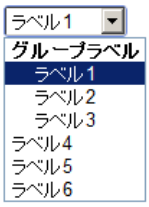
    ```
    1  <imart type="imuiRadio" name="radio" label="ラベル1" />
    2  <imart type="imuiRadio" name="radio" label="ラベル2" />
    3  <imart type="imuiRadio" name="radio" label="ラベル3" />
    ```

    > **注意**
    > 詳細は、スクリプト開発向けタグライブラリのPC版UIコンポーネント の imuiCheckbox 、 imuiRadio を参照してください。

- 共通ルール
  - 配置する際に、スペースや HTML タグによる空白の調整は不要です。
    CSS で margin: 5px 5px 0px 0px; を指定しています。
  - 選択不可の場合
    テキストボックスに準拠します。

セレクトボックス（imuiSelect）

- 外観と基本的なソース

  - サーバサイド JavaScript

```
var sampleList = [
  {
    type: "group",
    label: "グループラベル",
    list: [
      {
        label: "ラベル1",
        value: "value1"
      },
      {
        label: "ラベル2",
        value: "value2"
      },
      {
        label: "ラベル3",
        value: "value3"
      }
    ]
  },
  {
    label: "ラベル4",
    value: "value4"
  },
  {
    label: "ラベル5",
    value: "value5"
  },
  {
    label: "ラベル6",
    value: "value6"
  }
];
```

- HTML ソース

```
<imart type="imuiSelect" list=sampleList />
```

> 注意
> 詳細は、スクリプト開発向けタグライブラリのPC版UIコンポーネント の imuiSelect を参照してください。

- ルール
  - 入力ヒントを記述しましょう。
    - 表示方法1: optgroup（type: ‘group’）を使用し、リストの 1 行目に表示します。
      体言止めにしてください。（例：ロケールを選択）
      ※imuiSelect をクリックし、リストを表示すると optgroup が表示されます。
    - 表示方法2: imuiSelect の右側または近くにラベルで表示します。
      基本的に体言止めとします。（例：ロケールを選択）
      ユーザビリティを考慮し、敬体表示も可とします。（例：ロケールを選択してください）
      ※項目名で完結している場合は、不要です。
  - リストを生成する際のデータはソート順を指定して取得してください。

入力ヒント（ placeholder ）の表示

テキストボックスやテキストエリアの 入力ヒント（ placeholder ）について説明します。

- ソース例

```
<imart type="imuiTextbox" placeholder="ユーザ氏名、ユーザカナを入力してください。" />
```

- 画面

- ルール
  - placeholder （プレースホルダー）は、 テキストボックス、テキストエリアの入力欄に初期値として表示される文字列です。
    入力ヒントや操作ヒントとして使用します。
  - placeholder が非表示の状態で、入力内容がわからない場合は、項目名の表示や、ラベルでヒントを明示してください。
    placeholder の未対応のブラウザもあります。
  - width を placeholder に合わせる必要はありません。入力桁数や画面デザイン（全体のテキストボックスの横幅）を考慮してください。
  - 入力項目にラベルが無い、画面の構成上ラベルが付けられない場合のヒント
    - 入力OK：「ユーザコード、ユーザ名を入力してください。」
  - 入力項目のラベルを見ても入力値が想像しづらい場合のヒント
    - 文章の場合は、敬体（です/ます）を推奨します。
    - 入力OK：「画面上に表示される名前です。」「スペース区切りで単語を複数指定できます。」
  - ユーザに入力フォーマットを指示する場合
    - 入力OK：「000-0000」「0000000（ハイフン不要）」「2000/10/10」「YYYY/MM/DD（例: 「2012/05/04」）」
    - 入力NG：「YYYY-mm-dd」（エンドユーザが利用する画面では、専門用語は避けてください）

> **コラム**
>
> 「ｘｘして下さい（実質動詞）」ではなく、「ｘｘしてください（補助動詞）」と平仮名で記述します。

Tab キーによる移動順序（ tabindex ）

Tab キーによるフォーカスの移動順序を tabindex 属性について説明します。

- 基本的には、tabindex の指定は不要です。（Tab キーで移動は、左上から右下に流れるため）
- ただし、画面の構造上、フォーカス移動の順番を指定したい場合は、tabindex を指定してください。
- tab キーによる移動で、フォーカスをあてたくない部品は、tabindex=”-1” を指定してください。

文字寄せ（ align ）

- 対象：テキストボックス、テキストエリア
- 文字の寄せは、以下のとおりとします。

種類	文字寄せ
数値	右寄せ
文字列	左寄せ
日付／日時	左寄せ
区分／コード	左寄せ

- 文字寄せを指定するには、以下の CSS クラスを指定してください。

文字寄せ	class
左寄せ	不要 [3]
中央寄せ	align-C
右寄せ	align-R

[3] CSSの継承により、左寄せにならない場合、align-L を指定してください。

コラム

align-C 、 align-R 、 align-L は、それぞれ UIコンポーネントのCSSモジュールです。
詳細は、以下を参照してください。

- CSS Module List の文字寄せ（左）「.align-L」
- CSS Module List の文字寄せ（中央）「.align-C」
- CSS Module List の文字寄せ（右）「.align-R」

必須入力の表記方法

入力項目が、必須入力かどうか判別できるようにしましょう。
以下のように HTML に記述します。

- HTML

```
<table class="imui-form">
 <tbody>
  <tr>
   <th><label class="imui-required">必須項目</label></th>
   <td><input type="text" name="item1"></td>
  </tr>
  <tr>
   <th><label>項目</label></th>
   <td><input type="text" name="item2"></td>
  </tr>
 </tbody>
</table>
```

画面上には、「必須項目＊」と表示されます。

コラム

intra-mart Accel Platform では、 CSS クラスを用意しています。
以下が有効になります。

```
.imui-required: after {
  color: #e00;
  content: " *";
}
```

コラム

CSS Module List の必須入力記号「label.imui-required」 にて同じ情報を公開しています。

ボタン

本章では、ボタンの配置、サイズについて説明します。

項目

imuiButton と CSS クラスでボタン作成

intra-mart Accel Platform では、UIコンポーネントとして imuiButton を用意しています。
また、 CSSモジュールとして ボタンの複数サイズ用意しています。

本章では基本的なボタンの記述方法を説明します。

詳細は、 以下を参照してください。

- スクリプト開発向けタグライブラリのPC版UIコンポーネント の imuiButton
- CSS Module List の「ボタン」

- HTML 実装例

```
<imart type="imuiButton" value="ボタン" class="imui-medium-button" />
```

画面上には以下のように表示されます。

ボタンのサイズ種類と CSS クラス

ボタンのサイズは 4 種類用意しています。
以下にボタンの種類と CSS の クラス を下記にまとめます。

ボタンの種類	class	画面上の表示
大	imui-large-button	
中	imui-medium-button	
小	imui-small-button	
通常	imui-button	

> **コラム**
>
> ボタンは、 imuiButton の使用を推奨します。
> しかし、 やむをえず HTML タグの input タグ 、button タグ を使用する場合、上記の CSS クラスを使用してください。
> intra-mart Accel Platform の統一されたデザインのボタンを配置できます。

> **コラム**
>
> CSS クラスの詳細は、 CSS Module List の「ボタン」を参照してください。

ボタン配置

- 配置する順番は左から重要度、頻度などが高いものから配置します。

登録画面、編集画面
処理ボタン（登録／更新／削除ボタン）

- ボタンサイズ
  ボタンサイズの基本は「**大**」とします。
- ボタン配置
  - 画面下部の中央配置とします。
    CSS Module List のボタン配置エリア「.imui-operation-parts」 と組み合わせると、簡単に配置できます。
    下記の HTML ソースを参照してください。

```
<div class="imui-operation-parts">
  <imart type="imuiButton" value="登録" class="imui-large-button" />
</div>
```

```
<div class="imui-operation-parts">
 <imart type="imuiButton" value="更新" class="imui-large-button" />
 <imart type="imuiButton" value="削除" class="imui-large-button" />
</div>
```

- 連続登録が可能な登録画面は、左から **登録ボタン、連続登録ボタン** とします。
- 編集画面の配置順番は、左から **更新ボタン、削除ボタン** とします。

入力補助呼出し

本章は、ボタンについての説明ですが、本項に関しては関連する文字リンクなどもあわせて説明します。

- 入力補助画面を呼出し方法
  ボタン、文字リンク、アイコン、アイコン＋文字リンクいずれかを使用します。
- ボタンの場合

  - ボタンサイズの基本は「 **通常** 」とします。
  - 配置はテキストボックス等入力フォーム部品右隣とします。
- 文字リンクの場合
  配置はテキストボックス等入力フォーム部品右隣とします。
- アイコンの場合
  - 配置はテキストボックス等入力フォーム部品右隣とします。
  - アイコンのみの処理ボタンの場合、**title 属性を必ず指定** します。
    （ユーザビリティ向上のため。画像のみに意味を持たせないようにしてください。）
- アイコン＋文字リンクの場合
  - 配置はテキストボックス等入力フォーム部品右隣とします。
  - 左からアイコン、文字リンクの順に並べます。

ダイアログ

- 処理ボタン
  - ボタンサイズ
    ボタンサイズの基本は「 **中** 」とします。
  - ボタン配置
    配置は **右下配置** とします。
- 入力補助呼出し
  登録画面に従います。

検索画面（検索条件エリア）

- 簡単検索
  - ボタンサイズ
    ボタンサイズの基本は「 **通常** 」とします。
  - ボタン配置
    配置はテキストボックス等入力フォーム部品右隣とします。
- 詳細検索
  - ボタンサイズ
    ボタンサイズの基本は「 **中** 」とします。
  - ボタン配置
    配置は **右下配置** とします。

処理リンク／処理アイコン

本章では、処理リンク／処理アイコンについて説明します。
ツールバー の *処理リンク／処理アイコン配置方法まとめ* ではツールバーに関してのみ記述しています。
本章では、汎用的に使用する処理リンク／処理アイコンを説明します。

項目

配置ルール

- 処理アイコン／処理リンクは、一覧テーブルの処理アイコン群、一覧テーブルのセル、入力フォーム等に使用します。
- 配置の優先度は、左から重要度、頻度などが高いものから順番に配置します。
- 処理アイコン／処理リンクの周りは、誤ってクリックしないよう空白を設けましょう。
- 反対の意味の処理アイコン／処理リンクを配置する場合
  - 処理アイコン／処理リンクのサイズを十分とってください。
    処理アイコン／処理リンクの周りに空白をおくようにします。
    16px x 16px のアイコンは小さいため、ユーザによりわかりづらかったり、誤った操作が発生することがあります。
    よって、「アイコン＋文字リンク」を推奨します。
  - 処理アイコン／処理リンクの隙間を 10px 以上開けてください。

処理アイコンの表現方法と実装例

処理リンク／処理アイコンの配置の表現は以下の 3 通りあります。

	表現方法
1	アイコン
2	アイコン＋文字リンク
3	文字リンク

以下は 処理リンク／処理アイコンを 1 つ並べた場合と、2 つ並べた場合の実装例です。

ml-5、ml-10 は空白を設けるための調整用です。必要に応じて使用します。
複数配置する場合や、一覧表の左上に配置する場合は、CSS Module List の操作リストエリア「.imui-operation-list」 の中に <ul> を使って記述します。

1 アイコン

以下のとおり HTML コーディングを行います。

- 1 つの場合

```
<a href="AA" title="BB"><span class="CC"></span></a>
```

- 2 つの場合

```
<div class="imui-operation-list">
 <ul>
  <li><a href="AA" class="ml-5" title="BB"><span class="CC"></span></a></li>
  <li><a href="AA" class="ml-10" title="BB"><span class="CC"></span></a></li>
 </ul>
</div>
```

AA	遷移先を指定
BB	ツールチップの表示内容
CC	CSS Sprite の class 指定

> **注意**
> アイコンのみの表示の場合、ツールチップを指定しましょう。 国によって、記号やジェスチャの考え方が異なるため、誤解を招かないための対策としてください。

2 アイコン＋文字リンク

以下のとおり HTML コーディングを行います。

- 1 つの場合

```
<a href="AA" class="mr-5"><span class="BB mr-5"></span>CC</a>
```

- 2 つの場合

```
<div class="imui-operation-list">
 <ul>
  <li><a href="AA" class="ml-5"><span class="BB mr-5"></span>CC</a></li>
  <li><a href="AA" class="ml-10"><span class="BB mr-5"></span>CC</a></li>
 </ul>
</div>
```

AA	遷移先を指定
BB	CSS Sprite の class 指定
CC	文字リンクの表示内容

3 文字リンク

以下のとおり HTML コーディングを行います。

- 1 つの場合

```
<a href="AA">BB</a>
```

- 2 つの場合

```
<div class="imui-operation-list">
  <ul>
    <li><a href="AA" class="ml-5">BB</a></li>
    <li><a href="AA" class="ml-10">BB</a></li>
  </ul>
</div>
```

AA	遷移先を指定
BB	文字リンクの表示内容

文字リンクの CSS クラス

intra-mart Accel Platform は、文字リンクを 3 種類用意しています。
通常、a タグを指定すると、自動で青字（hover で＋下線）となります。
よって、通常の文字リンクにおいては CSS クラス の指定は不要になります。
ただし、表現箇所により黒字・青字をあえて指定する場合は、下記を指定してください。
（例：メニューでグラデーションがあるので文字色不要 ⇒ class=”imui-unaccented” を指定してください）

種類	class 指定なし	class=”imui-accent”	class=”imui-unaccented”
デフォルト	青字	青字	黒字
訪問済	青字	青字	黒字
未訪問	青字	青字	黒字
マウスオーバ	淡青字＋下線	淡青字＋下線	濃灰色字＋下線
アクティブ	淡青字＋下線	淡青字＋下線	濃灰色字＋下線

コラム

ツールバーでは、文字リンクは黒字になります。
青字にしたい場合は、a タグにclass=”imui-accent”を指定します。

```
<a href="AA" class="imui-accent">BB</a>
```

以下の外部ドキュメントにて、情報公開しています。

- CSS Module List の文字リンク（標準色）「a.imui-accent」
- CSS Module List の文字リンク（黒文字）「a.imui-unaccented」

エンターキー押下時のフォームの動作

本章では、サブミットについて説明します。

項目
- form のサブミット方法
- エンターキー押下時にサブミットさせる

form のサブミット方法

form のサブミットを行うにはいくつかの方法がありますが、 intra-mart Accel Platform では、以下の方法でサブミットします。

- imuiButton または、 input type=”button” を配置します。
- click イベントで form.submit() を実行します。
- imuiAjaxSend を実行します。
- imuiAjaxSubmit を実行します。

input type=”submit” はお勧めしません。 理由は以下の通りです。

- エンターキー押下でサブミットされる
- imuiConfirm を呼び出そうとしても、サブミットが先に実行されてしまい、確認ダイアログが表示されない

もし input type=”submit” を利用する際には以下のとおり、onsubmit 属性で return false; を記述してください。

```
<form onsubmit="return false;">
  ...
</form>
```

エンターキー押下時にサブミットさせる

検索画面などでエンターキー押下時にサブミットさせたいときは、input type=”submit” を利用しましょう。 ただし「form のサブミット方法」で注意書きした通り、imuiConfirm を呼び出そうとしても確認ダイアログが表示されないことに注意してください。

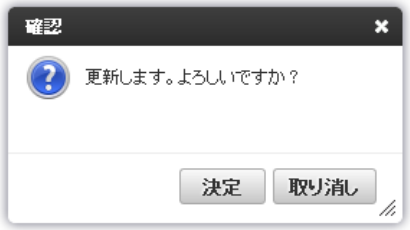
```
<form>
  <input type="submit"/>
</form>
```

アラートとコンファーム

本章は、アラートとコンファームについて説明します。

項目
- 実装方法
- imuiAlert 、 imuiConfirm と messageDialog の違い
- ボタン配置
- ダイアログのタイトル
- 表示アイコン
- メッセージルール

- コンファームの画面例



> **注意**
> intra-mart Accel Platform ではデザイン統一のために基本的に window.alert と window.confirm を利用しません。

実装方法

- window.alert() にあたる警告ダイアログは、 クライアントサイド JavaScript の imuiAlert を使用します。
- window.confirm() にあたる確認ダイアログは、 クライアントサイド JavaScript の imuiConfirm を使用します。
- その他は、 クライアントサイド JavaScript の imuiMessageDialog を使用します。

imuiAlert 、 imuiConfirm と messageDialog の違い

imuiAlert 、 imuiConfirm と messageDialog の違いは、以下の通りです。

UIコンポーネント名	imuiAlert	imuiConfirm	messageDialog
アイコンの変更	不可	不可	可能
ボタンのラベル変更	不可	不可	可能
ボタンの数変更	不可	不可	可能
ボタンの数	一つ	二つ	任意

- imuiAlert は、警告メッセージを表示したい時に使用します。
  「決定」ボタンを表示します。
- imuiConfirm は、 確認メッセージを行いたい時に使用します。
  「決定」ボタンと「取り消し」ボタンを表示します。
- messageDialog は、上記の UI コンポーネントで表現できないメッセージを表示する時に使用します。

ボタン配置

- imuiAlert、imuiConfirm の場合
  ボタンの配置は変更できません。
- messageDialog の場合
  「**決定**」「**取り消し**」のボタンを配置します。
  ただし、画面の内容に対し、ボタン名が合わない場合は、「入力」「確定」「登録」など変更します。
  ボタン名は、機能内で統一します。

ダイアログのタイトル

「入力エラー」「登録確認」のように **体言止め** で設定します。

表示アイコン

- imuiAlert、imuiConfirm の場合
  以下のアイコンが表示されます。変更はできません。

表示アイコン	UIコンポーネント名	メッセージ例
	imuiAlert	「削除対象のＸＸＸを選択してください。」
	imuiConfirm	「登録します。よろしいですか。」

> コラム
>
> 詳細は、 クライアントサイド JavaScript の以下を参照してください。
>
> - imuiAlert
> - imuiConfirm

- messageDialog の場合

アイコンは、CSS Sprite の class を指定します。
アイコンとメッセージの関連付けは、以下で統一します。

表示アイコン	意味	メッセージ例
	処理を続行できない失敗、エラーなど	「処理に失敗しました。」「登録できませんでした。」
	ユーザ操作により処理を続行できる警告など	「削除対象のＸＸＸを選択してください。」
	正常、成功など、処理が正常に終了	「処理に成功しました。」「登録しました。」
	確認など、ユーザ操作の続行を確認	「登録します。よろしいですか。」

