intra-mart Accel Platform

IM-Workflow プログラミングガイド

2014/04/01 第 7 版

＜＜　変更履歴　＞＞

変更年月日	変更内容
2012/10/01	初版
2012/12/21	第 2 版 ● 「2.2 リクエストパラメータ」 の 「imwSerialProcParams」 に関する説明を追加・修正しました。 ● 「7.4.1 実装例」 にコールバック関数が受け取れる情報について説明を追加しました。 ● 「7.4.3 特記事項」、「7.4.3.1 IM-Workflowバージョン 8.0.2 における改善」 を追加しました。 ● 「7.5」 章の見出しを変更しました。また 「7.5」 章以下の章立てを見直し、説明を追加しました。 ➢ 変更前: 処理画面から受け取るリクエストパラメータ ➢ 変更後: 処理完了後の画面遷移 ● 「7.6 ユーザコンテンツと連続処理／連続確認の連携方法」 を追加しました。
2013/04/01	第 3 版 ● 「1.2 前提条件」 を修正しました。 ● 「2.2 リクエストパラメータ」 に、スマートフォン用画面の説明を追記しました。 ● 「3 画面の作成」 に、スマートフォン用画面の説明を追記しました。 ● 「3.5 制限事項」 に、章の説明を追記しました。 ● 「6.2.1 画面」 に、スマートフォン用画面の説明を追記しました。 ● 「7.1 呼び出し画面の初期表示値指定」 に、スマートフォン用画面の説明を追記しました。 ● 「7.3 画面入力情報の保持」 に、スマートフォン用画面の説明を追記しました。 ● 「7.4 呼び出し画面からのコールバック関数の指定」 に、スマートフォン用画面の説明を追記しました。 ● 「7.5 処理完了後の画面遷移」 に、スマートフォン用画面の説明を追記しました。 ● 「7.6 ユーザコンテンツと連続処理／連続確認の連携方法」 に、スマートフォン用画面の説明を追記しました。 ● 「7.7 PC版ユーザコンテンツをスマートフォン用画面としても利用する」 を追加しました。 ● 上記のほか、誤字脱字などを修正しました。
2013/07/01	第 4 版 ● 「7.1.2 実装例」 の実装サンプル記述を修正しました。 ● 「7.4.2 標準画面を非同期で実行する場合の注意点」を追加しました。
2013/10/01	第 5 版 ● 「2.2 リクエストパラメータ」 の記述を修正しました。 ➢ 「imwAuthUserCode」 に関するただし書きを削除しました。
2014/01/01	第 6 版 ● サンプル java ソースのプログラムパスを修正しました。 ● 「2.2 リクエストパラメータ」 の記述を修正しました。 ➢ 「imwGroupId」 が非推奨である旨を記述しました。 ● 「7.8 ユーザコンテンツ画面への不正な直接アクセスを抑止する」 を追加しました。
2014/04/01	第 7 版 ● 「2.3 案件処理系APIと画面動作仕様の違い」を追加しました。

＜＜　目次　＞＞

1 はじめに

1.1 目的

本書は、IM-Workflow で利用することが可能な画面およびモジュールを作成する方法について説明します。

本書は、IM-Workflow の機能を使用する方法を記述しています。

本書で使用するサンプルプログラムはあくまでも、IM-Workflow の機能および API 等の使用方法を理解することに主眼をおいています。そのため、必ずしもベストなコーディング方法とはいえない方法もあえて取っている個所があります。あくまでも、サンプルとしての位置付けでとらえるようにしてください。

1.2 前提条件

- 本書に記述されているサンプルプログラムは、JavaEE 開発モデルおよびスクリプト開発モデルで記述されています。そのため、JavaEE 開発モデルおよびスクリプト開発モデルに関する理解は必須です。各開発モデルに関しては、付属する各種マニュアルおよび API リストを参照してください。
- 本書を理解するには、基本的な IM-Workflow に関する理解が必要になります。付属する各種マニュアル、API リスト、および制限事項を参照してください。
- 本書に記述されているサンプルプログラムのパスは、以下のディレクトリ配下のパスです。
  <（展開した war）/WEB-INF/ >

1.3 準備

IM-Workflow のサンプルプログラムを実行するための準備をします。
「intra-mart Accel Platform / セットアップガイド」を参考に、IM-Workflow が動作する環境を構築します。

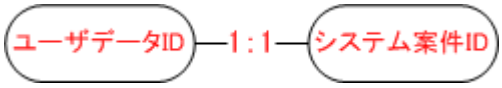
製品のインストール後は、システム管理者でログインし、メニュー[テナント環境セットアップ]より、テナント環境セットアップを行い、サンプルデータセットアップも必ず行ってください。