- CSS Sprite の 指定方法

span タグの class 属性に CSS Sprite の class 名を指定します。

```
<span class="CSS Spriteのclass名"></span>
```

intra-mart Accel Platform が提供するアイコンリストは、 CSS Sprite Image List の PC 向け を参照してください。
アイコンサイズは、16 、 24 、 32pxを用意しています。
画面によりサイズを選んでください。
メッセージ性を伝えるために、 32px 、または、 24px をおすすめします。
以下に 表示アイコンに対して、 該当の CSS Sprite の class 名（ 24px ）をまとめました。

表示アイコン	CSS Sprite の class名
	im-ui-icon-common-24-error
	im-ui-icon-common-24-warning
	im-ui-icon-common-24-confirmation
	im-ui-icon-common-24-question

> コラム
> imuiMessageDialog の詳細は、 クライアントサイド JavaScript の imuiMessageDialog を参照してください。

メッセージルール

メッセージの文章は、敬体「です。」「ます。」で文章を記載します。

画面遷移

本章では、画面遷移について説明します。

項目

作成ルール

以下は intra-mart Accel Platform の画面遷移の基本ルールです。

- intra-mart Accel Platform は参照画面を用意しません。
  データの参照は、更新画面で行います。
- 画面遷移図の 💬 は、処理確認の **コンファーム** です。
  トランザクションが発生する場所では、表示することを推奨します。
  ただし、連続登録をする場合など、ユーザビリティが低下する箇所は、コンファーム不要とします。
- 新規登録画面／更新画面の呼び出しは、画面遷移／ダイアログウィンドウどちらでも可能です。
  ただし、各要件での統一をしてください。

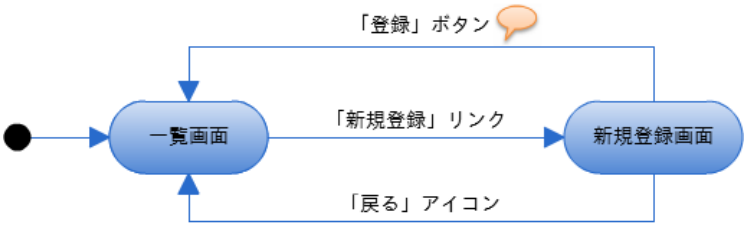
画面遷移パターン（基本編）

画面遷移で、基本的なものを画面種類ごとに説明します。

> **コラム**
>
> 画面遷移図の 💬 は、処理確認の **コンファーム** です。
> コンファームの詳細は、 *アラートとコンファーム* を参照してください。

新規登録画面

- 新規登録画面から画面遷移するパターンは以下のようになります。



配置内容	配置場所
「新規登録」リンク	ツールバー
「戻る」アイコン	ツールバー
「登録」ボタン	画面下部中央

更新画面

- 更新画面画面から画面遷移するパターンは以下のようになります。

配置内容	配置場所
「編集」アイコン	一覧表の編集列
文字リンク	一覧表の任意列
「戻る」アイコン	ツールバー
「更新」ボタン	画面下部中央

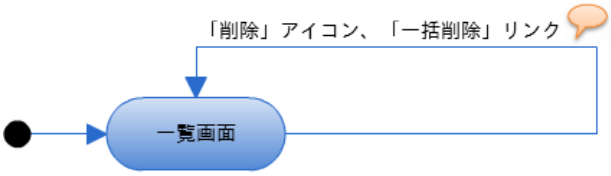
削除画面

- 編集画面と削除画面でそれぞれの画面遷移するパターンは以下のようになります。

編集画面で削除

配置内容	配置場所
「編集」アイコン	一覧表の編集列
文字リンク	一覧表の任意列
「戻る」アイコン	ツールバー
「削除」ボタン	画面下部中央

一覧画面で削除



配置内容	配置場所
「削除」アイコン	一覧表の削除列
「一括削除」リンク	一覧表の左上

注意事項

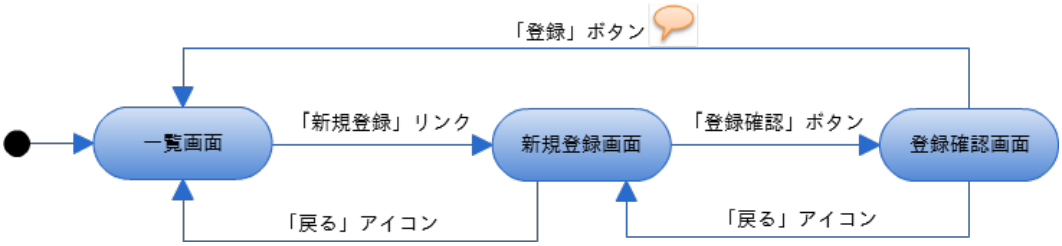
- 削除方法は、ユーザビリティ考慮してどちらか一方でも双方配置も可能です。
  ただし、利用ユーザの単位や、機能のまとまりを考慮してください。
  マスタ画面Ａでは編集画面で削除、マスタ画面Ｂでは一覧で削除だとユーザに違和感を与えます。

画面遷移パターン（応用編）

- 業務システムの利用方法を考慮した場合、下記確認画面を挟む画面遷移を選択することも考慮します。
- 確認画面を挟む事で、一度ユーザに入力内容を考えさせることも可能です。

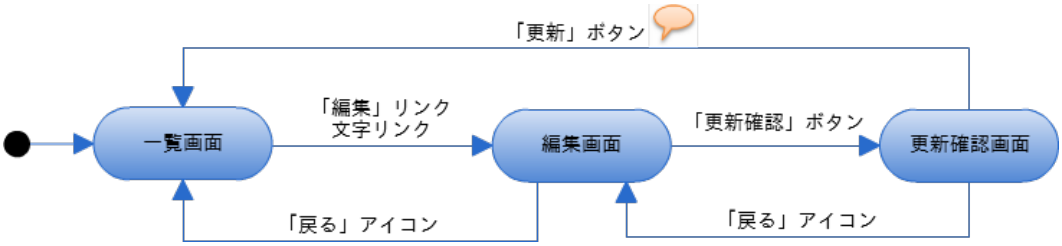
新規登録画面（確認画面有）

- 確認ダイアログがある場合の新規登録画面からの画面遷移は以下のようになります。



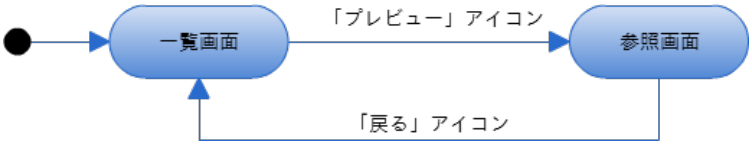
編集画面（確認画面有）

- 確認ダイアログがある場合の編集画面からの画面遷移は以下のようになります。



参照画面（例外）

intra-mart Accel Platform の基本ルールでは、参照画面を用意しません。
しかし、各機能内で用意する時は、下記遷移方法とします。



画面遷移データ保持

画面遷移時に、データ保持は次画面まで保持とします。
詳細は以下のとおりとします。

検索条件

- 一覧画面から遷移した画面で「戻る」クリック時、保持した状態を表示します。
- 登録/更新/削除ボタンクリック後、一覧画面が表示された時、初期表示とします。

ページ番号、ページ件数、ソート順

- 一覧画面から遷移した画面で「戻る」クリック時、保持した状態を表示します。
- 登録/更新/削除ボタンクリック後、一覧画面が表示された時、初期表示とします。
- ページャの操作時は、ページ件数、ソート順を保持します。

登録画面／編集画面

- 確認画面を設けた場合、確認画面から戻った時、保持した状態を表示します。
- 登録/更新/削除ボタンクリック後、一覧画面が表示された時、初期表示とします。

例外的処置

- ユーザビリティが低下する場合は、データ保持をしてください。
- 例：一覧で「searchItem」の入ったデータを一括削除したい。
  - ✖ searchItem 検索→ページ上全選択→削除を繰り返します。
  - ○ searchItem 検索の後、ページ上全選択→削除を繰り返します。

エラー処理

バリデーション

jQuery Validation を利用したクライアント側でのバリデーションをかけることができます。

バリデーションをかける場合、通常はクライアント側とサーバ側の両方にバリデーションを実装します。
Jssp Validator と連携すると、サーバ側に設定ファイルを書くだけでサーバ側とクライアント側の
両方に共通なバリデーションを行います。

なお、クライアント側だけで完結させることも可能です。
この場合は、jQuery Validation の記述方法に則ったルール、メッセージをクライアント側で実装してください。

> **注意**
> Jssp Validatorはスクリプト開発モデルのみで利用できます。

項目

前提

バリデーションは jQuery Validation Plugin を利用します。ライブラリは自動的に読み込まれません。
バリデーションを利用する際は以下のような実装を行ってください。

```
<imart type="head">
  <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
</imart>
```

JSSP Validation と連携する場合

Jssp Validator の設定ファイルをクライアント側に読み込み、jQuery Validation Plugin の設定に変換することで、サーバ側とクライアント側とでバリデーションのルールを共有できます。

共有するには、バリデーション設定ファイル の記述、imuiValidationRule タグの記述、imuiValidate 関数の呼び出しが必要です。

> **注意**
> JSSP Validationに独自に追加したバリデーションルールは、クライアント側で共有できません。
> クライアント側で同じバリデーションを行いたい場合は、クライアント側でも同様のルールを定義してください。
> クライアント側のカスタムバリデーションについては クライアントサイド JavaScript の imuiAddValidationRule を参照してください。

実装例

以下のような HTML を例に取り、 user_cd に対して user_cd バリデーションをかけます。

```
<div class="imui-form-container">
  <form id="sampleform" name="sampleform">
    <input type="text" name="user_cd"/>
    <input type="button" id="validate-button" value="validate"/>
  </form>
</div>
```

バリデーション設定ファイルの記述

項目 user_cd に対して userCd バリデーションをかける、というバリデーション設定ファイルを作成します。
バリデーション設定ファイルとして以下のようなファイルを WEB-INF/jssp/src/validator/sample.js として保存します。
設定ファイルの内容の詳細は バリデーションルール を参照してください。

```
var init = {
   'user_cd': {
      caption: 'user_cd',
      userCd: true
   }
}
```

> **コラム**
> JSSP Validator の「 file 」「 mimeType 」ルールは クライアントサイド JavaScript バリデーション（ imuiValidate ）では動作しませんので注意してください。

imuiValidationRule タグの記述

上記で作成したバリデーション設定ファイルを読み込むために、imuiValidationRule タグを HTML に追記します。
imuiValidationRule タグの rule 属性に、バリデーション設定ファイルのパスを指定します。
また、後述の imuiValidate 関数の引数となる rulesName, messagesName 属性も指定します。

```
<imart type="head">
   <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
   <imart type="imuiValidationRule" rule="validator/sample#init" rulesName="rules" messagesName="messages"></imart>
</imart>
<div class="imui-form-container">
   <form id="sampleform" name="sampleform">
      <input type="text" name="user_cd"/>
      <input type="button" id="validate-button" value="validate"/>
   </form>
</div>
```

> **コラム**
> imuiValidationRule タグは script タグを出力するので、script タグの内部に書いてはいけません。

imuiValidation 関数の記述

バリデーションを実行するため、imuiValidate 関数の呼び出しを追記します。
imuiValidate 関数の引数に、form の id 、 imuiValidationRule タグの rulesName 、 messagesName に指定した値の３つを指定します。

```
<imart type="head">
   <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
   <imart type="imuiValidationRule" rule="validator/sample#init" rulesName="rules" messagesName="messages"></imart>
   <script type="text/javascript">
   jQuery(document).ready(function() {
      jQuery('#validate-button').click(function() {
         imuiValidate('#sampleform', rules, messages);
      });
   });
   </script>
</imart>
<div class="imui-form-container">
   <form id="sampleform" name="sampleform">
      <input type="text" name="user_cd"/>
      <input type="button" id="validate-button" value="validate"/>
   </form>
</div>
```

テキストボックスにひらがなを入力し、validate ボタンをクリックするとエラーが表示されます。

クライアント側だけで完結させる場合

クライアント側だけで完結させる場合、jQuery Validation Plugin のルール、メッセージの指定の仕方に則って実装します。
使用できるバリデーションルールは、バリデーション設定ファイル と同様です。

実装例

バリデーションのルールと、バリデーションの結果、チェックにかかった場合のメッセージを指定します。

指定しない場合、メッセージは英語で表示されます。

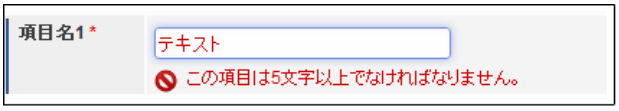
```
var rules = {
  user_cd: {
    required: true,
    minlength: 5,
  }
};

var messages = {
  user_cd: {
    required:  '必須です',
    minlength: '少なくとも5文字必要です'
   }
};
imuiValidate(
  '#account',                 // 送信するフォームのID
  rules,                      // バリデーションルール
  messages                    // バリデーションメッセージ
);
```

上記のように指定した場合、user_cd のバリデーションルールに該当した際に、同じキーのメッセージが表示されます。

必須チェックに該当した場合

最小長さチェックに該当した場合



バリデーションをリセットする方法

- imuiValidate を実行すると、対象フォームに対し常にチェックをかけるようになるため、
  一度投稿処理をしてからでも、画面を再読込みしない限り常にチェックがかかります。
- この動作を変える必要がある場合、imuiResetForm メソッドでチェックをかけた form を初期化してください。

```
...
// バリデーションチェック
if (imuiValidate('#form', rules, messages)) {
  // バリデーションチェックに成功したら投稿処理
  doSomething();
  // バリデーションのリセット
  imuiResetForm('#form');
}
...
```

汎用メッセージ画面

アプリケーション内で発生したエラーによっては、回復不能な場合があります。
このような場合に表示する画面と、その画面に遷移するための API を提供します。

API

表示できるメッセージ表示画面の種類、その画面に遷移するための API は以下のとおりです。

- エラー
  - Transfer.toErrorPage
- 警告
  - Transfer.toWarningPage
- インフォメーション
  - Transfer.toInformationPage

メッセージ表示画面に表示できるのは、以下 3 つです。

- タイトル
- メッセージ
- 詳細なエラーメッセージ

また、メッセージ表示画面表示後の遷移先を指定することも可能です。
遷移先 URL、遷移先 URL のラベルを指定すると、画面遷移のボタンが表示されます。
遷移先 URL へ引き渡すパラメータを指定すると、そのパラメータが遷移先に引き渡されます。

メッセージ表示画面へはリダイレクトを利用して遷移します。

実装例

エラー画面へ遷移するための例を示します。
パラメータの詳細は、API リファレンスを参照してください。

```
function error(request) {
 Transfer.toErrorPage({
   title: 'タイトル',
   message: 'メッセージ',
   detail: ['詳細メッセージ1', '詳細メッセージ2'],
   returnUrl: '/login', // 戻り先 URL
   returnUrlLabel: 'ログイン画面へ戻る',
   parameter: {
     key: 'value',
     list: ['1','2','3']
   }
 });
}
```

コラム
上記パラメータのメッセージは多言語対応していません。多言語対応したい場合は、MessageManager から取得したメッセージをセットしてください。

コラム
テーマを適用させたくない場合には、API を呼び出す前に request.setAttribute(‘imui-theme-builder-module’, ‘notheme’) を実行し、テーマを適用しないようにしてください。

汎用エラー画面表示後の戻り先画面

汎用メッセージ画面 の仕組みで表示したエラー画面の戻り先を説明します。

注意
本章は、基本ルールです。個別の機能の事情で従うのが無理な場合は従う必要はありません。
ただし、機能内での統一性は担保してください。

汎用エラー画面の「戻る」ボタンを表示する場合

（例）「メニュー」⇒「一覧画面」⇒「登録/更新画面」の様な画面遷移

- 「登録/更新画面」でエラーが発生し、汎用エラー画面に遷移した場合
- エラーが発生した画面の遷移元に戻る。（「一覧画面」）
- ※ 遷移元が複数階層ある場合は、（可能であれば）エラーが発生した画面に一番近い遷移元に遷移します。

エラー画面の「戻る」ボタンを表示しない場合

（例）「メニュー」⇒「一覧画面」⇒「登録/更新画面」の様な画面遷移

- 「一覧画面」でエラーが発生し、汎用エラー画面に遷移した場合。
- 「戻る」ボタンを表示しません。

Ajax のエラー処理

Ajax の通信先で回復不能なエラーが発生した場合、エラー画面へ遷移したい場合があります。
ここではこのような場合にエラー画面へ遷移するための仕組みと実装方法を説明します。

項目

エラー画面へ遷移するための仕組み

サーバ側で例外が発生すると httpXXX.jsp（http500.jsp, http401.jsp など）が呼び出されます。

Ajax のリクエストの場合、httpXXX.jsp のレスポンスは Ajax 内で閉じるため画面にエラーが表示されません。
Ajax のリクエストでもエラー画面に遷移するため、以下のような仕組みを実装しています。

1. Ajax のリクエストを送る際に、HTTP Header へ x-jp-co-intra-mart-ajax-request-from-imui-form-util = true をセットする。
2. サーバで例外が発生した際、httpXXX.jsp は HTTP Header へ x-jp-co-intra-mart-ajax-request-to-imui-form-util = true をセットし、エラー情報を JSON としてレスポンスを返す。
3. $.ajax の error: で、x-jp-co-intra-mart-ajax-request-to-imui-form-util が HTTP Header に存在する場合、エラー情報をパースし、エラー情報を Form にセットして元の画面（httpXXX.jsp）へサブミットする。
4. 画面にサーバで発生した例外が表示される。

実装方法

jQuery.ajax の headers, error に以下のように指定します。

```
$.ajax({
  ...,
  headers: { 'x-jp-co-intra-mart-ajax-request-from-imui-form-util': 'true' },
  error: imuiTransitionToErrorPage,
  ...
});
```

このように実装することで、サーバで例外が発生した際にエラー画面へ遷移させることができます。

独自のエラー処理をしながらエラー画面へ遷移したい

上記サンプルでは、エラー画面へ遷移するだけで他の処理はしません。
独自の処理を追加したい場合は以下のように実装することができます。

```
$.ajax({
  ...,
  headers: { 'x-jp-co-intra-mart-ajax-request-from-imui-form-util': 'true' },
  error: function(XMLHttpRequest, textStatus, errorThrown) {
    //
    // 独自の処理を実行
    //

    imuiTransitionToErrorPage(jqXHR, textStatus, errorThrown);
  },
  ...
});
```