本書に記述されている JavaEE 開発モデルの[java ファイル]は、配置する場所を示します。
実際に配置されているファイルは、[class ファイル]です。
JavaEE 開発モデル[java ファイル]のサンプルプログラムについては、製品メディアに保存されています。
また、製品最新情報ダウンロードページ（http://www.intra-mart.jp/download/product/index.html）から入手することもできます。

2 概要

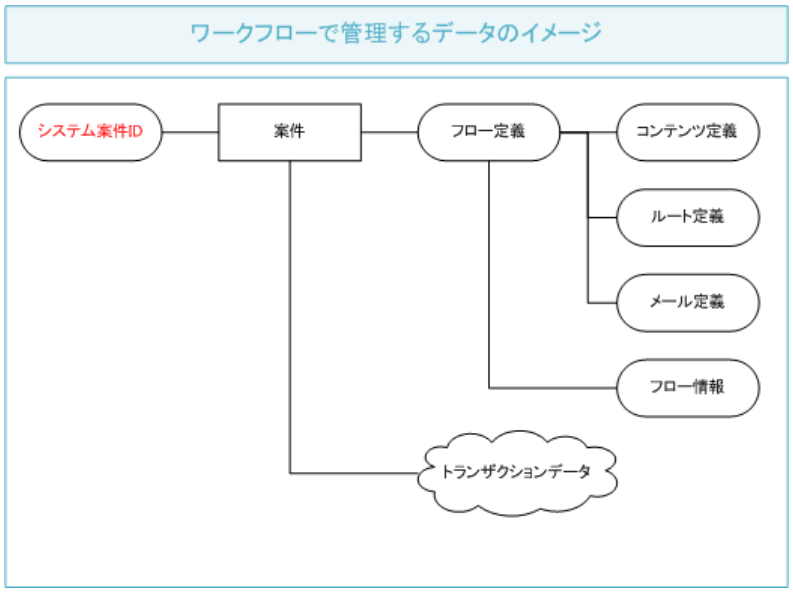
2.1 ユーザアプリケーションデータとIM-Workflowの関係

ユーザアプリケーションデータとIM-Workflowのデータは、それぞれ”ユーザデータID“と“システム案件ID”という2つのキーによって一意に特定されます。2つのキーは1対1の関係で関連付けられます。



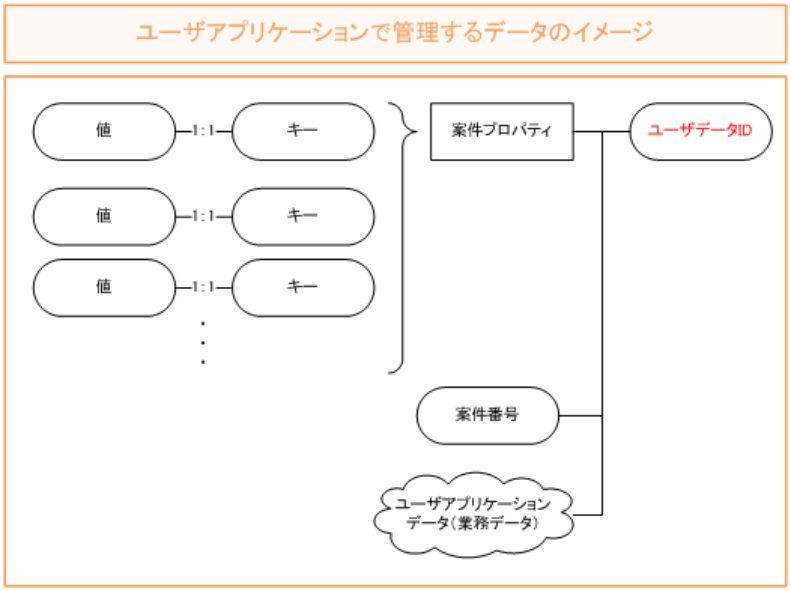
2.1.1 システム案件ID

システム案件IDとは、IM-Workflowにおいて一意となるキーです。
IM-Workflowのモジュールにおいて採番され、外部より指定することはできません。
システム案件IDは、IM-WorkflowのAPIやタグライブラリ等で案件を特定するために使用され、画面等に表示されることはありません。



2.1.2 ユーザデータID

ユーザデータ ID とは、ユーザアプリケーション側で一意となるキーとして、ユーザアプリケーションで採番するキーです。
申請または起票を行う場合に、IM-Workflow の提供する API およびタグライブラリの引数として渡されます。
ユーザデータ ID は、システム案件 ID と同様に、IM-Workflow の API やタグライブラリ等で案件を特定するために使用され、画面等に表示されることはありません。



2.1.3 案件プロパティ

案件プロパティとは、いわゆる Key-Value Store です。「Key（キー）」と「Value（値）」のペアからなるデータモデルを案件単位に IM-Workflow で保存します。

IM-Workflow が提供する API を通じて、任意のタイミングにおいて、登録・更新・削除および取得が可能となります。
また、IM-Workflow が提供する各種一覧画面に表示したり、分岐条件におけるルール定義で参照する値として使用することができます。

2.2 リクエストパラメータ

各種一覧画面から呼び出される申請および処理等の画面で、必要な情報をリクエストパラメータとして受け取る事ができます。

No	パラメータ(物理名)	パラメータ(論理名)	詳細
1	imwGroupId	グループ ID	非推奨です。 過去との互換のために残されています。
2	imwUserCode	処理者 CD	
3	imwPageType	画面種別	表示された画面の種別 ・申請画面 ・一時保存画面 ・申請(起票案件)画面 ・再申請画面 ・処理画面 ・確認画面 ・処理詳細 ・参照詳細 ・確認詳細 ・過去案件詳細 ・申請画面(スマートフォン用) ・一時保存画面(スマートフォン用) ・申請(起票案件)画面(スマートフォン用) ・再申請画面(スマートフォン用) ・処理画面(スマートフォン用) ・確認画面(スマートフォン用)
4	imwUserDataId	ユーザデータ ID	
5	imwSystemMatterId	システム案件 ID	
6	imwNodeId	ノード ID	
7	imwArriveType	到達種別	
8	imwAuthUserCode	権限者 CD	ログインユーザが案件を処理する際に選択可能な権限者 CD です。具体的には、ログインユーザ本人や、ログインユーザを代理先として代理設定されている場合は代理元ユーザ CD が該当します。 権限者が複数存在する場合、当パラメータは配列で渡されます。※ ただし、権限者が複数存在する場合でも、申請/一時保存画面表示の際は一覧上で権限者が特定されているため、特定済みの権限者 CD のみが渡されます。
9	imwApplyBaseDate	申請基準日	「yyyy/mm/dd」形式

10	imwContentsId	コンテンツ ID	
11	imwContentsVersionId	コンテンツバージョン ID	
12	imwRouteId	ルート ID	
13	imwRouteVersionId	ルートバージョン ID	
14	imwFlowId	フローID	
15	imwFlowVersionId	フローバージョン ID	
16	imwSerialProcParams	連続処理パラメータ	連続処理用のパラメータ IM-Workflow バージョン 8.0.2 より、当パラメータは無効になりました。 必ず空文字("")が渡されるため、ユーザコンテンツ間での当パラメータの引き回しは不要です。 連続処理用の情報は「imwCallOriginalParams」に内包されます。
17	imwCallOriginalParams	呼出元パラメータ	呼出元ページのパラメータ
18	imwCallOriginalPagePath	呼出元ページパス	呼出元のページパス

※imwAuthUserCode(権限者 CD)について、各開発モデルでの取得例を以下に記述します。

ここで記載している内容は、次の観点において共通です。

■ クライアントタイプ

スクリプト開発モデル

```
function init(request) {
    var imwAuthUserCodeList = request.getParameterValues("imwAuthUserCode"); //権限者 CD の配列
}
```

javaEE 開発モデル

```
HttpServletRequest request = getRequest();
String[] imwAuthUserCodeList = request.getParameterValues("imwAuthUserCode"); //権限者 CD の配列
```