国際化情報入力項目

本章では、国際化入力について、記載します。

項目

国際化情報の入力・登録・更新について

ここでは入力・登録・更新する場合の国際化情報についてを説明します。
intra-mart Accel Platform でもこの処理例を利用しています。

入力項目の表示に関して

国際化入力項目の仕様

- 国際化入力項目が必須入力の場合、**標準表示名** とシステムにインストールしているロケール分の入力項目を用意します。

  例： 日本語ロケール、英語ロケールに対応する場合
  　標準表示名、日本語、英語の 3 つの入力項目が表示されます。

- 項目の表示順
  - 標準表示名、テナントのデフォルトロケール、その他のロケールの順番に表示します。
  - テナントのデフォルトロケール以外の並び順は、 SystemLocale#getLocaleInfos() が返した順になります。
  - テナントのデフォルトロケールが設定されていない場合は、システムデフォルトのロケールになります。
- **標準表示名** を必須項目とします。
- 各ロケールは、任意入力項目とします。
- 各ロケールが未入力の場合、標準表示名が、有効になります。

入力項目が、「全て必須」 かつ 「項目数が少ない」場合

- 標準表示名とシステムにインストールしているロケール分、入力項目をすべて表示します。
- デフォルトロケール以外の言語ロケールを、あらかじめ隠しておくなどの処理は、一切しません。

入力項目が、「一部必須」、または 「項目数が多い」 場合

標準表示名のみ表示し、サーバ側のロジックで各ロケールに何を登録するか決定します。

コラム
国際化情報の対応を行う場合は、表示する文字列をプロパティファイルで管理します。 入力項目と同様に、標準表示名とロケール毎に管理します。 詳細は、 スクリプト開発モデル プログラミングガイド の 多言語対応 を参照してください。

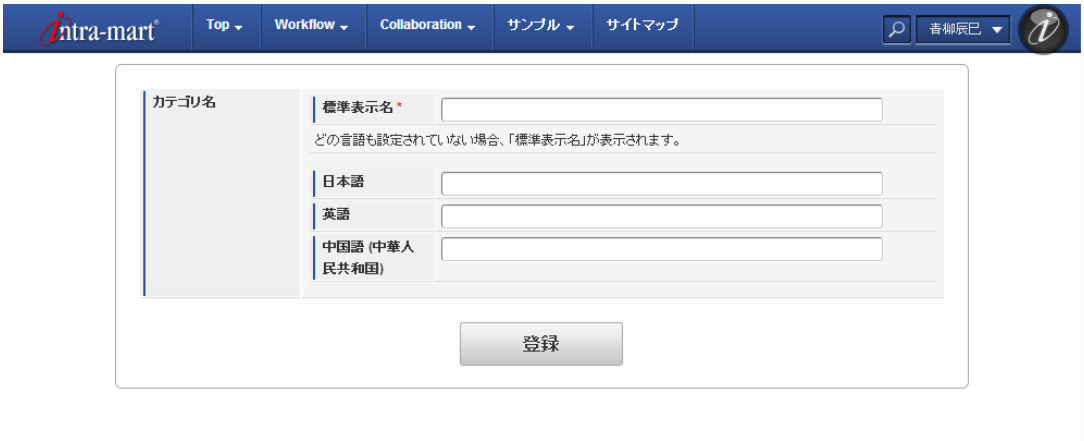
画面からの入力、および、登録・更新処理に関して

- 国際化入力項目が必須入力の場合、 **標準表示名は必須入力** 、 **各ロケールは任意入力** とします。
  任意入力の項目に何も入力されなかった場合は、標準表示名の項目値と同じ値が入力されたものとみなします。
- 登録・更新時、任意項目が未入力の場合、標準表示名より該当する項目の内容をコピーします。
- コピー処理は、サーバサイド JavaScript 側で行います。
  API 内部でコピーは、行ってはなりません。
- 任意入力の国際化入力項目値を補完は、サーバサイド JavaScript 側で行います。

バリデーションチェック

- 国際化入力項目等、動的に変化する項目に関しては、 クライアントサイド JavaScript とサーバサイド JavaScript で独自チェックします。
  それ以外の固定的な項目は、通常どおり JS Validation を利用したチェックをします。

実装例



validator.js （ サーバサイド JavaScript ）

- バリデーションルールを記述します。
- 国際化入力項目は required = true に設定しておきます。

```
var init = {
   dataId : {
      caption: MessageManager.getMessage("CAP.UI.DEFAULT.LABEL"),
      required: true,
      maxlength: 256
   },
   dataName : {
      caption: MessageManager.getMessage("CAP.UI.FACILITY.CATEGORY.NAME"),
      maxlength: 256
   }
};
```

page.js（サーバサイド JavaScript ）

テナントデフォルト言語、または、システムデフォルト言語が上に来るように調整します。

```
var $result = {};

function init(request) {
    getLocaleInfo($result);
}

function getLocaleInfo(obj) {
    var tenantInfoManager = new TenantInfoManager();
    obj.defaultLocaleId = tenantInfoManager.getTenantInfo(true).data.locale;
    if (isBlank(obj.defaultLocaleId)) {
        obj.defaultLocaleId = SystemLocale.getDefaultLocaleInfo().data.locale;
    }
    obj.locales = [];
    var localeInfos = SystemLocale.getLocaleInfos().data;
    for (var i = 0; i < localeInfos.length; i++) {
        if (obj.defaultLocaleId == localeInfos[i].locale) obj.locales.push({
            id:localeInfos[i].locale,
            name:localeInfos[i].displayName
        });
    }
    for (var i = 0; i < localeInfos.length; i++) {
        if (obj.defaultLocaleId != localeInfos[i].locale) obj.locales.push({
            id:localeInfos[i].locale,
            name:localeInfos[i].displayName
        });
    }
}
```

page.html （クライアントサイド JavaScript、HTML）

バリデーションルール、メッセージの読み込みと動的バリデーションの設定をします。
サーバ側に国際化項目を送る場合は、ベースとなる値（デフォルト言語）と、全言語分を個別で送ります。
これにより、デフォルト言語の必須と長さを自動でチェックできます。

```
<imart type="head">
  <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
  <imart type="imuiValidationRule" rule="sample/validator#init" rulesName="validateRules" messagesName="validateMessages" />

  <script type="text/javascript">
    jQuery(function() {
      // 生成されたテキストボックスに name 属性を付与します
      jQuery("#input_form [name=dataNames] input").each(function() {
        var newName = "dataName_" + jQuery(this).attr("locale");
        var caption = jQuery(this).parent().prev().text();
        jQuery(this).attr("name", newName);
        // 付与した name 属性に対するバリデーションを追加します
        validateRules[newName] = {caption:caption,__caption__:caption,maxlength: 256};
      });

      jQuery("#submit-button").click(function() {
        // クライアントサイドのバリデーションチェックを行います
        if (imuiValidate("#input_form", validateRules, validateMessages)) {
          var names = {};
          jQuery("#input_form [name=dataNames] input").each(function() {
            names[jQuery(this).attr("locale")] = jQuery(this).val();
          });

          jQuery.ajax({
            type: "POST",
            url: "sample/ajax",
            dataType: "json",
            data: {
              dataId: jQuery("#input_form [name=dataId]").val(),
              dataName: jQuery("#input_form [name=dataNames] input:first").val(),
              dataNames: ImJson.toJSONString(names)
            },
            success: function(result) {
              if (result.error) {
                imuiShowErrorMessage(result.errorMessage, result.detailMessages);
                return;
              }  else  {
                imuiShowSuccessMessage(result.successMessage);
              }
            },
```

```
        error: function(request, textStatus, errorThrown) {
          imuiShowErrorMessage(request.statusText + "(" + request.status + ")", "");
        }
       });
      }
    });
  });
 </script>

</imart>

<form id="input_form" onsubmit="return false;" class="imui-form-container">
 <table class="imui-form">
  <tbody>
   <tr>
    <th class="wd-20"><label><imart type="message" id="CAP.UI.FACILITY.CATEGORY.NAME" escapeXml="true" escapeJs="false" />
</label></th>
    <td>
     <table class="imui-form" >
      <tbody>
       <tr>
        <th class="wd-20"><label class="imui-required" data-locale="<imart type="string" value=$bind.defaultLocaleId
escapeXml="true" escapeJs="false" />"><imart type="message" id="CAP.UI.DEFAULT.LABEL" escapeXml="true" escapeJs="false" />
</label></th>
        <td><imart type="imuiTextbox" name="dataId" style="width: 400px;"/></td>
       </tr>
       <tr>
        <td colspan="2"><imart type="message" id="CAP.UI.DESCRIPTION.LOCALE.SETTING" escapeXml="true" escapeJs="false" />
</td>
       </tr>
      </tbody>
     </table>
     <table class="imui-form" name="dataNames">
      <tbody>
       <imart type="repeat" list=$result.locales item="record">
        <tr>
         <th class="wd-20"><label><imart type="string" value=record.name /></label></th>
         <td><imart type="imuiTextbox" locale=record.id style="width: 400px;" /></td>
        </tr>
       </imart>
      </tbody>
     </table>
    </td>
   </tr>
  </tbody>
 </table>
 <div class="imui-operation-parts">
  <imart type="imuiButton" id="submit-button" value="%CAP.UI.ACT.REGISTRATION" class="imui-large-button" />
 </div>
</form>
```

ajax.js （サーバサイド JavaScript ）

```
/**
 * @param request
 * @validate sample/validator#init
 * @onerror handleErrors
 */
function init(request) {
    // 国際化項目のみ追加チェック（必須チェック済なので長さチェックのみ）
    var dataNames = isBlank(request.dataNames) ? {} : ImJson.parseJSON(request.dataNames);
    for (var localeId in dataNames) {
        var dataName = dataNames[localeId];
        var result = {};
            if (isString(dataName) && dataName.length > 256) {
            // バリデーションエラー
            result = {
                error: true,
                successMessage: "",
                errorMessage: MessageManager.getMessage('MSG.E.IWP.ASYNC.TASKQUE.ADD.SERIESQUEUE.INPUT'),
                detailMessages: MessageManager.getMessage("MSG.W.IWP.JSSP.VALIDATION.MAXLENGTH",
MessageManager.getMessage("CAP.UI.FACILITY.CATEGORY.NAME"), "256")
            };
            sendJSONString(result, result.errorMessage, result.detailMessages);
            return;
        }
    }
    result.error = false;
    sendJSONString(result, MessageManager.getMessage('CAP.UI.SUCCESS.MESSAGE'));
}

function handleErrors(request, validationErrors) {
    let result = {
        error: true,
        successMessage: '',
        errorMessage: MessageManager.getMessage('MSG.E.IWP.ASYNC.TASKQUE.ADD.SERIESQUEUE.INPUT'),
        detailMessages: validationErrors.getMessages()
    };
    sendJSONString(result, result.errorMessage, result.detailMessages);
}

function sendJSONString(result, message, detailMessages) {
    message = (message == null) ? '' : message;
    detailMessages = (detailMessages == null) ? '' : detailMessages;

    var response = Web.getHTTPResponse();
    response.setContentType('application/json; charset=utf-8');
    response.sendMessageBodyString(ImJson.toJSONString({
        error: result.error,
        successMessage: result.error ? '' : message,
        errorMessage: result.error ? message : '',
        detailMessages: result.error ? detailMessages : ''
    }));
}
```

HTML ／ CSSコーディング Tips

本章では、HTML/CSS コーディングの Tips を記載します。

項目

ブロック要素を横に並べたい、div を横に並べたい

対処方法

親要素に clearfix のクラス、子要素に float 用のクラスを指定してください。

実装例

HTML

```
<div class="imui-box-operation cf （1）">
 <div class="float-L （2）pl-10 （3）">
  <imart type="imuiTextbox"/>
  <imart type="imuiButton" value="検索"/>
  <imart type="imuiButton" value="クリア"/>
 </div>
</div>
```

（1）	cf	clearfix クラス	各ブラウザでブロック要素の回り込みを解除
（2）	float-L	float 用クラス	float:left が入ります。右に配置する場合は、float-R クラスを指定
（3）	pl-10	assist クラス	padding-left:10px が入る

文字リンクを青字／黒字にしたい

仕様

- 以下は、青字（マウスオーバー時、淡青字＋下線）になります。
  - a タグ class 指定なし
  - 以下のテーブル
    - imui-table-box
    - imui-table table
    - imui-table-calendar
    - imui-table-sort
    - imui-table-mixed

        - imui-table-inner
        - imui-form
        - imui-form-search-condition
    - 以下のテーブルの td
        - imui-table-sort
- 以下は、黒字（マウスオーバー時、濃灰色字＋下線）になります。
    - ツールバーの文字リンク

対処方法

- 黒字の文字リンクをあえて青字にしたい場合、<a class=”imui-accent”> を指定してください。
- 青字の文字リンクをあえて黒字にしたい場合、<a class=”imui-unaccented”> を指定してください

> **コラム**
>
> 以下の外部ドキュメントにて、情報公開しています。
>
> - CSS Module List の文字リンク（標準色）「a.imui-accent」
> - CSS Module List の文字リンク（黒文字）「a.imui-unaccented」

実装例

HTML

```
<a class="imui-accent">強制青字</a>

<a class="imui-unaccent">強制黒字</a>
```

オペレーションボックスにタイトルバー／ツールバーを配置したい

対処方法

CSS Module List のツールボックス「.imui-box-toolbox」 を用意しています。
以下に実装例を示します。詳細は、CSS Module List のツールボックス「.imui-box-toolbox」 を参照してください。

実装例

構造

HTML を簡単に表すと以下のようになります。

```
<div>ツールボックス</div>
  <div>タイトルバー</div>
  <div>ツールバー</div>
  <div>コンテンツ</div>
</div>
```

HTML

```
<div class="imui-box-toolbox （1） mt-20">
 <div class="imui-box-title （2） imui-box-toolbox-look （3）">
  <h3>これは、H3</h3>
 </div>
 <div class="imui-toolbar-wrap">
  <div class="imui-box-toolbar-inner">
   <ul class="imui-list-box-toolbar">
    （中略）
   </ul>
  </div>
 </div>
 <div class="imui-box-toolbox-content">
  <!-- 簡単検索 -->
  <imart type="imuiTextbox" class="wd-225px" placeholder="ユーザ氏名、ユーザカナを入力してください。" autofocus></imart>
  <imart type="imuiButton" value="検索" class="imui-button"></imart>
  <imart type="imuiButton" value="クリア" class="imui-button"></imart>
 </div>
</div>
```

追加したスタイル

1. ツールボックス（imui-box-operation と同じ枠線）
   imui-box-toolbox
2. タイトルバー（h2, h3は同じ見た目。imui-chapter-title.h2 使用時は、imui-box-title.h3 を使用）
   imui-box-title, imui-box-title.h2, imui-box-title.h3
3. タイトルバー／ツールバー上部角丸効果（（1）にフィットさせる）
   imui-box-toolbox-look
4. コンテンツ
   imui-box-toolbox-content
5. ツールバー
   imui-toolbar-wrap, imui-box-toolbar-inner, imui-list-box-toolbar

imuiDialog にツールバーを配置したい

imuiDialog に属性を用意しています。
以下に実装例を示します。詳細は、imuiDialog を参照してください。

対処方法

imuiDialog の toolbarLeft 属性、toolbarRight 属性をご利用ください。 詳細は、APIリスト imuiDialog をご確認ください。

テーマカラーを border 、背景色とし使用したい

テーマカラーを border 、背景色を指定する場合、色情報のみ CSS Module として提供しています。

以下の CSS をご利用ください。

テーマカラーを指定するプロパティ	指定するクラス名
border の色指定	imui-theme-border-color

同じ情報を CSS Module List のテーマカラー線色「.imui-theme-border-color」 にて公開しています。

対処方法

同じ HTML 要素に以下を指定します。

- class=”imui-theme-border-color”
- 線の位置と線の太さを指定します。（ class 属性、 style 属性どちらで指定も可）

実装例

left-border に 100px の実線を引く場合

左側に 100px の実線が表示されます。色は薄いテーマカラーです。

CSS

width と style を必ず入れてください。

```
.new_class {
  border-left-width: 100px;
  border-left-style: solid;

  ...
}
```

HTML

```
<div class="new_class imui-theme-border-color">左側に 100px の実線が表示されます。色は薄いテーマカラーです。</div>
```

> **コラム**
> 色を変更することはできません。

left-border に 2px の実線を引く場合

CSS

width と style を必ず入れてください。

```
.new_class {
  border-left-width: 2px;
  border-left-style: solid;

  ...
}
```

HTML

```
<div class="new_class imui-theme-border-color">左側に 2px の実線が表示されます。色は薄いテーマカラーです。</div>
```

> **コラム**
> 色を変更することはできません。

画面に「ページの説明文」を配置したい

対処方法

画面の操作内容や、凡例などを画面上に表示したい場合、「ページの説明文」を配置します。
以下にアイコンと組み合わせた実装例を1つ示します。
その他は、別ドキュメントの CSS Module List の補足ボックス「.imui-box-supplementation」 を参照してください。

実装例

アイコンを組み合わせることも可能です。
アイコンは任意の画像を指定します。

HTML

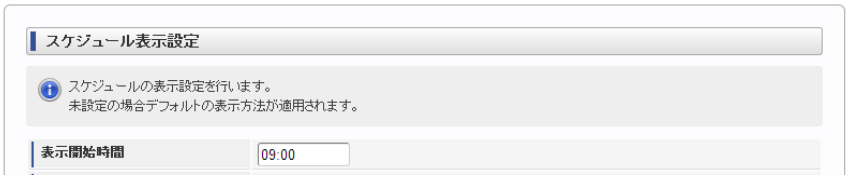
```
<div class="imui-box-supplementation">
   <span class="im-ui-icon-common-24-information float-L"></span>
   <p class="imui-pgh-section" style="padding-left:30px;">アイコンを組み合わせることも可能です。<br>アイコンは任意の画像を指定します。</p>
</div>
```

補足

- 実装例の表示アイコンは、 インフォメーション用です。
  その他のアイコンを使用したい場合は、 *アラートとコンファーム* の *表示アイコン* を参照してください。

表示アイコン	意味
	入力フォームの上部に情報を表示するインフォメーション

- 配置について
  セクションを全体説明する場合はセクション表題（ h3 など）の下に説明文を記載します。



基本的な画面の作り方 スクリプト開発編

登録画面

作成ガイド

登録画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

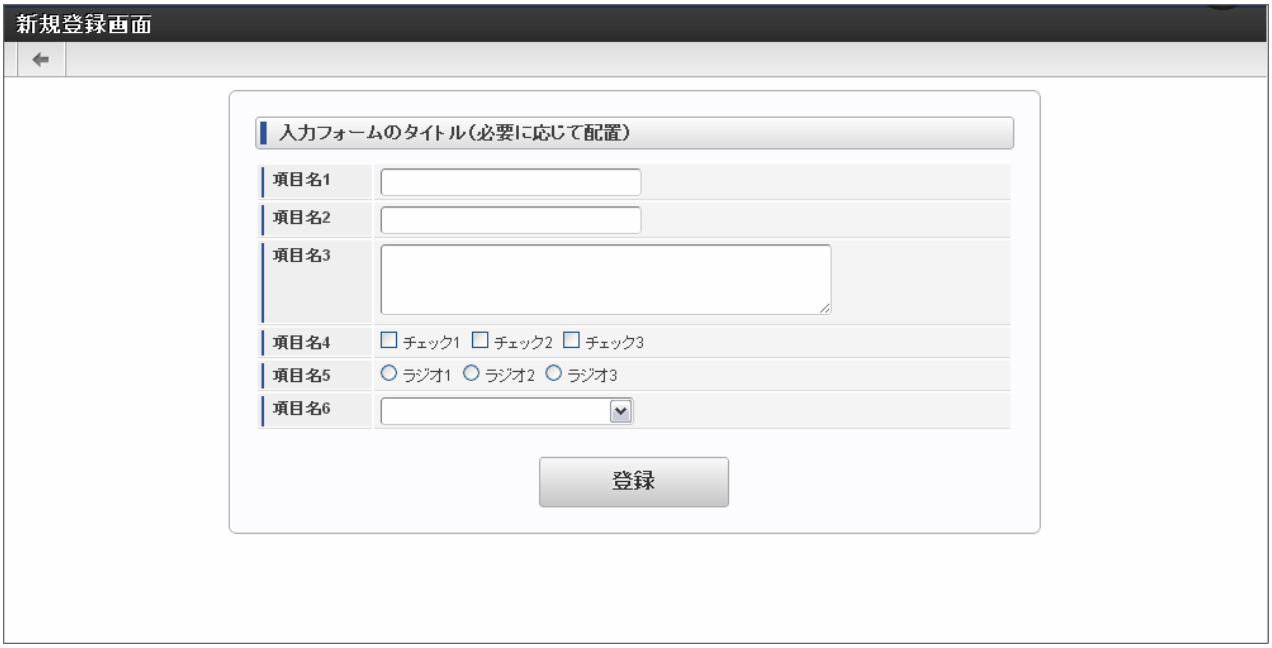
- *画面レイアウト*

補足事項

- 見出し（ h2 ～ h6 ）は必要に応じて配置します。必須部品ではありません。
- ボタンラベルは、基本的に「登録」とします。
  ユーザビリティを考慮して「作成」などにする変更する場合、関係画面の統一を図ってください。
- 同じデータの更新画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト



HTML / クライアントサイド JavaScript

```
<imart type="head">
 <title>新規登録画面</title>
       ・・・
</imart>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>新規登録画面</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <ul class="imui-list-toolbar">
   <!-- 戻るボタン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-narrow">
 <!-- h2 -->
 <div class="imui-chapter-title">
  <h2>入力フォームのタイトル（必要に応じて配置）</h2>
 </div>

 <form id="form" method="POST" class="target_form mt-10" action="hoge/register" onsubmit="return false;">
  <table class="imui-form">
   <tr>
    <th><label>項目名1</label></th>
    <td><imart type="imuiTextbox" id="textbox" name="textbox" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名2</label></th>
    <td><imart type="imuiPassword" id="password" name="password" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名3</label></th>
    <td><imart type="imuiTextArea" id="textarea" name="textarea" width="350px" height="50px" /></td>
   </tr>
   <tr>
    <th><label>項目名4</label></th>
    <td>
     <imart type="imuiCheckbox" id="checkbox_1" name="checkbox" label="チェック1" />
     <imart type="imuiCheckbox" id="checkbox_2" name="checkbox" label="チェック2" />
     <imart type="imuiCheckbox" id="checkbox_3" name="checkbox" label="チェック3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名5</label></th>
    <td>
     <imart type="imuiRadio" id="radio_1" name="radio" label="ラジオ1" />
     <imart type="imuiRadio" id="radio_2" name="radio" label="ラジオ2" />
     <imart type="imuiRadio" id="radio_3" name="radio" label="ラジオ3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名6</label></th>
    <td>
     <imart type="imuiSelect" id="inputItem6" name="inputItem6" width="200px" />
    </td>
   </tr>
  </table>
  <div class="imui-operation-parts">
   <imart type="imuiButton" value="登録" id="register-button" class="imui-large-button" />
  </div>
 </form>
</div>
```

更新画面

項目

作成ガイド

更新画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

補足事項

- 見出し（h2 ～ h6）は必要に応じて配置します。必須部品ではありません。
- 更新画面で、更新／削除を行えるようにする場合は、画面下部に左から更新ボタン、削除ボタンの順番に配置します。
  ボタンラベルは、基本的に「更新」「削除」とします。
  ユーザビリティを考慮して「保存」などにする場合、関係画面の統一を図ってください。
- 同じデータの登録画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト

HTML / クライアントサイド JavaScript

```
<imart type="head">
  <title>更新／削除画面</title>
        ・・・
 </imart>

 <!-- 画面タイトル -->
 <div class="imui-title">
  <h1>更新／削除画面</h1>
 </div>

 <!-- ツールバー -->
 <div class="imui-toolbar-wrap">
  <div class="imui-toolbar-inner">
   <ul class="imui-list-toolbar">
    <!-- 戻るボタン -->
    <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
   </ul>
  </div>
 </div>

 <!-- コンテンツエリア -->
 <div class="imui-form-container-narrow">
  <!-- h2 -->
  <div class="imui-chapter-title">
   <h2>入力フォームのタイトル（必要に応じて配置）</h2>
  </div>

  <form id="form" method="POST" class="target_form mt-10" action="hoge/update_delete" onsubmit="return false;">
   <table class="imui-form">
    <tr>
     <th><label>項目名1</label></th>
     <td><imart type="imuiTextbox" id="textbox" name="textbox" style="width: 200px;" /></td>
    </tr>
    <tr>
     <th><label>項目名2</label></th>
     <td><imart type="imuiPassword" id="password" name="password" style="width: 200px;" /></td>
    </tr>
    <tr>
     <th><label>項目名3</label></th>
     <td><imart type="imuiTextArea" id="textarea" name="textarea" width="350px" height="50px" /></td>
    </tr>
    <tr>
     <th><label>項目名4</label></th>
     <td>
      <imart type="imuiCheckbox" id="checkbox_1" name="checkbox" label="チェック1" />
      <imart type="imuiCheckbox" id="checkbox_2" name="checkbox" label="チェック2" />
      <imart type="imuiCheckbox" id="checkbox_3" name="checkbox" label="チェック3" />
     </td>
    </tr>
    <tr>
     <th><label>項目名5</label></th>
     <td>
      <imart type="imuiRadio" id="radio_1" name="radio" label="ラジオ1" />
      <imart type="imuiRadio" id="radio_2" name="radio" label="ラジオ2" />
      <imart type="imuiRadio" id="radio_3" name="radio" label="ラジオ3" />
     </td>
    </tr>
    <tr>
     <th><label>項目名6</label></th>
     <td>
      <imart type="imuiSelect" id="inputItem6" name="inputItem6" width="200px" />
     </td>
    </tr>
   </table>
   <div class="imui-operation-parts">
    <imart type="imuiButton"  value="更新" id="update-button" class="imui-large-button" />
    <imart type="imuiButton"  value="削除" id="delete-button" class="imui-large-button" />
   </div>
  </form>
 </div>
```

一覧画面

作成ガイド

一覧画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。

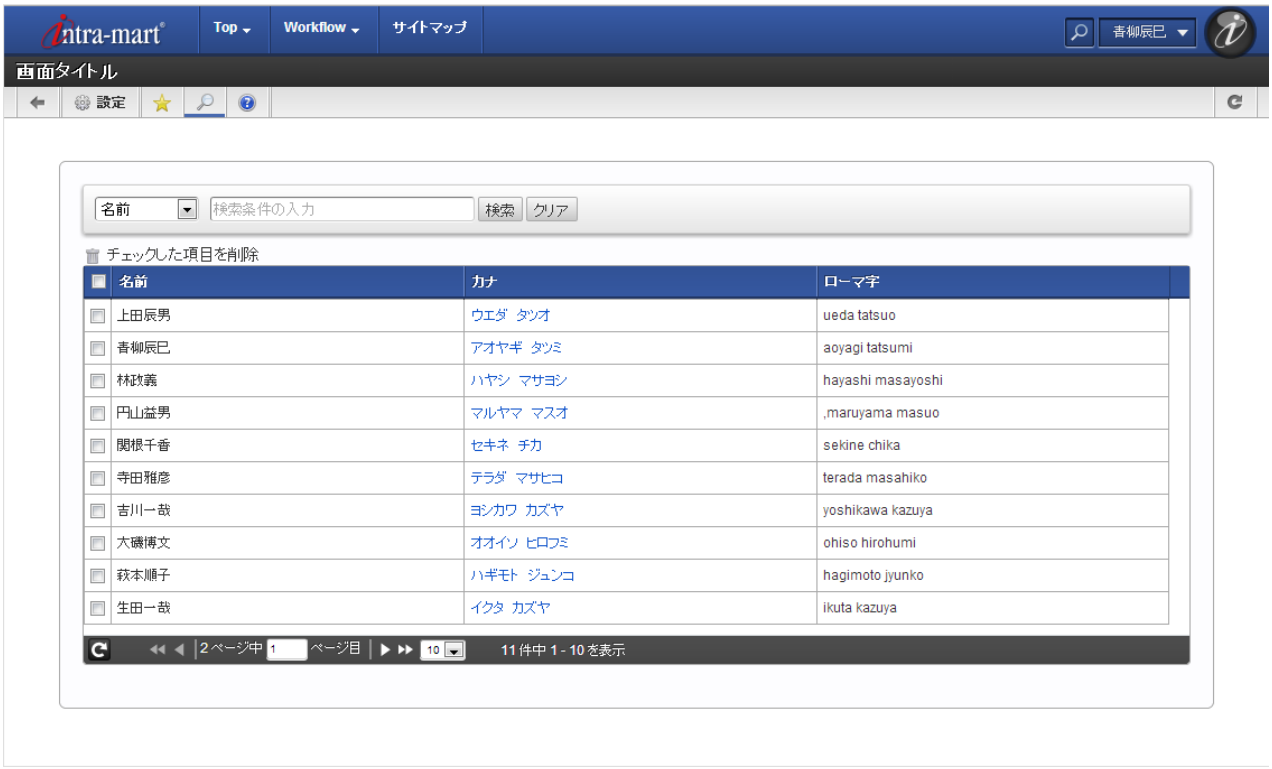
- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

実装サンプル 1

画面レイアウト



HTML / クライアントサイド JavaScript

```
<!-- ヘッド情報 -->
<imart type="head">
 <title>一覧画面</title>
</imart>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
  <ul class="imui-list-toolbar">
   <!-- 戻る -->
   <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- 設定ボタン -->
   <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- タブ お気に入りボタン -->
   <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
   <!-- タブ 検索ボタン -->
   <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
   <!-- タブ ヘルプボタン -->
   <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
  </ul>
  <!-- ツールバー右側 -->
  <ul class="imui-list-box-toolbar-utility">
   <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
 <div class="imui-box-operation cf">
  <div class="pr-10 float-L">
   <imart type="imuiSelect" id="search-list" list=$bind.selectbox />
  </div>
  <div class="float-L">
   <imart type="imuiTextbox" id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
   <imart type="imuiButton" value="検索" id="submit" class="imui-button" />
   <imart type="imuiButton" value="クリア" id="clear" class="imui-button" />
  </div>
 </div>
 <a id="on-delete-row" class="imui-unaccented" ><span class="im-ui-icon-common-16-trashbox mr-5"></span>チェックした項
目を削除</a>
 <imart type="imuiListTable" data=$bind.listtable id="listtable"  multiSelect="true" viewRecords="true" height="300">
  <pager rowNum="10" rowList="1,5,10" />
  <cols>
   <col name="col1" caption="名前" />
   <col name="col2" caption="カナ" >
    <showLink baseLinkUrl="sample" showAction="/listTable"/>
   </col>
   <col name="col3" caption="ローマ字" />
  </cols>
 </imart>
</div>

<script>
 $(function(){
  $('#on-delete-row').on('click',function(){
   imuiConfirm('選択したデータを削除します。よろしいですか？','',function(){
    // 削除処理
   })
  })
  $('#submit').on('click',function(){
   // 検索ボタン押下時の処理
  })
```

```
    $('#clear').on('click',function(){
      // クリアボタン押下時の処理
    });
  })

</script>
```

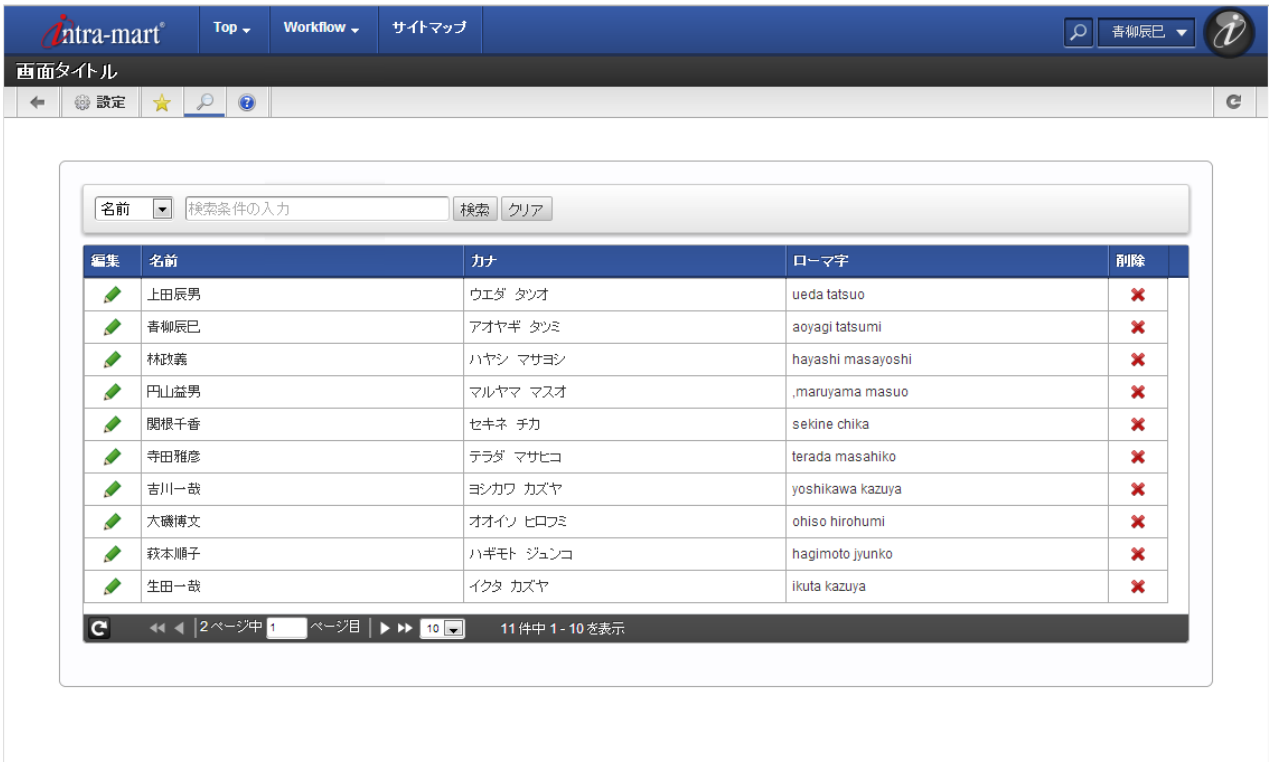
サーバサイド JavaScript

```
let $bind = {};

function init(request) {
    $bind.listtable = [
        {"col1":"上田辰男", "col2":"ウエダ　タツオ", "col3":"ueda tatsuo"},
        {"col1":"青柳辰巳", "col2":"アオヤギ　タツミ", "col3":"aoyagi tatsumi"},
        {"col1":"林政義", "col2":"ハヤシ　マサヨシ",　"col3":"hayashi masayoshi"},
        {"col1":"円山益男", "col2":"マルヤマ　マスオ", "col3":"maruyama masuo"},
        {"col1":"関根千香", "col2":"セキネ　チカ", "col3":"sekine chika"},
        {"col1":"寺田雅彦", "col2":"テラダ　マサヒコ", "col3":"terada masahiko"},
        {"col1":"吉川一哉", "col2":"ヨシカワ　カズヤ", "col3":"yoshikawa kazuya"},
        {"col1":"大磯博文", "col2":"オオイソ　ヒロフミ", "col3":"ohiso hirohumi"},
        {"col1":"萩本順子", "col2":"ハギモト　ジュンコ", "col3":"hagimoto jyunko"},
        {"col1":"生田一哉", "col2":"イクタ　カズヤ", "col3":"ikuta kazuya"},
        {"col1":"片山聡", "col2":"カタヤマ　サトシ",　"col3":"katayama satoshi"}
    ];
    $bind.selectbox = [ {
      type : "group",
      label : "列検索",
      list : [ {
        label : "名前",
        value : 1,
         selected : true
      }, {
        label : "カナ",
        value : 2
      }, {
        label : "ローマ字",
        value : 3
      } ]
    } ];
}
```