No	パラメータ	申請※	一時保存※	起票※	再申請※	処理※	確認※	処理詳細	参照詳細	確認詳細	過去案件詳細
1	imwGroupId	○	○	○	○	○	○	○	○	○	○
2	imwUserCode	○	○	○	○	○	○	○	○	○	○
3	imwPageType	○	○	○	○	○	○	○	○	○	○
4	imwUserDataId	-	○	○	○	○	○	○	○	○	○
5	imwSystemMatterId	-	-	○	○	○	○	○	○	○	○
6	imwNodeId	○	○	○	○	○	○	-	-	-	-
7	imwArriveType	○	○	○	○	○	-	-	-	-	-
8	imwAuthUserCode	○	○	○	○	○	-	-	-	-	-
9	imwApplyBaseDate	○	○	○	○	○	○	○	○	○	○
10	imwFlowId	○	○	○	○	○	○	○	○	○	○
11	imwFlowVersionId	○	○	○	○	○	○	○	○	○	○
12	imwContentsId	○	○	○	○	○	○	○	○	○	○
13	imwContentsVersionId	○	○	○	○	○	○	○	○	○	○
14	imwRouteId	○	○	○	○	○	○	○	○	○	○
15	imwRouteVersionId	○	○	○	○	○	○	○	○	○	○
16	imwSerialProcParams	-	-	~~○~~ -	~~○~~ -	~~○~~ -	~~○~~ -	-	-	-	-
17	imwCallOriginalParams	○	○	○	○	○	○	-	-	-	-
18	imwCallOriginalPagePath	○	○	○	○	○	○	-	-	-	-

＜「○」：取得可能 ／「-」：取得不可能＞

※ スマートフォン用の場合も同様です。
~~ただし、「imwAuthUserCode」のみ、スマートフォン用の起票、再申請、処理画面では取得することができません。~~

2.3 案件処理系APIと画面動作仕様の違い

画面上からの操作とは異なり、案件処理系 API（Web サービスを含む）を直接利用して案件の処理を行う場合は、業務的なチェックを行わずに処理が実行されます。
ここでいう業務的なチェックとは、以下のようなチェックを指します。

- API 引数として指定した処理権限者が、到達処理で展開された処理対象者に含まれるか
- API 引数として指定した処理権限者と処理実行者が異なる場合、両者間で有効な代理設定が存在するか

画面上からの操作と同等の機能を、API を利用して独自に実装する場合は、案件処理 API の実行前に各種 API（処理権限判定 API など）を併せて利用してください。

3 画面の作成

この章では、IM-Workflow が提供する案件の各処理画面と連携するための画面実装の基本部分について、「画面種別」、「開発モデル」、「クライアントタイプ」の観点で説明します。

上記観点の内訳は、次のとおりです。

- 画面種別
  - 申請画面
  - 一時保存画面
  - 申請(起票案件)画面
  - 再申請画面
  - 処理画面
  - 確認画面
  - 処理詳細
  - 参照詳細
  - 確認詳細
  - 過去案件詳細
- 開発モデル
  - スクリプト開発モデル
  - javaEE 開発モデル
- クライアントタイプ
  - PC
  - スマートフォン

また、画面作成における応用実装について、「7 カスタマイズ」 で説明しています。
必要に応じて参照してください。

3.1 申請画面の呼び出し

IM-Workflow で提供する申請を行うための画面（以下、申請画面）と連携する方法を説明します。

申請画面を表示するためには、IM-Workflow が提供するタグライブラリおよび Client-side JavaScript API を使用します。

3.1.1 スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.1.1.1 PC用画面の場合

申請画面と連携する画面のヘッダ部(<imart type="head"> ～ </imart>)に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

申請画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ファンクション・コンテナで採番する必要があります。

```
<imart type="workflowOpenPage"
        name="applyForm"
        id="applyForm"
        method="POST"
        target="_top "
        imwUserDataId=oRequest.imwUserDataId
        imwAuthUserCode=oRequest.imwAuthUserCode
        imwApplyBaseDate=oRequest.imwApplyBaseDate
        imwNodeId=oRequest.imwNodeId
        imwFlowId=oRequest.imwFlowId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage('0');

</script>
```

3.1.1.2 スマートフォン用画面の場合

申請画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

申請画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ファンクション・コンテナで採番する必要があります。

```
<imart type="spWorkflowOpenPage"
        name="applyForm"
        id="workflowOpenPageForm"
        method="POST"
        target="_top"
        imwUserDataId=$data.imwUserDataId
        imwAuthUserCode=$data.imwAuthUserCode
        imwApplyBaseDate=$data.imwApplyBaseDate
        imwNodeId=$data.imwNodeId
        imwFlowId=$data.imwFlowId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('10');

</script>
```

3.1.2 JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.1.2.1 PC用画面の場合

申請画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

申請画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ServiceController などで採番する必要があります。

```
<workflow:workflowOpenPage
		name="applyForm"
		id="applyForm"
		method="POST"
		target="_top "
		imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
		imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
		imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
		imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
		imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</workflow:workflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage( '0' );

</script>
```