実装サンプル 2

画面レイアウト



HTML / クライアントサイド JavaScript

```
<!-- ヘッド情報 -->
<imart type="head">
 <title>一覧画面</title>
</imart>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
  <ul class="imui-list-toolbar">
   <!-- 戻る -->
   <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- 設定ボタン -->
   <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
   <!-- タブ お気に入りボタン -->
   <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
   <!-- タブ 検索ボタン -->
   <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
   <!-- タブ ヘルプボタン -->
   <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
   <!-- 区切り線 -->
   <li class="icon-split"></li>
  </ul>
  <!-- ツールバー右側 -->
  <ul class="imui-list-box-toolbar-utility">
   <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
 <div class="imui-box-operation cf">
  <div class="pr-10 float-L">
   <imart type="imuiSelect" id="search-list" list=$bind.selectbox />
  </div>
  <div class="float-L">
   <imart type="imuiTextbox" id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
   <imart type="imuiButton" value="検索" id="submit" class="imui-button" />
   <imart type="imuiButton" value="クリア" id="clear" class="imui-button" />
  </div>
 </div>
 <imart type="imuiListTable" data=$bind.listtable id="listtable" viewRecords="true" onCellSelect="onCellSelect" height="300">
  <pager rowNum="10" rowList="1,5,10" />
  <cols>
   <col name="col0" caption="編集" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
    <showIcon iconClass="im-ui-icon-common-16-update" />
   </col>
   <col name="col1" caption="名前" />
   <col name="col2" caption="カナ" />
   <col name="col3" caption="ローマ字" />
   <col name="col0" caption="削除" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
    <showIcon iconClass="im-ui-icon-common-16-ng" />
   </col>
  </cols>
 </imart>
</div>


<script>
 $(function(){
  onCellSelect = function(rowid,iCol,cellcontent,e){
   if($(cellcontent).hasClass('im-ui-icon-common-16-ng')){
    imuiConfirm('削除します。よろしいですか？','',function(){
     // 削除処理
   })
```

```
    }
    if($(cellcontent).hasClass('im-ui-icon-common-16-update')){
     // 編集処理
    }
   }
   $('#submit').on('click',function(){
    // 検索ボタン押下時の処理
   })
   $('#clear').on('click',function(){
    // クリアボタン押下時の処理
   });
  })
  function onCellAttr(){
   return 'style="min-width:30px;"'
  }
</script>
```

サーバサイド JavaScript

```
let $bind = {};

function init(request) {
   $bind.listtable = [
      {"col0":"", "col1":"上田辰男", "col2":"ウエダ　タツオ", "col3":"ueda tatsuo"},
      {"col0":"", "col1":"青柳辰巳", "col2":"アオヤギ　タツミ", "col3":"aoyagi tatsumi"},
      {"col0":"", "col1":"林政義", "col2":"ハヤシ　マサヨシ",  "col3":"hayashi masayoshi"},
      {"col0":"", "col1":"円山益男", "col2":"マルヤマ　マスオ", "col3":"maruyama masuo"},
      {"col0":"", "col1":"関根千香", "col2":"セキネ　チカ", "col3":"sekine chika"},
      {"col0":"", "col1":"寺田雅彦", "col2":"テラダ　マサヒコ", "col3":"terada masahiko"},
      {"col0":"", "col1":"吉川一哉", "col2":"ヨシカワ　カズヤ", "col3":"yoshikawa kazuya"},
      {"col0":"", "col1":"大磯博文", "col2":"オオイソ　ヒロフミ", "col3":"ohiso hirohumi"},
      {"col0":"", "col1":"萩本順子", "col2":"ハギモト　ジュンコ", "col3":"hagimoto jyunko"},
      {"col0":"", "col1":"生田一哉", "col2":"イクタ　カズヤ", "col3":"ikuta kazuya"},
      {"col0":"", "col1":"片山聡", "col2":"カタヤマ　サトシ",  "col3":"katayama satoshi"}
   ];
   $bind.selectbox = [ {
    type : "group",
      label : "列検索",
      list : [ {
        label : "名前",
        value : 1,
         selected : true
      }, {
        label : "カナ",
        value : 2
      }, {
        label : "ローマ字",
        value : 3
      } ]
     } ];
  }
```

画面遷移

項目

画面遷移がある場合（登録、更新など）の実装例

画面遷移がある場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajax でのデータ送信と画面遷移 – imuiAjaxSubmit

画面遷移あり HTML （ヘッド情報）

```
<imart type="head">
 <title>新規登録画面</title>
 <!-- Load library -->
 <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
 <script src="ui/js/imui-form-util.js"></script>
 <imart type="imuiValidationRule" rule="foo/bar#hoge" rulesName="rules" messagesName="messages" />
 <script type="text/javascript">
  (function($) {
    $(document).ready(function() {
      // Form の 2 度押し防止
      imuiDisableOnSubmit('#form');

      // 参照画面へ引き渡すキーの配列生成
      var optionalData = ['user_cd'];

      // 登録ボタンクリック
      $('#register-button').click(function() {

        // バリデーションチェック
        if (imuiValidate('#form', rules, messages)) {
          // 確認ダイアログ表示
          imuiConfirm(
            '<imart type="string" value=$data.dialogMessages.message escapeJs="true" />', // メッセージ
            '<imart type="string" value=$data.dialogMessages.title escapeJs="true" />',  // タイトル
            function() { // OKクリック時のコールバック関数
              // Ajax でのデータ送信と次画面への遷移
              imuiAjaxSubmit('#form', 'POST', 'json', 'reference/list/views/list');
            }
          );
        }
      });
    });
  })(jQuery);
 </script>
</imart>
```

画面遷移あり HTML （画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。
または、 *更新画面の実装サンプル* を参照してください。

コラム
imuiValidate を利用するときは、<input type=”submit”/> ではなく、<input type=”button”/> を利用してください。

画面遷移あり サーバサイドJavaScript

```
/**
 * 初期化処理
 *
 * ・・・
 * @validate foo/bar_validation#init
 * @onerror handleErrors
 */
function init(request) {
  // 初期化の際の処理
}
function handleErrors() {
  // 入力チェックに失敗した際の処理
}
```

コラム

- バリデーションチェックについて
  実装例の中で、バリデーションチェック（送信値チェック）をしています。
  送信値が規定のルールに反する場合は、確認ダイアログを表示せず処理を中断し、ルールに反している入力箇所にエラーの旨を表示します。
  詳しくは、 Jssp Validator を参照してください。
- Ajaxについて
  Ajax の呼び出し先 を参照してください。
- 登録・更新画面について
  基本的な画面の作成方法があります。
  詳しくは、 *登録画面* を参照してください。
  または、 *更新画面* を参照してください。

画面遷移がない場合の実装例

画面遷移がない場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajaxでのデータ送信 – imuiAjaxSend

画面遷移なし HTML（ヘッド情報）

```
<imart type="head">
 <title>更新／削除画面</title>
 <!-- Load library -->
 <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
 <script src="ui/js/imui-form-util.js"></script>
 <imart type="imuiValidationRule" rule="foo/bar#hoge" rulesName="rules" messagesName="messages" />
 <script type="text/javascript">
  (function($) {
   $(document).ready(function() {
    // Form の 2 度押し防止
    imuiDisableOnSubmit('#form');

    // 参照画面へ引き渡すキーの配列生成
    var optionalData = ['user_cd'];

    // 更新ボタンクリック
    $('#update-button').click(function() {
     // バリデーションチェック
     if (imuiValidate('#form', rules, messages)) {
      // 確認ダイアログ表示
      imuiConfirm(
       '<imart type="string" value=$data.dialogMessages.message escapeJs="true" />', // メッセージ
       '<imart type="string" value=$data.dialogMessages.title escapeJs="true" />', // タイトル
       function() { // OKクリック時のコールバック関数
        // Ajaxでのデータ送信
        imuiAjaxSend('#form', 'POST', 'json');
       }
      );
     }
    });
   });
  })(jQuery);
 </script>
</imart>
```

画面遷移なし HTML （画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。
または、 *更新画面の実装サンプル」* を参照してください。

画面遷移なし サーバサイドJavaScript

```
/**
 * 初期化処理
 *
 * ・・・
 * @validate foo/bar_validation#init
 * @onerror handleErrors
 */

function init(request) {
   // 初期化の際にしたい処理
}
function handleErrors() {
   // 入力チェックに失敗した際にしたい処理
}
```

コラム

- バリデーションチェックについて
  実装例の中で、バリデーションチェック（送信値チェック）をしています。
  送信値が規定のルールに反する場合は、確認ダイアログを表示せず処理を中断し、ルールに反している入力箇所にエラーの旨を表示します。
  詳しくは、 Jssp Validator を参照してください。
- Ajaxについて
  Ajax の呼び出し先 を参照してください。
- 登録・更新画面について

  基本的な画面の作成方法があります。
  詳しくは、 *登録画面* を参照してください。
  または、 *更新画面* を参照してください。

Ajax 通信の利用方法

imuiAjaxSend, imuiAjaxSubmit を使用して Ajax 通信をする場合、呼び出し先のページでは処理成功時のメッセージ、処理失敗時のメッセージ、処理成功後に遷移するページへ引き渡すパラメータ、を返すことができます。

属性名	説明	型	必須
error	処理が失敗した場合 true、成功した場合 false を指定します。	boolean	o
successMessage	処理成功時のメッセージ。error: false の場合表示されます。	String	–
errorMessage	処理失敗時のメッセージ。error: true の場合表示されます。	String	–
detailMessages	処理失敗時の詳細なメッセージ。error: true の場合表示されます。	String/String[]	–
parameter	処理成功後に遷移するページへ引き渡すパラメータ。	Object	–

parameter について

オブジェクトのキーを input タグの name 属性に、値を value 属性にセットして submit します。
値に配列を指定することが可能です。ただし、1 次元配列のみサポートします。

parameter の指定方法と、次画面での取得例は以下のようになります。

parameter の指定

```
parameter: {
   key1: 'value1',
   key2: 'value2',
   array1: [ 'array1', 'array2', 'array3' ]
}
```

次画面での取得

```
function init(request) {
   var v1 = request.key1; // 'value1'
   var v2 = request.key2; // 'value2'
   var a1 = request.getParameterValues('array1'); // ['array1', 'array2', 'array3']
   ...
}
```

となります。

Ajax 実装例

```
function init(request) {
   ...
   response.setContentType('application/json; charset=utf-8');
   ...
   var resultObject = SomeAPI.doSomething();
   if (resultObject.error) {
      // 処理が失敗した場合
      response.sendMessageBodyString(
         ImJson.toJSONString({
            error: resultObject.error,
            errorMessage: resultObject.message,
            detailMessages: ['管理者にお問い合わせください', '連絡先：admin@xxx.xxx.xxx']
         })
      );
   }

   // 処理が成功した場合
   response.sendMessageBodyString(
      ImJson.toJSONString({
         error: false,
         errorMessage: '',
         successMessage: '登録しました',
         parameter:{
            param1: 'value1',
            param2: 'value2',
            array1: ['array1', 'array2', 'array3']
         }
      })
   );
}
```

基本的な画面の作り方 SAStruts+S2JDBC 開発編

登録画面

作成ガイド

登録画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明しています。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*

- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

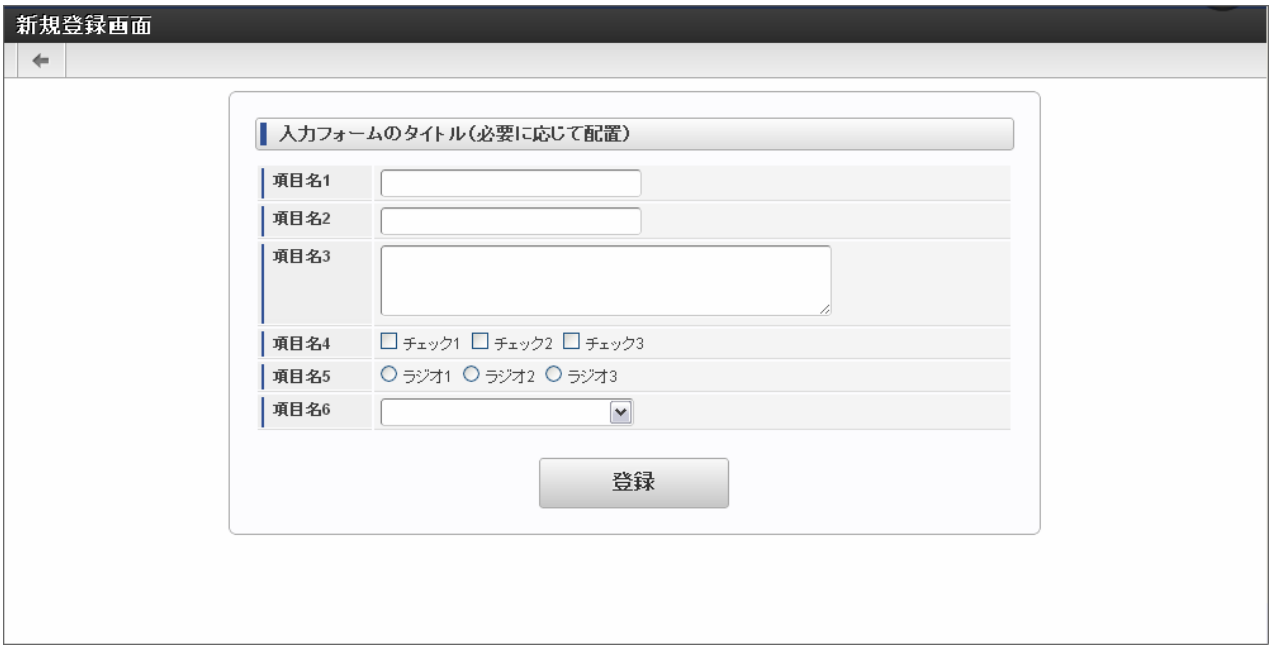
- *画面レイアウト*

補足事項

- 見出し（ h2 ～ h6 ）は必要に応じて配置します。必須部品ではありません。
- ボタンラベルは、基本的に「登録」とします。
  ユーザビリティを考慮して「作成」などにする変更する場合、関係画面の統一を図ってください。
- 同じデータの更新画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト



JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<!-- ヘッド情報 -->
<imui:head>
 <title>新規登録画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>新規登録画面</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <ul class="imui-list-toolbar">
   <!-- 戻るボタン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-narrow">
 <!-- h2 -->
 <div class="imui-chapter-title">
  <h2>入力フォームのタイトル（必要に応じて配置）</h2>
 </div>

 <form id="form" method="POST" class="target_form mt-10" action="hoge/register" onsubmit="return false;">
  <table class="imui-form">
   <tr>
    <th><label>項目名1</label></th>
    <td><imui:textbox id="textbox" name="textbox" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名2</label></th>
    <td><imui:password id="password" name="password" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名3</label></th>
    <td><imui:textArea id="textarea" name="textarea" width="350px" height="50px" /></td>
   </tr>
   <tr>
    <th><label>項目名4</label></th>
    <td>
     <imui:checkbox id="checkbox_1" name="checkbox" label="チェック1" />
     <imui:checkbox id="checkbox_2" name="checkbox" label="チェック2" />
     <imui:checkbox id="checkbox_3" name="checkbox" label="チェック3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名5</label></th>
    <td>
     <imui:radio id="radio_1" name="radio" label="ラジオ1" />
     <imui:radio id="radio_2" name="radio" label="ラジオ2" />
     <imui:radio id="radio_3" name="radio" label="ラジオ3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名6</label></th>
    <td>
     <imui:select id="inputItem6" name="inputItem6" width="200px" />
    </td>
   </tr>
  </table>
  <div class="imui-operation-parts">
   <imui:button value="登録" id="register-button" class="imui-large-button" />
  </div>
 </form>
</div>
```

作成ガイド

更新画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明ています。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

補足事項

- 見出し（ h2 ～ h6 ）は必要に応じて配置します。必須部品ではありません。
- 更新画面で、更新／削除を行えるようにする場合は、画面下部に左から更新ボタン、削除ボタンの順番に配置します。
  ボタンラベルは、基本的に「更新」「削除」とします。
  ユーザビリティを考慮して「保存」などにする場合、関係画面の統一を図ってください。
- 同じデータの登録画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト

JSP

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<!-- ヘッド情報 -->
<imui:head>
 <title>更新／削除画面</title>
       ・・・
</imui:head>

<!-- 画面タイトル  -->
<div class="imui-title">
 <h1>更新／削除画面</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <ul class="imui-list-toolbar">
   <!-- 戻るボタン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-narrow">
 <!-- h2 -->
 <div class="imui-chapter-title">
  <h2>入力フォームのタイトル（必要に応じて配置）</h2>
 </div>

 <form id="form" method="POST" class="target_form mt-10" action="hoge/update_delete" onsubmit="return false;">
  <table class="imui-form">
   <tr>
    <th><label>項目名1</label></th>
    <td><imui:textbox id="textbox" name="textbox" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名2</label></th>
    <td><imui:password id="password" name="password" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名3</label></th>
    <td><imui:textArea id="textarea" name="textarea" width="350px" height="50px" /></td>
   </tr>
   <tr>
    <th><label>項目名4</label></th>
    <td>
     <imui:checkbox id="checkbox_1" name="checkbox" label="チェック1" />
     <imui:checkbox id="checkbox_2" name="checkbox" label="チェック2" />
     <imui:checkbox id="checkbox_3" name="checkbox" label="チェック3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名5</label></th>
    <td>
     <imui:radio id="radio_1" name="radio" label="ラジオ1" />
     <imui:radio id="radio_2" name="radio" label="ラジオ2" />
     <imui:radio id="radio_3" name="radio" label="ラジオ3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名6</label></th>
    <td>
     <imui:select id="inputItem6" name="inputItem6" width="200px" />
    </td>
   </tr>
  </table>
  <div class="imui-operation-parts">
   <imui:button value="更新" id="update-button" class="imui-large-button" />
   <imui:button value="削除" id="delete-button" class="imui-large-button" />
  </div>
 </form>
</div>
```

一覧画面

項目

作成ガイド

一覧画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明しています。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*

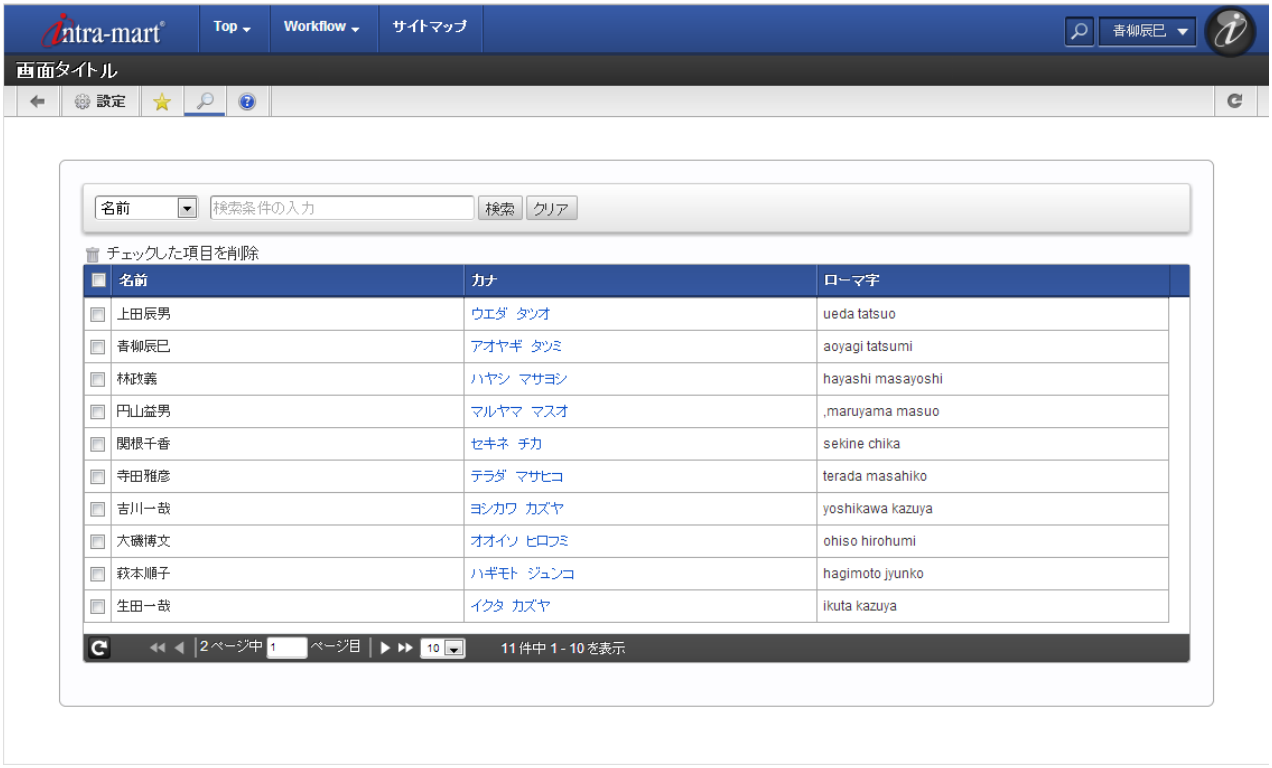
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

実装サンプル 1

画面レイアウト



JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Map"%>
<%@ page import="java.util.HashMap"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<%!