3.1.2.2 スマートフォン用画面の場合

申請画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

申請画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。
「imwUserDataId」は、申請一覧画面からのリクエストパラメータには含まれません。
ServiceController などで採番する必要があります。

```
<workflow:spWorkflowOpenPage
        name="applyForm"
        id="applyForm"
        method="POST"
        target="_top "
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</ workflow:spWorkflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、申請画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('10');

</script>
```

3.2 一時保存画面の呼び出し

IM-Workflow で提供する一時保存を行うための画面（以下、一時保存画面）と連携する方法を説明します。

一時保存画面を表示するためには、IM-Workflow が提供するタグライブラリおよび Client-side JavaScript API を使用します。

3.2.1 スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.2.1.1 PC用画面の場合

一時保存画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

一時保存画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top "
        imwUserDataId=oRequest.imwUserDataId
        imwAuthUserCode=oRequest.imwAuthUserCode
        imwApplyBaseDate=oRequest.imwApplyBaseDate
        imwNodeId=oRequest.imwNodeId
        imwFlowId=oRequest.imwFlowId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage('1');

</script>
```

3.2.1.2 スマートフォン用画面の場合

一時保存画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

一時保存画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top "
        imwUserDataId=$data.imwUserDataId
        imwAuthUserCode=$data.imwAuthUserCode
        imwApplyBaseDate=$data.imwApplyBaseDate
        imwNodeId=$data.imwNodeId
        imwFlowId=$data.imwFlowId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('11');

</script>
```

3.2.2 JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.2.2.1 PC用画面の場合

一時保存画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

一時保存画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top "
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</workflow:workflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage('1');

</script>
```

3.2.2.2 スマートフォン用画面の場合

一時保存画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

一時保存画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常申請一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:spWorkflowOpenPage
        name="tempForm"
        id="tempForm"
        method="POST"
        target="_top "
        imwUserDataId='<%=(String)request.getAttribute("imwUserDataId")%>'
        imwAuthUserCode='<%=(String)request.getAttribute("imwAuthUserCode")%>'
        imwApplyBaseDate='<%=(String)request.getAttribute("imwApplyBaseDate")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'
        imwFlowId='<%=(String)request.getAttribute("imwFlowId")%>'>
</ workflow:spWorkflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、一時保存画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('11');

</script>
```

3.3 申請(起票案件)／再申請／処理画面の呼び出し

IM-Workflow で提供する申請(起票案件)／再申請／処理を行うための画面(以下、処理画面)と連携する方法を説明します。

処理画面を表示するためには、IM-Workflow が提供するタグライブラリおよび Client-side JavaScript API を使用します。

3.3.1 スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.3.1.1 PC用画面の場合

処理画面と連携する画面のヘッダ部(<imart type="head"> ～ </imart>)に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

処理画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top "
        imwSystemMatterId=$data.imwSystemMatterId
        imwNodeId=$data.imwNodeId >
</imart>
```

下記の Client-side JavaScript API を実行することにより、処理画面が表示されます。

■ 申請(起票案件)

```
<script type="text/javascript">

workflowOpenPage('2');

</script>
```

■ 再申請

```
<script type="text/javascript">

workflowOpenPage('3');

</script>
```

■ 処理

```
<script type="text/javascript">

workflowOpenPage('4');

</script>
```

3.3.1.2 スマートフォン用画面の場合

処理画面と連携する画面のヘッダ部(<imart type="head"> ～ </imart>)に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

処理画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top "
        imwSystemMatterId=$data.imwSystemMatterId
        imwNodeId=$data.imwNodeId >
</imart>
```

下記の Client-side JavaScript API を実行することにより、処理画面が表示されます。

■ 申請(起票案件)

```
<script type="text/javascript">

workflowOpenPage4Sp('12');

</script>
```

■ 再申請

```
<script type="text/javascript">

workflowOpenPage4Sp('13');

</script>
```

■ 処理

```
<script type="text/javascript">

workflowOpenPage4Sp('14');

</script>
```

3.3.2 JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.3.2.1 PC用画面の場合

処理画面と連携する画面のヘッダ部(<imui:head> ～ </imui:head>)に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

処理画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top "
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:workflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、処理画面が表示されます。

■ 申請(起票案件)

```
<script type="text/javascript">

workflowOpenPage('2');

</script>
```

■ 再申請

```
<script type="text/javascript">

workflowOpenPage('3');

</script>
```

■ 処理

```
<script type="text/javascript">

workflowOpenPage('4');

</script>
```

3.3.2.2 スマートフォン用画面の場合

処理画面と連携する画面のヘッダ部(<imui:head> ～ </imui:head>)に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

処理画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常未処理一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:spWorkflowOpenPage
        name="approveForm"
        id="approveForm"
        method="POST"
        target="_top "
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:spWorkflowOpenPage >
```

下記の Client-side JavaScript API を実行することにより、処理画面が表示されます。

■ 申請(起票案件)

```
<script type="text/javascript">

workflowOpenPage4Sp('12');

</script>
```

■ 再申請

```
<script type="text/javascript">

workflowOpenPage4Sp('13');

</script>
```

■ 処理

```
<script type="text/javascript">

workflowOpenPage4Sp('14');

</script>
```

3.4 確認画面の呼び出し

IM-Workflow で提供する確認を行うための画面（以下、確認画面）と連携する方法を説明します。

確認画面を表示するためには、IM-Workflow が提供するタグライブラリおよび Client-side JavaScript API を使用します。

3.4.1 スクリプト開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.4.1.1 PC用画面の場合

確認画面と連携する画面のヘッダ部（<imart type="head"> ～ </imart>）に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />

</imart>
```

確認画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="workflowOpenPage"
        name="confirmForm"
        id="confirmForm"
        method="POST"
        target="_top "
        imwSystemMatterId=$data.imwSystemMatterId
        imwNodeId=$data.imwNodeId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage('5');

</script>
```

3.4.1.2 スマートフォン用画面の場合

確認画面と連携する画面のヘッダ部(<imart type="head"> ～ </imart>)に、下記の IMART タグを記述します。

```
<imart type="head">

<imart type="spWorkflowOpenPageCsjs" />

</imart>
```

確認画面と連携する画面のボディ部に、下記の IMART タグを記述します。

IMART タグに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<imart type="spWorkflowOpenPage"
        name="confirmForm"
        id="confirmForm"
        method="POST"
        target="_top "
        imwSystemMatterId=$data.imwSystemMatterId
        imwNodeId=$data.imwNodeId>
</imart>
```

下記の Client-side JavaScript API を実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp('15');

</script>
```

3.4.2 JavaEE開発モデル

IM-Workflow 用のタグライブラリの使用方法については、API リストも併せて参照してください。

3.4.2.1 PC用画面の場合

確認画面と連携する画面のヘッダ部（<imui:head> ～ </imui:head>）に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:workflowOpenPageCsjs />

</imui:head>
```

確認画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<workflow:workflowOpenPage
        name="confirmForm"
        id="confirmForm"
        method="POST"
        target="_top "
        imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
        imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:workflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage( '5' );

</script>
```

3.4.2.2 スマートフォン用画面の場合

確認画面と連携する画面のヘッダ部(<imui:head> ～ </imui:head>)に、下記のタグライブラリを記述します。

```
<imui:head>

<workflow:spWorkflowOpenPageCsjs />

</imui:head>
```

確認画面と連携する画面のボディ部に、下記のタグライブラリを記述します。

タグライブラリに指定する属性は、通常確認一覧画面から取得したリクエストパラメータを指定します。

```
<workflow: spWorkflowOpenPage
            name="confirmForm"
            id="confirmForm"
            method="POST"
            target="_top "
            imwSystemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
            imwNodeId='<%=(String)request.getAttribute("imwNodeId")%>'>
</workflow:spWorkflowOpenPage>
```

下記の Client-side JavaScript API を実行することにより、確認画面が表示されます。

```
<script type="text/javascript">

workflowOpenPage4Sp( '15' );