List<Map<String, Object>> selectbox = new ArrayList<Map<String, Object>>();
{
    List<Map<String, Object>> selectboxOption = new ArrayList<Map<String, Object>>();
    Map<String, Object> optGrp = new HashMap<String, Object>();
    optGrp.put("type", "group");
    optGrp.put("label", "行検索");
    List<Map<String, Object>> optList = new ArrayList<Map<String, Object>>();
    Map<String, Object> option = new HashMap<String, Object>();
    option.put("label", "名前");
    option.put("value","1");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "カナ");
    option.put("value","2");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "ローマ字");
    option.put("value","3");
    optList.add(option);
    optGrp.put("list", optList);
    selectbox.add(optGrp);
}
Map<String, Object> createValueMap(Object... objArray) {
    Map<String, Object> map = new HashMap<String, Object>();

    int size = objArray.length;

    for (int i = 0; i < size; i++) {

        if ((i + 1) % 2 == 1) {
            map.put(objArray[i].toString(), objArray[i + 1]);
        }
    }

    return map;
}
%>
<%

 //リストテーブルデータ
 List<Map<String, Object>> listTable = new ArrayList<Map<String, Object>>();
 listTable.add(createValueMap("col1","上田辰男", "col2","ウエダ　タツオ", "col3","ueda tatsuo"));
 listTable.add(createValueMap("col1","青柳辰巳", "col2","アオヤギ　タツミ", "col3","aoyagi tatsumi"));
 listTable.add(createValueMap("col1","林政義", "col2","ハヤシ　マサヨシ", "col3","hayashi masayoshi"));
 listTable.add(createValueMap("col1","円山益男", "col2","マルヤマ　マスオ", "col3","maruyama masuo"));
 listTable.add(createValueMap("col1","関根千香", "col2","セキネ　チカ", "col3","sekine chika"));
 listTable.add(createValueMap("col1","寺田雅彦", "col2","テラダ　マサヒコ" , "col3","terada masahiko"));
 listTable.add(createValueMap("col1","吉川一哉", "col2","ヨシカワ　カズヤ", "col3","yoshikawa kazuya"));
 listTable.add(createValueMap("col1","大磯博文", "col2","オオイソ　ヒロフミ", "col3","ohiso hirohumi"));
 listTable.add(createValueMap("col1","萩本順子", "col2","ハギモト　ジュンコ", "col3","hagimoto jyunko"));
 listTable.add(createValueMap("col1","生田一哉", "col2","イクタ　カズヤ", "col3","ikuta kazuya"));
 listTable.add(createValueMap("col1","片山聡", "col2","カタヤマ　サトシ", "col3","katayama satoshi"));
%>

<!-- ヘッド情報 -->
<imui:head>
  <title>一覧画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
  <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
  <div class="imui-toolbar-inner">
```

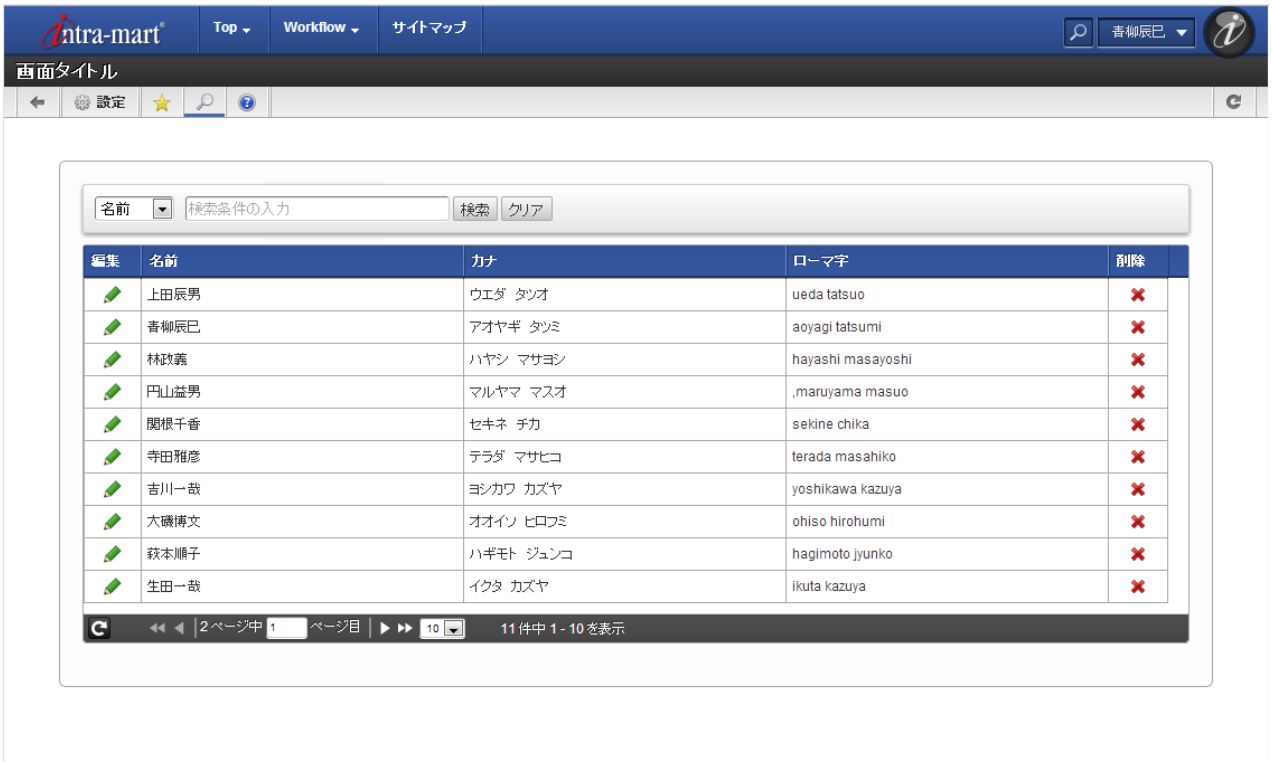
```
    <!-- ツールバー左側 -->
    <ul class="imui-list-toolbar">
      <!-- 戻る -->
      <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- 設定ボタン -->
      <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- タブ お気に入りボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
      <!-- タブ 検索ボタン -->
      <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
      <!-- タブ ヘルプボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
    </ul>
    <!-- ツールバー右側 -->
    <ul class="imui-list-box-toolbar-utility">
      <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
    </ul>
  </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
  <div class="imui-box-operation cf">
    <div class="pr-10 float-L">
      <imui:select id="search-list" list="<%= selectbox %>" />
    </div>
    <div class="float-L">
      <imui:textbox id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
      <imui:button value="検索" id="submit" class="imui-button" />
      <imui:button value="クリア" id="clear" class="imui-button" />
    </div>
  </div>
  <a id="on-delete-row" class="imui-unaccented" ><span class="im-ui-icon-common-16-trashbox mr-5"></span>チェックした
項目を削除</a>
  <imui:listTable data="<%= listTable %>" id="listtable"  multiSelect="true" viewRecords="true" height="300">
    <pager rowNum="10" rowList="1,5,10" />
    <cols>
      <col name="col1" caption="名前" />
      <col name="col2" caption="カナ" >
        <showLink baseLinkUrl="sample" showAction="/listTable"/>
      </col>
      <col name="col3" caption="ローマ字" />
    </cols>
  </imui:listTable>
</div>

<script>
  $(function(){
    $('#on-delete-row').on('click',function(){
      imuiConfirm('選択したデータを削除します。よろしいですか？','',function(){
      // 削除処理
      })
    })
    $('#submit').on('click',function(){
      // 検索ボタン押下時の処理
    })
    $('#clear').on('click',function(){
      // クリアボタン押下時の処理
    });
  })
</script>
```

実装サンプル 2

画面レイアウト


intra-mart
Top
Workflow
サイトマップ
青柳辰巳
画面タイトル
設定
名前
検索条件の入力
検索
クリア
編集
名前
カナ
ローマ字
削除
上田辰男
ウエダ タツオ
ueda tatsuo
青柳辰巳
アオヤギ タツミ
aoyagi tatsumi
林政義
ハヤシ マサヨシ
hayashi masayoshi
円山益男
マルヤマ マスオ
,maruyama masuo
関根千香
セキネ チカ
sekine chika
寺田雅彦
テラダ マサヒコ
terada masahiko
吉川一哉
ヨシカワ カズヤ
yoshikawa kazuya
大磯博文
オオイソ ヒロフミ
ohiso hirohumi
荻本順子
ハギモト ジュンコ
hagimoto jyunko
生田一哉
イクタ カズヤ
ikuta kazuya
2 ページ中 1 ページ目
10
11 件中 1 - 10 を表示

JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Map"%>
<%@ page import="java.util.HashMap"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<%!

List<Map<String, Object>> selectbox = new ArrayList<Map<String, Object>>();
{
    List<Map<String, Object>> selectboxOption = new ArrayList<Map<String, Object>>();
    Map<String, Object> optGrp = new HashMap<String, Object>();
    optGrp.put("type", "group");
    optGrp.put("label", "行検索");
    List<Map<String, Object>> optList = new ArrayList<Map<String, Object>>();
    Map<String, Object> option = new HashMap<String, Object>();
    option.put("label", "名前");
    option.put("value","1");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "カナ");
    option.put("value","2");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "ローマ字");
    option.put("value","3");
    optList.add(option);
    optGrp.put("list", optList);
    selectbox.add(optGrp);
}
Map<String, Object> createValueMap(Object... objArray) {
    Map<String, Object> map = new HashMap<String, Object>();

    int size = objArray.length;

    for (int i = 0; i < size; i++) {

        if ((i + 1) % 2 == 1) {
            map.put(objArray[i].toString(), objArray[i + 1]);
        }
    }

    return map;
}
%>
<%
 //リストテーブルデータ
 List<Map<String, Object>> listTable = new ArrayList<Map<String, Object>>();
 listTable.add(createValueMap("col0","","col1","上田辰男", "col2","ウエダ　タツオ", "col3","ueda tatsuo"));
 listTable.add(createValueMap("col0","","col1","青柳辰巳", "col2","アオヤギ　タツミ", "col3","aoyagi tatsumi"));
 listTable.add(createValueMap("col0","","col1","林政義", "col2","ハヤシ　マサヨシ", "col3","hayashi masayoshi"));
 listTable.add(createValueMap("col0","","col1","円山益男", "col2","マルヤマ　マスオ", "col3","maruyama masuo"));
 listTable.add(createValueMap("col0","","col1","関根千香", "col2","セキネ　チカ", "col3","sekine chika"));
 listTable.add(createValueMap("col0","","col1","寺田雅彦", "col2","テラダ　マサヒコ" , "col3","terada masahiko"));
 listTable.add(createValueMap("col0","","col1","吉川一哉", "col2","ヨシカワ　カズヤ", "col3","yoshikawa kazuya"));
 listTable.add(createValueMap("col0","","col1","大磯博文", "col2","オオイソ　ヒロフミ", "col3","ohiso hirohumi"));
 listTable.add(createValueMap("col0","","col1","萩本順子", "col2","ハギモト　ジュンコ", "col3","hagimoto jyunko"));
 listTable.add(createValueMap("col0","","col1","生田一哉", "col2","イクタ　カズヤ", "col3","ikuta kazuya"));
 listTable.add(createValueMap("col0","","col1","片山聡", "col2","カタヤマ　サトシ", "col3","katayama satoshi"));
%>

<!-- ヘッド情報 -->
<imui:head>
  <title>一覧画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
  <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
  <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
```

```
    <ul class="imui-list-toolbar">
      <!-- 戻る -->
      <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- 設定ボタン -->
      <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- タブ お気に入りボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
      <!-- タブ 検索ボタン -->
      <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
      <!-- タブ ヘルプボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
    </ul>
    <!-- ツールバー右側 -->
    <ul class="imui-list-box-toolbar-utility">
      <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
    </ul>
  </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
  <div class="imui-box-operation cf">
    <div class="pr-10 float-L">
      <imui:select id="search-list" list="<%= selectbox %>" />
    </div>
    <div class="float-L">
      <imui:textbox id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
      <imui:button value="検索" id="submit" class="imui-button" />
      <imui:button value="クリア" id="clear" class="imui-button" />
    </div>
  </div>
  <imui:listTable data="<%= listTable %>" id="listtable" viewRecords="true" onCellSelect="onCellSelect" height="300">
    <pager rowNum="10" rowList="1,5,10" />
    <cols>
      <col name="col0" caption="編集" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
        <showIcon iconClass="im-ui-icon-common-16-update" />
      </col>
      <col name="col1" caption="名前" />
      <col name="col2" caption="カナ" />
      <col name="col3" caption="ローマ字" />
      <col name="col0" caption="削除" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
        <showIcon iconClass="im-ui-icon-common-16-ng" />
      </col>
    </cols>
  </imui:listTable>
</div>

<script>
  $(function(){
    onCellSelect = function(rowid,iCol,cellcontent,e){
      if($(cellcontent).hasClass('im-ui-icon-common-16-ng')){
        imuiConfirm('削除します。よろしいですか？','',function(){
          // 削除処理
      })
      }
      if($(cellcontent).hasClass('im-ui-icon-common-16-update')){
        // 編集処理
      }
    }
    $('#submit').on('click',function(){
      // 検索ボタン押下時の処理
    })
    $('#clear').on('click',function(){
      // クリアボタン押下時の処理
    });
  })
  function onCellAttr(){
    return 'style="min-width:30px;"'
  }
</script>
```

画面遷移

項目

画面遷移がある場合（登録、更新など）の実装例

画面遷移がある場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajax でのデータ送信と画面遷移 – imuiAjaxSubmit

画面遷移あり JSP （ヘッド情報）

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
 <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
 <%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
 <imui:head>
  <title>新規登録画面</title>
  <!-- Load library -->
  <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
  <script type="text/javascript">

   var rules = {
    textbox: {
     required:true,
     maxlength:20
    },
    textarea: {
     required:true
    }
   };

   var messages = {
    textbox: {
     required:"タイトルは必須です。",
     maxlength:"タイトルは20文字以内で入力してください"
    },
    textarea: {
     required:"テキストエリアは必須です。"
    }
   };

   (function($) {
    $(document).ready(function() {
     // Form の 2 度押し防止
     imuiDisableOnSubmit('#form');

     // 参照画面へ引き渡すキーの配列生成
     var optionalData = ['user_cd'];

     // 登録ボタンクリック
     $('#register-button').click(function() {

      // バリデーションチェック
      if (imuiValidate('#form', rules, messages)) {
       // 確認ダイアログ表示
       imuiConfirm(
        '<c:out value="message" />', // メッセージ
        '<c:out value="title" />' ,  // タイトル
        function() { // OKクリック時のコールバック関数
         // Ajax でのデータ送信と次画面への遷移
         imuiAjaxSubmit('#form', 'POST', 'json', 'reference/list/views/list');
        }
       );
      }
     });
    });
   })(jQuery);
  </script>
 </imui:head>
```

画面遷移あり JSP （画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。
または、 *更新画面の実装サンプル* を参照してください。

> **コラム**
> imuiValidate を利用するときは、<input type="submit"/> ではなく、<input type="button"/> を利用してください。

画面遷移がない場合の実装例

画面遷移がない場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajaxでのデータ送信 – imuiAjaxSend

画面遷移なし JSP （ヘッダー／フッター）

```
<imui:head>
 <title>更新／削除画面</title>
 <!-- Load library -->
 <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
 <script type="text/javascript">

  var rules = {
   textbox: {
    required:true,
    maxlength:20
   },
   textarea: {
    required:true
   }
  };

  var messages = {
   textbox: {
    required:"タイトルは必須です。",
    maxlength:"タイトルは20文字以内で入力してください"
   },
   textarea: {
    required:"テキストエリアは必須です。"
   }
  };

  (function($) {
   $(document).ready(function() {
    // Form の 2 度押し防止
    imuiDisableOnSubmit('#form');

    // 参照画面へ引き渡すキーの配列生成
    var optionalData = ['user_cd'];

    // 更新ボタンクリック
    $('#update-button').click(function() {
     // バリデーションチェック
     if (imuiValidate('#form', rules, messages)) {
      // 確認ダイアログ表示
      imuiConfirm(
       '<c:out value="message" />', // メッセージ
       '<c:out value="title" />' ,  // タイトル
       function() { // OKクリック時のコールバック関数
        // Ajaxでのデータ送信
        imuiAjaxSend('#form', 'POST', 'json');
       }
      );
     }
    });
   });
  })(jQuery);
 </script>
</imui:head>
```

画面遷移なし JSP（画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。
または、 *更新画面の実装サンプル* を参照してください。

Ajax 通信の利用方法

imuiAjaxSend, imuiAjaxSubmit を使用して Ajax 通信をする場合、呼び出し先のページでは処理成功時のメッセージ、処理失敗時のメッセージ、処理成功後に遷移するページへ引き渡すパラメータ、を返すことができます。

属性名	説明	型	必須
error	処理が失敗した場合 true、成功した場合 false を指定します。	boolean	o
successMessage	処理成功時のメッセージ。error: false の場合表示されます。	String	–
errorMessage	処理失敗時のメッセージ。error: true の場合表示されます。	String	–
detailMessages	処理失敗時の詳細なメッセージ。error: true の場合表示されます。	String/String[]	–

属性名	説明	型	必須
parameter	処理成功後に遷移するページへ引き渡すパラメータ。	Object	–

parameter について

オブジェクトのキーを input タグの name 属性に、値を value 属性にセットして submit します。
値に配列を指定することが可能です。ただし、1 次元配列のみサポートします。

parameter の指定方法と、次画面での取得例は以下のようになります。

parameter の指定

```
Map<String, Object> parameter = new HashMap<String, Object>();
ArrayList<String> arrayList = new ArrayList<String>();
arrayList.add("array1");
arrayList.add("array2");
arrayList.add("array3");

parameter.put("param1", "value1");
parameter.put("param2", "value2");
parameter.put("array1", arrayList);
```

次画面での取得

```
public class Test2Action {

   public String param1;

   public String param2;

   public ArrayList<String> array1;

   @Execute(validator=false)
   public String index(){
     return "/sa/hello/output.jsp";
   }
}
```

となります。

Ajax 実装例

非同期で返却するレスポンスオブジェクトを定義します。

```
package sample.sastruts.dto.samplesp;

public class MyAjaxResponse {
   public String result;
   public boolean error;
   public String errorMessage;
   public String successMessage;
   public String[] detailMessages;
   public Map<String, Object> parameter;
}
```

アクションクラスでは、上記レスポンスオブジェクトに対して値をセットし、JSON 文字列に変換します。
変換した結果の文字列をプロパティにセットします。

```
public String ajaxResponse;

@Execute(validator = false)
public String ajax() {
    MyAjaxResponse responseObject = new MyAjaxResponse();
    try {
        Map<String, Object> parameter = new HashMap<String, Object>();
        ArrayList<String> arrayList = new ArrayList<String>();
        arrayList.add("array1");
        arrayList.add("array2");
        arrayList.add("array3");

        parameter.put("param1", "value1");
        parameter.put("param2", "value2");
        parameter.put("array1", arrayList);

        // 成功時
        responseObject.result = "success";
        responseObject.error = false;
        responseObject.successMessage = "登録が完了しました。";
        responseObject.parameter = parameter;
    } catch (Exception e) {
        responseObject.error = true;
        responseObject.errorMessage = "データ登録時にエラーが発生しました。";
        responseObject.detailMessages = new String[] { "管理者にお問い合わせください。" };
    }
    // レスポンスオブジェクトをJSON文字列に変換
    ajaxResponse = JSON.encode(responseObject);

    return "sample/ajaxResponse.jsp";
}
```

アクションクラスで生成した JSON 文字列をクライアントに application/json として返します。
上記 ajax メソッドの返却値 “sample/ajaxResponse.jsp” に相当します。

```
<%@ page language="java" contentType="application/json; charset=UTF-8" pageEncoding="UTF-8" %>
${ajaxResponse}
```

基本的な画面の作り方 TERASOLUNA Server Framework for Java (5.x) 開発編

登録画面

項目

作成ガイド

登録画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明ています。

- *画面タイトル*

- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

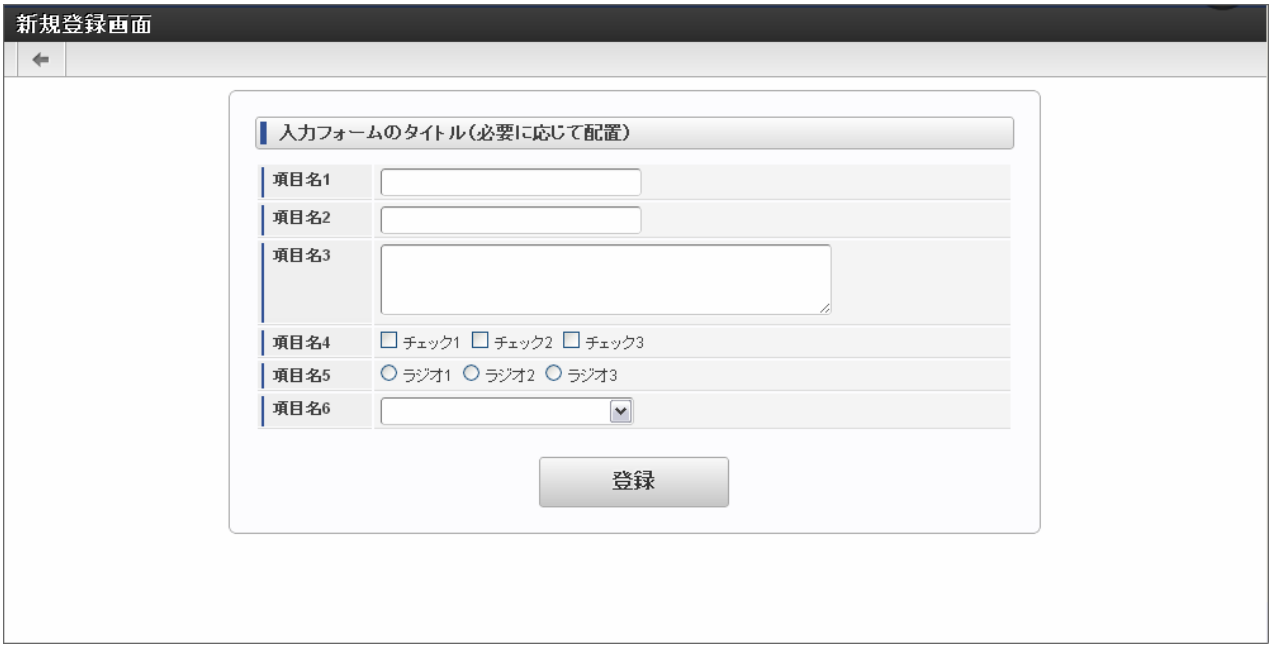
- *画面レイアウト*

補足事項

- 見出し（ h2 ～ h6 ）は必要に応じて配置します。必須部品ではありません。
- ボタンラベルは、基本的に「登録」とします。
  ユーザビリティを考慮して「作成」などにする変更する場合、関係画面の統一を図ってください。
- 同じデータの更新画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト



JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<!-- ヘッド情報 -->
<imui:head>
 <title>新規登録画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>新規登録画面</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <ul class="imui-list-toolbar">
   <!-- 戻るボタン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-narrow">
 <!-- h2 -->
 <div class="imui-chapter-title">
  <h2>入力フォームのタイトル（必要に応じて配置）</h2>
 </div>

 <form id="form" method="POST" class="target_form mt-10" action="example/tgfw/register" onsubmit="return false;">
  <table class="imui-form">
   <tr>
    <th><label>項目名1</label></th>
    <td><imui:textbox id="textbox" name="textbox" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名2</label></th>
    <td><imui:password id="password" name="password" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名3</label></th>
    <td><imui:textArea id="textarea" name="textarea" width="350px" height="50px" /></td>
   </tr>
   <tr>
    <th><label>項目名4</label></th>
    <td>
     <imui:checkbox id="checkbox_1" name="checkbox" label="チェック1" />
     <imui:checkbox id="checkbox_2" name="checkbox" label="チェック2" />
     <imui:checkbox id="checkbox_3" name="checkbox" label="チェック3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名5</label></th>
    <td>
     <imui:radio id="radio_1" name="radio" label="ラジオ1" />
     <imui:radio id="radio_2" name="radio" label="ラジオ2" />
     <imui:radio id="radio_3" name="radio" label="ラジオ3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名6</label></th>
    <td>
     <imui:select id="inputItem6" name="inputItem6" width="200px" />
    </td>
   </tr>
  </table>
  <div class="imui-operation-parts">
   <imui:button value="登録" id="register-button" class="imui-large-button" />
  </div>
 </form>
</div>
```

更新画面

項目

作成ガイド

更新画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明しています。

- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

補足事項

- 見出し（ h2 ～ h6 ）は必要に応じて配置します。必須部品ではありません。
- 更新画面で、更新／削除を行えるようにする場合は、画面下部に左から更新ボタン、削除ボタンの順番に配置します。
  ボタンラベルは、基本的に「更新」「削除」とします。
  ユーザビリティを考慮して「保存」などにする場合、関係画面の統一を図ってください。
- 同じデータの登録画面が存在する場合は、レイアウトの差異は最小限にしましょう。

実装サンプル

画面レイアウト

JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<!-- ヘッド情報 -->
<imui:head>
 <title>更新／削除画面</title>
       ・・・
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
 <h1>更新／削除画面</h1>
</div>

<!-- ツールバー -->
<div class="imui-toolbar-wrap">
 <div class="imui-toolbar-inner">
  <ul class="imui-list-toolbar">
   <!-- 戻るボタン -->
   <li><a href="#" class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
  </ul>
 </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-narrow">
 <!-- h2 -->
 <div class="imui-chapter-title">
  <h2>入力フォームのタイトル（必要に応じて配置）</h2>
 </div>

 <form id="form" method="POST" class="target_form mt-10" action="example/tgfw/update_delete" onsubmit="return
false;">
  <table class="imui-form">
   <tr>
    <th><label>項目名1</label></th>
    <td><imui:textbox id="textbox" name="textbox" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名2</label></th>
    <td><imui:password id="password" name="password" style="width: 200px;" /></td>
   </tr>
   <tr>
    <th><label>項目名3</label></th>
    <td><imui:textArea id="textarea" name="textarea" width="350px" height="50px" /></td>
   </tr>
   <tr>
    <th><label>項目名4</label></th>
    <td>
     <imui:checkbox id="checkbox_1" name="checkbox" label="チェック1" />
     <imui:checkbox id="checkbox_2" name="checkbox" label="チェック2" />
     <imui:checkbox id="checkbox_3" name="checkbox" label="チェック3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名5</label></th>
    <td>
     <imui:radio id="radio_1" name="radio" label="ラジオ1" />
     <imui:radio id="radio_2" name="radio" label="ラジオ2" />
     <imui:radio id="radio_3" name="radio" label="ラジオ3" />
    </td>
   </tr>
   <tr>
    <th><label>項目名6</label></th>
    <td>
     <imui:select id="inputItem6" name="inputItem6" width="200px" />
    </td>
   </tr>
  </table>
  <div class="imui-operation-parts">
   <imui:button value="更新" id="update-button" class="imui-large-button" />
   <imui:button value="削除" id="delete-button" class="imui-large-button" />
  </div>
 </form>
</div>
```

一覧画面

項目

作成ガイド

一覧画面の基本的な作成ガイドについて記載します。
詳細は以下リンク先を参照してください。
ただし、以下リンク先は、スクリプト開発にて説明ています。

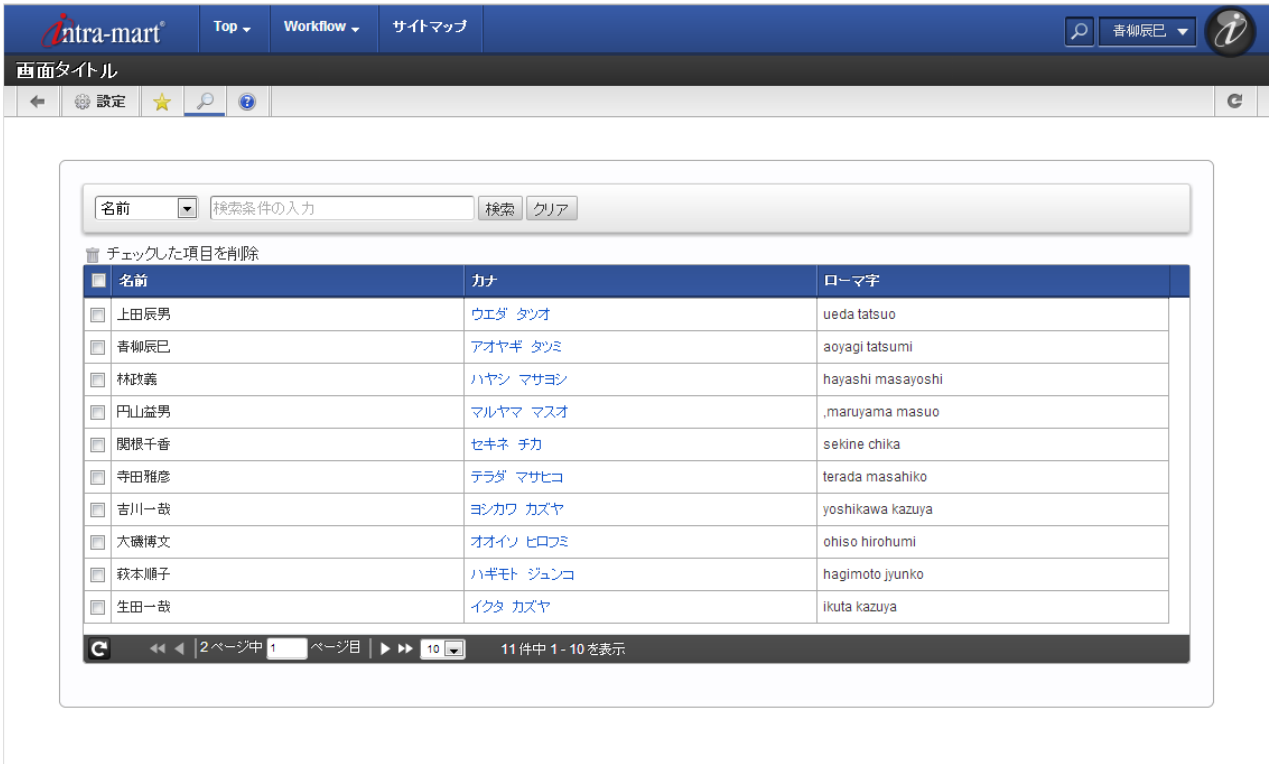
- *画面タイトル*
- *ツールバー*
- *見出し*
- *テーブル*
- *入力フォーム*
- *ボタン*
- *処理リンク／処理アイコン*
- *画面遷移*
- *アラートとコンファーム*

実装サンプルは、*基本的な画面レイアウト* を使用しています。
詳細は以下リンク先を参照してください。

- *画面レイアウト*

実装サンプル 1

画面レイアウト



JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Map"%>
<%@ page import="java.util.HashMap"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<%!

List<Map<String, Object>> selectbox = new ArrayList<Map<String, Object>>();
{
   List<Map<String, Object>> selectboxOption = new ArrayList<Map<String, Object>>();
   Map<String, Object> optGrp = new HashMap<String, Object>();
   optGrp.put("type", "group");
   optGrp.put("label", "行検索");
   List<Map<String, Object>> optList = new ArrayList<Map<String, Object>>();
   Map<String, Object> option = new HashMap<String, Object>();
   option.put("label", "名前");
   option.put("value","1");
   optList.add(option);
   option = new HashMap<String, Object>();
   option.put("label", "カナ");
   option.put("value","2");
   optList.add(option);
   option = new HashMap<String, Object>();
   option.put("label", "ローマ字");
   option.put("value","3");
   optList.add(option);
   optGrp.put("list", optList);
   selectbox.add(optGrp);
}
Map<String, Object> createValueMap(Object... objArray) {
   Map<String, Object> map = new HashMap<String, Object>();

   int size = objArray.length;

   for (int i = 0; i < size; i++) {

      if ((i + 1) % 2 == 1) {
         map.put(objArray[i].toString(), objArray[i + 1]);
      }
   }

   return map;
}
%>
<%

 //リストテーブルデータ
 List<Map<String, Object>> listTable = new ArrayList<Map<String, Object>>();
 listTable.add(createValueMap("col1","上田辰男", "col2","ウエダ　タツオ", "col3","ueda tatsuo"));
 listTable.add(createValueMap("col1","青柳辰巳", "col2","アオヤギ　タツミ", "col3","aoyagi tatsumi"));
 listTable.add(createValueMap("col1","林政義", "col2","ハヤシ　マサヨシ", "col3","hayashi masayoshi"));
 listTable.add(createValueMap("col1","円山益男", "col2","マルヤマ　マスオ", "col3","maruyama masuo"));
 listTable.add(createValueMap("col1","関根千香", "col2","セキネ　チカ", "col3","sekine chika"));
 listTable.add(createValueMap("col1","寺田雅彦", "col2","テラダ　マサヒコ" , "col3","terada masahiko"));
 listTable.add(createValueMap("col1","吉川一哉", "col2","ヨシカワ　カズヤ", "col3","yoshikawa kazuya"));
 listTable.add(createValueMap("col1","大磯博文", "col2","オオイソ　ヒロフミ", "col3","ohiso hirohumi"));
 listTable.add(createValueMap("col1","萩本順子", "col2","ハギモト　ジュンコ", "col3","hagimoto jyunko"));
 listTable.add(createValueMap("col1","生田一哉", "col2","イクタ　カズヤ", "col3","ikuta kazuya"));
 listTable.add(createValueMap("col1","片山聡", "col2","カタヤマ　サトシ", "col3","katayama satoshi"));
%>

<!-- ヘッド情報 -->
<imui:head>
  <title>一覧画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
  <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
  <div class="imui-toolbar-inner">
```