</script>
```

3.5 制限事項

画面の作成における制限事項を説明します。
ここに記載のない制限事項については、「intra-mart Accel Platform / リリースノート」を参照してください。

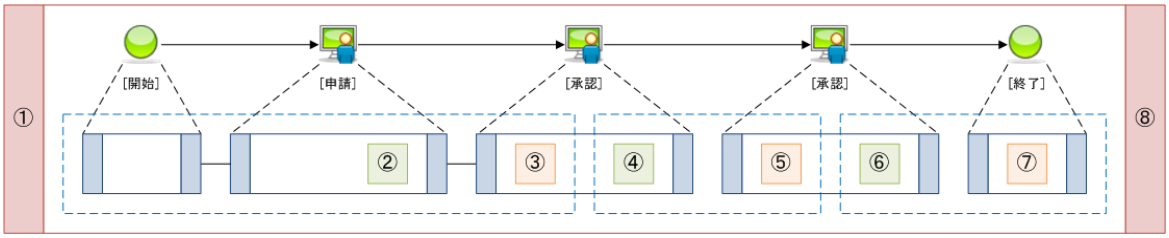
3.5.1 imwプレフィックスのパラメータについて

「workflowOpenPage」タグ、および「spWorkflowOpenPage」タグでは、ワークフロー処理時の制御用に imw プレフィックスの hidden タグを複数出力します。

パラメータ名が重複すると処理が正常に行われない恐れがあるため、以下で明示的に記載を許可されたもの以外、imw プレフィックス名称のパラメータは記述しないでください。

- 7.1 呼び出し画面の初期表示値指定
- 7.3 画面入力情報の保持

4 ユーザプログラムの作成



No	処理名	項番
1	案件開始処理	①
2	案件終了処理	⑧
3	アクション処理	② ④ ⑥
4	到達処理	③ ⑤ ⑦

4.1 案件開始処理

案件開始処理とは、案件が開始する時に、一度実行される処理です。
下記の場合に実行されます。

- 申請者が”申請”を行った場合
- ”起票”の案件を作成した場合（API のみ）

案件開始処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中では DB トランザクション制御を行うことはできません。

4.2 案件終了処理

案件終了処理とは、案件が終了する時に、一度実行される処理です。
下記の場合に実行されます。

- 最後の承認者が”承認”を行った場合
- 承認者が”承認終了”を行った場合
- 承認者が“否認”を行った場合
- 申請者が“取止め”を行った場合
- 案件操作で終了ノードに到達した場合

案件終了処理は、直前のアクション処理や到達処理とは独立した処理（トランザクション）となります。そのため、案件終了処理でエラーが発生した場合、直前の処理を戻す（ロールバック）することはできません。

案件終了処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中では DB トランザクション制御を行うことはできません。

4.3 アクション処理

アクション処理とは、下記のような行為を行った場合に実行される処理です。

No	アクション	メソッド
1	申請	apply
2	再申請	reapply
3	申請（一時保存）	applyFromTempSave
4	申請（未処理）	applyFromUnapply
5	取止め	discontinue
6	引戻し	pullBack
7	差戻し後引戻し	sendBackToPullBack
8	承認	approve
9	承認終了	approveEnd
10	否認	deny
11	差戻し	sendBack
12	保留	reserve
13	保留解除	reserveCancel
14	案件操作	matterHandle
15	一時保存（新規登録）	tempSaveCreate
16	一時保存（更新）	tempSaveUpdate
17	一時保存（削除）	tempSaveDelete

アクション処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中では DB トランザクション制御を行うことはできません。

4.4 到達処理

到達処理とは、ノードに到達した場合に実行される処理です。
この処理は、アクション処理や IM-Workflow の内部処理とは独立した処理（thread）として実行されます。
そのため、到達処理でエラーが発生した場合、直前の処理を戻す（ロールバック）することはできません。
（直前のアクション処理とは、トランザクションも別となります。）

このプログラム中で、データベースの登録／更新／削除処理を行う場合は、独自に DB トランザクション制御を行ってください。

下記のような場合に実行されます。

- 前ノードの処理者が、”申請”または”承認”を行って到達した場合
- 他のノードから、“差戻し“され到達した場合
- “引戻し”を行って到達した場合
- 案件操作で到達した場合

4.5 分岐開始処理

分岐開始処理とは、分岐開始ノードで「ユーザプログラムで分岐する」を選択した場合に、実行される処理です。
分岐先ノード毎に順番に実行されます。

分岐開始処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中では DB トランザクション制御を行うことはできません。

分岐開始処理において、ルート遷移可否として 遷移する（true） を返却することにより、実行中の分岐開始処理が設定された分岐先ノードに進みます。
全ての分岐開始処理のルート遷移可否が 遷移しない（false） の場合は、案件は分岐開始ノードで停止します。
このような場合は、案件操作処理で案件を進めて下さい。

4.6 分岐終了処理

分岐終了処理とは、分岐終了ノードで「ユーザプログラムで分岐終了する」を選択した場合に、実行される処理です。
分岐終了ノードに案件が到達する度に実行されます。

分岐終了処理は、IM-Workflow モジュールのトランザクション内で実行されるため、このプログラム中では DB トランザクション制御を行うことはできません。

分岐終了処理において、ルート遷移可否として 結合する（true） を返却することにより、未到達のノードを待たずに次のノードに進みます。
全てのノードが到達しても結果が全て 結合しない（false） の場合は、案件は分岐終了ノードで停止します。
このような場合は、案件操作処理で案件を進めて下さい。

5 その他プログラムの作成

5.1 未完了案件削除処理リスナー

未完了案件削除処理リスナーとは、未完了案件を削除した際に実行されるプログラムです。
通常、「案件操作」画面より“案件削除“を行った場合、または未完了案件を削除する API を実行した際に呼び出されます。

未完了案件削除処理リスナーは、通常「**コンテンツ定義**」に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

%PUBLIC_STORAGE_PATH%/im_workflow/conf/param/param_group_%テナント ID%.xml