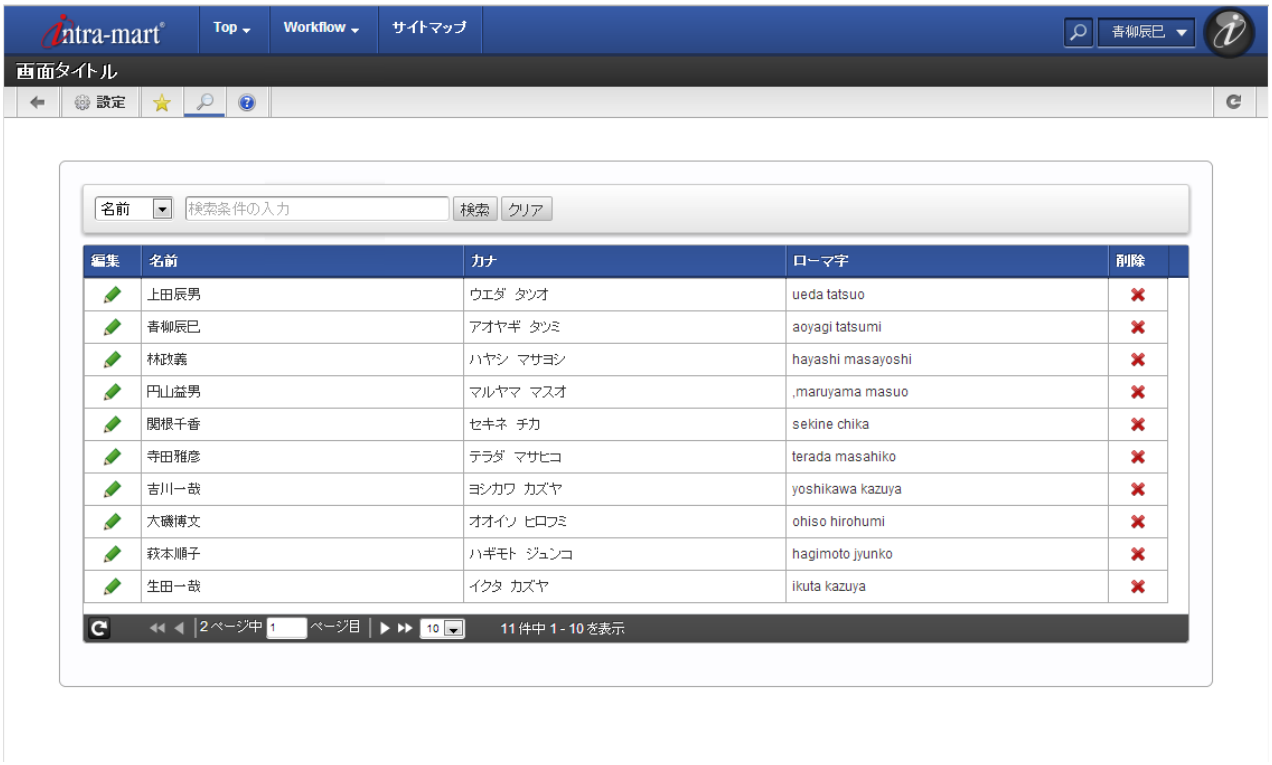
```
    <!-- ツールバー左側 -->
    <ul class="imui-list-toolbar">
      <!-- 戻る -->
      <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- 設定ボタン -->
      <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- タブ お気に入りボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
      <!-- タブ 検索ボタン -->
      <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
      <!-- タブ ヘルプボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
    </ul>
    <!-- ツールバー右側 -->
    <ul class="imui-list-box-toolbar-utility">
      <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
    </ul>
  </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
  <div class="imui-box-operation cf">
    <div class="pr-10 float-L">
      <imui:select id="search-list" list="<%= selectbox %>" />
    </div>
    <div class="float-L">
      <imui:textbox id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
      <imui:button value="検索" id="submit" class="imui-button" />
      <imui:button value="クリア" id="clear" class="imui-button" />
    </div>
  </div>
  <a id="on-delete-row" class="imui-unaccented" ><span class="im-ui-icon-common-16-trashbox mr-5"></span>チェックした
項目を削除</a>
  <imui:listTable data="<%= listTable %>" id="listtable"  multiSelect="true" viewRecords="true" height="300">
    <pager rowNum="10" rowList="1,5,10" />
    <cols>
      <col name="col1" caption="名前" />
      <col name="col2" caption="カナ" >
        <showLink baseLinkUrl="sample" showAction="/listTable"/>
      </col>
      <col name="col3" caption="ローマ字" />
    </cols>
  </imui:listTable>
</div>

<script>
  $(function(){
    $('#on-delete-row').on('click',function(){
      imuiConfirm('選択したデータを削除します。よろしいですか？','',function(){
      // 削除処理
      })
    })
    $('#submit').on('click',function(){
      // 検索ボタン押下時の処理
    })
    $('#clear').on('click',function(){
      // クリアボタン押下時の処理
    });
  })
</script>
```

実装サンプル 2

画面レイアウト


intra-mart
Top
Workflow
サイトマップ
青柳辰巳
画面タイトル
設定
名前
検索条件の入力
検索
クリア
編集
名前
カナ
ローマ字
削除
上田辰男
ウエダ タツオ
ueda tatsuo
青柳辰巳
アオヤギ タツミ
aoyagi tatsumi
林政義
ハヤシ マサヨシ
hayashi masayoshi
円山益男
マルヤマ マスオ
,maruyama masuo
関根千香
セキネ チカ
sekine chika
寺田雅彦
テラダ マサヒコ
terada masahiko
吉川一哉
ヨシカワ カズヤ
yoshikawa kazuya
大磯博文
オオイソ ヒロフミ
ohiso hirohumi
荻本順子
ハギモト ジュンコ
hagimoto jyunko
生田一哉
イクタ カズヤ
ikuta kazuya
2 ページ中 1 ページ目
10
11 件中 1 - 10 を表示

JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Map"%>
<%@ page import="java.util.HashMap"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<%!

List<Map<String, Object>> selectbox = new ArrayList<Map<String, Object>>();
{
    List<Map<String, Object>> selectboxOption = new ArrayList<Map<String, Object>>();
    Map<String, Object> optGrp = new HashMap<String, Object>();
    optGrp.put("type", "group");
    optGrp.put("label", "行検索");
    List<Map<String, Object>> optList = new ArrayList<Map<String, Object>>();
    Map<String, Object> option = new HashMap<String, Object>();
    option.put("label", "名前");
    option.put("value","1");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "カナ");
    option.put("value","2");
    optList.add(option);
    option = new HashMap<String, Object>();
    option.put("label", "ローマ字");
    option.put("value","3");
    optList.add(option);
    optGrp.put("list", optList);
    selectbox.add(optGrp);
}
Map<String, Object> createValueMap(Object... objArray) {
    Map<String, Object> map = new HashMap<String, Object>();

    int size = objArray.length;

    for (int i = 0; i < size; i++) {

        if ((i + 1) % 2 == 1) {
            map.put(objArray[i].toString(), objArray[i + 1]);
        }
    }

    return map;
}
%>
<%
 //リストテーブルデータ
  List<Map<String, Object>> listTable = new ArrayList<Map<String, Object>>();
  listTable.add(createValueMap("col0","","col1","上田辰男", "col2","ウエダ　タツオ", "col3","ueda tatsuo"));
  listTable.add(createValueMap("col0","","col1","青柳辰巳", "col2","アオヤギ　タツミ", "col3","aoyagi tatsumi"));
  listTable.add(createValueMap("col0","","col1","林政義", "col2","ハヤシ　マサヨシ", "col3","hayashi masayoshi"));
  listTable.add(createValueMap("col0","","col1","円山益男", "col2","マルヤマ　マスオ", "col3","maruyama masuo"));
  listTable.add(createValueMap("col0","","col1","関根千香", "col2","セキネ　チカ", "col3","sekine chika"));
  listTable.add(createValueMap("col0","","col1","寺田雅彦", "col2","テラダ　マサヒコ" , "col3","terada masahiko"));
  listTable.add(createValueMap("col0","","col1","吉川一哉", "col2","ヨシカワ　カズヤ", "col3","yoshikawa kazuya"));
  listTable.add(createValueMap("col0","","col1","大磯博文", "col2","オオイソ　ヒロフミ", "col3","ohiso hirohumi"));
  listTable.add(createValueMap("col0","","col1","萩本順子", "col2","ハギモト　ジュンコ", "col3","hagimoto jyunko"));
  listTable.add(createValueMap("col0","","col1","生田一哉", "col2","イクタ　カズヤ", "col3","ikuta kazuya"));
  listTable.add(createValueMap("col0","","col1","片山聡", "col2","カタヤマ　サトシ", "col3","katayama satoshi"));
%>

<!-- ヘッド情報 -->
<imui:head>
  <title>一覧画面</title>
</imui:head>

<!-- 画面タイトル -->
<div class="imui-title">
  <h1>一覧画面</h1>
</div>

<!-- 以下ツールバー -->
<div class="imui-toolbar-wrap">
  <div class="imui-toolbar-inner">
  <!-- ツールバー左側 -->
```

```
    <ul class="imui-list-toolbar">
      <!-- 戻る -->
      <li><a  class="imui-toolbar-icon" title="戻る"><span class="im-ui-icon-common-16-back"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- 設定ボタン -->
      <li><a class="imui-toolbar-icon imui-unaccented"><span class="im-ui-icon-common-16-settings mr-5"></span>設定</a>
</li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
      <!-- タブ お気に入りボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-star"></span></a></li>
      <!-- タブ 検索ボタン -->
      <li><a class="imui-list-toolbar-tab-selected"><span class="im-ui-icon-common-16-search"></span></a></li>
      <!-- タブ ヘルプボタン -->
      <li><a class="imui-toolbar-tab"><span class="im-ui-icon-common-16-question"></span></a></li>
      <!-- 区切り線 -->
      <li class="icon-split"></li>
    </ul>
    <!-- ツールバー右側 -->
    <ul class="imui-list-box-toolbar-utility">
      <li><a class="imui-toolbar-icon" title="最新情報"><span class="im-ui-icon-common-16-refresh"></span></a></li>
    </ul>
  </div>
</div>

<!-- コンテンツエリア -->
<div class="imui-form-container-wide">
  <div class="imui-box-operation cf">
    <div class="pr-10 float-L">
      <imui:select id="search-list" list="<%= selectbox %>" />
    </div>
    <div class="float-L">
      <imui:textbox id="search-text" maxlength="50" placeholder="検索条件の入力" style="width:230px"/>
      <imui:button value="検索" id="submit" class="imui-button" />
      <imui:button value="クリア" id="clear" class="imui-button" />
    </div>
  </div>
  <imui:listTable data="<%= listTable %>" id="listtable" viewRecords="true" onCellSelect="onCellSelect" height="300">
    <pager rowNum="10" rowList="1,5,10" />
    <cols>
      <col name="col0" caption="編集" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
        <showIcon iconClass="im-ui-icon-common-16-update" />
      </col>
      <col name="col1" caption="名前" />
      <col name="col2" caption="カナ" />
      <col name="col3" caption="ローマ字" />
      <col name="col0" caption="削除" sortable="false" width="25" align="center" onCellAttr="onCellAttr" >
        <showIcon iconClass="im-ui-icon-common-16-ng" />
      </col>
    </cols>
  </imui:listTable>
</div>

<script>
  $(function(){
    onCellSelect = function(rowid,iCol,cellcontent,e){
      if($(cellcontent).hasClass('im-ui-icon-common-16-ng')){
        imuiConfirm('削除します。よろしいですか？','',function(){
          // 削除処理
      })
      }
      if($(cellcontent).hasClass('im-ui-icon-common-16-update')){
        // 編集処理
      }
    }
    $('#submit').on('click',function(){
      // 検索ボタン押下時の処理
    })
    $('#clear').on('click',function(){
      // クリアボタン押下時の処理
    });
  })
  function onCellAttr(){
    return 'style="min-width:30px;"'
  }
</script>
```

画面遷移

項目

画面遷移がある場合（登録、更新など）の実装例

画面遷移がある場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajax でのデータ送信と画面遷移 – imuiAjaxSubmit

画面遷移あり JSP （ヘッド情報）

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui"%>

<imui:head>
  <title>新規登録画面</title>
  <!-- Load library -->
  <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
  <script type="text/javascript">

    var rules = {
      textbox: {
        required:true,
        maxlength:20
      },
      textarea: {
        required:true
      }
    };

    var messages = {
      textbox: {
        required:"タイトルは必須です。",
        maxlength:"タイトルは20文字以内で入力してください"
      },
      textarea: {
        required:"テキストエリアは必須です。"
      }
    };

    (function($) {
      $(document).ready(function() {
        // Form の 2 度押し防止
        imuiDisableOnSubmit('#form');

        // 参照画面へ引き渡すキーの配列生成
        var optionalData = ['user_cd'];

        // 登録ボタンクリック
        $('#register-button').click(function() {

          // バリデーションチェック
          if (imuiValidate('#form', rules, messages)) {
            // 確認ダイアログ表示
            imuiConfirm(
              '<c:out value="message" />', // メッセージ
              '<c:out value="title" />' ,  // タイトル
              function() { // OKクリック時のコールバック関数
                // Ajax でのデータ送信と次画面への遷移
                imuiAjaxSubmit('#form', 'POST', 'json', 'example/tgfw/reference/list');
              }
            );
          }
        });
      });
    })(jQuery);
  </script>
</imui:head>
```

画面遷移あり JSP （画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。

または、 *更新画面の実装サンプル* を参照してください。

> **コラム**
> imuiValidate を利用するときは、<input type=”submit”/> ではなく、<input type=”button”/> を利用してください。

画面遷移がない場合の実装例

画面遷移がない場合の概要と実装方法についての説明です。

- 処理の流れ
  - Form の 2 度押し防止 – imuiDisableOnSubmit （セレクタ）
  - バリデーションチェック – imuiValidate
  - 確認ダイアログ表示 – imuiConfirm
  - Ajaxでのデータ送信 – imuiAjaxSend

画面遷移なし JSP （ヘッダー／フッター）

```
<imui:head>
 <title>更新／削除画面</title>
 <!-- Load library -->
 <script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
 <script type="text/javascript">

  var rules = {
   textbox: {
    required:true,
    maxlength:20
   },
   textarea: {
    required:true
   }
  };

  var messages = {
   textbox: {
    required:"タイトルは必須です。",
    maxlength:"タイトルは20文字以内で入力してください"
   },
   textarea: {
    required:"テキストエリアは必須です。"
   }
  };

  (function($) {
   $(document).ready(function() {
    // Form の 2 度押し防止
    imuiDisableOnSubmit('#form');

    // 参照画面へ引き渡すキーの配列生成
    var optionalData = ['user_cd'];

    // 更新ボタンクリック
    $('#update-button').click(function() {
     // バリデーションチェック
     if (imuiValidate('#form', rules, messages)) {
      // 確認ダイアログ表示
      imuiConfirm(
       '<c:out value="message" />', // メッセージ
       '<c:out value="title" />' ,  // タイトル
       function() { // OKクリック時のコールバック関数
        // Ajaxでのデータ送信
        imuiAjaxSend('#form', 'POST', 'json');
       }
      );
     }
    });
   });
  })(jQuery);
 </script>
</imui:head>
```

画面遷移なし JSP （画面タイトル／ツールバー／コンテンツエリア）

登録画面の実装サンプル を参照してください。
または、 *更新画面の実装サンプル* を参照してください。

Ajax 通信の利用方法

imuiAjaxSend, imuiAjaxSubmit を使用して Ajax 通信をする場合、呼び出し先のページでは処理成功時のメッセージ、処理失敗時のメッセージ、処理成功後に遷移するページへ引き渡すパラメータ、を返すことができます。

属性名	説明	型	必須
error	処理が失敗した場合 true、成功した場合 false を指定します。	boolean	o
successMessage	処理成功時のメッセージ。error: false の場合表示されます。	String	–
errorMessage	処理失敗時のメッセージ。error: true の場合表示されます。	String	–
detailMessages	処理失敗時の詳細なメッセージ。error: true の場合表示されます。	String/String[]	–

属性名	説明	型	必須
parameter	処理成功後に遷移するページへ引き渡すパラメータ。	Object	-

parameter について

オブジェクトのキーを input タグの name 属性に、値を value 属性にセットして submit します。
値に配列を指定することが可能です。ただし、1 次元配列のみサポートします。

parameter の指定方法と、次画面での取得例は以下のようになります。

parameter の指定
Ajaxの処理を行うControllerの @RequestMapping メソッドで返り値のオブジェクトに parameter を設定します。

```
@RequestMapping(value = "register", method = RequestMethod.POST, produces = "application/json")
@ResponseBody
public Map<String, ? extends Object> ajax(RegisterForm form) {

    Map<String, Object> result = new HashMap<String, Object>();

    // 次の画面に渡すパラメータを設定します。
    Map<String, Object> parameters = new HashMap<String, Object>();
    List<String> arrayList = new ArrayList<String>();
    arrayList.add("element1");
    arrayList.add("element2");
    parameters.put("param1", "value1");
    parameters.put("param2", "value2");
    parameters.put("array1", arrayList);
    result.put("parameters", parameters);

    return result;
}
```

次画面での取得（Controllerクラスで取得する）

```
@Controller
@RequestMapping("example/tgfw/reference")
public class NextController {

    @RequestMapping("list")
    public String list(@ModelAttribute ParamForm form) {
        return "example/tgfw/reference/list.jsp";
    }
}
```

```
public class ParamForm {

    private String param1;

    private String param2;

    private List<String> array1;

    // getter, setter
    // ...
}
```

となります。

Ajax 実装例

コントローラークラスでは、@RequestMapping メソッドの実行結果をJSONとして応答するために、 メソッドに @ResponseBody を設定し、 メソッドの返り値をオブジェクトにします。

```java
@RequestMapping(value = "register", method = RequestMethod.POST, produces = "application/json")
@ResponseBody
public Map<String, ? extends Object> ajax(@Validated RegisterForm form, Errors errors) {

    Map<String, Object> result = new HashMap<String, Object>();
    // validationのエラー処理
    if (errors.hasErrors()) {
        // Errorsからメッセージを生成。
        // result.put("errorMessage", "エラーメッセージ");
        // result.put("detailMessages", {"詳細メッセージ1", "詳細メッセージ2", ... , "詳細メッセージn"});
        result.put("error", Boolean.TRUE);
        return result;
    }

    // 業務処理実行
    // ...

    // 正常終了
    result.put("error", Boolean.FALSE);
    // result.put("successMessage", "OK!");

    // 必要に応じて、次の画面に渡すパラメータを設定します。
    Map<String, Object> parameters = new HashMap<String, Object>();
    List<String> arrayList = new ArrayList<String>();
    arrayList.add("element1");
    arrayList.add("element2");
    parameters.put("param1", "value1");
    parameters.put("param2", "value2");
    parameters.put("array1", arrayList);
    result.put("parameters", parameters);

    return result;
}
```