```
<!--
    未完了案件削除リスナーの種類
        [java] or [script] or []（指定なし）
        []（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
    <param-name>delete-active-matter-type</param-name>
    <param-value></param-value>
</param>
<!--
    未完了案件削除リスナーのパス
    1. 案件削除リスナーの種類が java：パッケージ名
    2. 案件削除リスナーの種類が script：WEB-INF/jssp からのパス
-->
<param>
    <param-name>delete-active-matter-listener-path</param-name>
    <param-value></param-value>
</param>
```

※[ワークフローパラメータ]]画面からも設定することが可能です。

設定方法の詳細については、「IM-Workflow 管理者操作ガイド」または「IM-Workflow 仕様書」を参照してください。

5.2 完了案件削除処理リスナー

完了案件削除処理リスナーとは、完了案件を削除した際に実行されるプログラムです。
通常、「参照」画面の完了案件タブより案件の“削除“を行った場合、または完了案件を削除する API を実行した際に呼び出されます。

完了案件削除処理リスナーは、通常「**コンテンツ定義**」に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

%PUBLIC_STORAGE_PATH %/im_workflow/conf/param/param_group_%テナント ID%.xml

```
<!--
    完了案件削除リスナーの種類
        [java] or [script] or []（指定なし）
        []（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
    <param-name>delete-complete-matter-listener-type</param-name>
    <param-value></param-value>
</param>
<!--
    完了案件削除リスナーのパス
    1. 案件削除リスナーの種類が java：パッケージ名
    2. 案件削除リスナーの種類が script：WEB-INF/jssp　からのパス
-->
<param>
    <param-name>delete-complete-matter-listener-path</param-name>
    <param-value></param-value>
</param>
```

※[ワークフローパラメータ]]画面からも設定することが可能です。

設定方法の詳細については、「IM-Workflow　管理者操作ガイド」または「IM-Workflow　仕様書」を参照してください。

5.3 過去案件削除処理リスナー

過去案件削除処理リスナーとは、過去案件を削除した際に実行されるプログラムです。

過去案件削除処理リスナーは、通常「**コンテンツ定義**」に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

%PUBLIC_STORAGE_PATH %/im_workflow/conf/param/param_group_%テナント ID%.xml

```
----------------------------------------------------------------------------------------------------------------------

<!--
	過去案件削除リスナーの種類
		[java] or [script] or []（指定なし）
		[]（指定なし）を設定した場合はリスナーを起動しない
-->
<param>
	<param-name>delete-archive-matter-listener-type</param-name>
	<param-value></param-value>
</param>
<!--
	過去案件削除リスナーのパス
	1. 案件削除リスナーの種類が java：パッケージ名
	2. 案件削除リスナーの種類が script：WEB-INF/jssp からのパス
-->
<param>
	<param-name>delete-archive-matter-listener-path</param-name>
	<param-value></param-value>
</param>
```

※[ワークフローパラメータ]]画面からも設定することが可能です。

設定方法の詳細については、「IM-Workflow 管理者操作ガイド」または「IM-Workflow 仕様書」を参照してください。

5.4 案件退避処理リスナー

案件退避処理リスナーとは、案件を退避した際に実行されるプログラムです。
通常、ジョブ「IM-Workflow/アーカイブ」を実行した際に呼び出されます。

案件退避処理リスナーは、通常「**コンテンツ定義**」に設定します。

また、テナント単位で処理を行う場合は、下記のファイルに設定します。

%PUBLIC_STORAGE_PATH %/im_workflow/conf/param/param_group_%テナント ID%.xml

```
----------------------------------------------------------------------------------------------------------------

<!--
    案件退避リスナーの種類
        [java] or [script] or []（指定なし）
        []（指定なし）を設定した場合はリスナを起動しない
-->
<param>
    <param-name>archive-proc-listener-type</param-name>
    <param-value>java</param-value>
</param>
<!--
    案件退避リスナーのパス
    1. 案件退避リスナーの種類が java：パッケージ名
    2. 案件退避リスナーの種類が script：WEB-INF/jssp からのパス
-->
<param>
    <param-name>archive-proc-listener-path</param-name>
    <param-value></param-value>
</param>
```

※[ワークフローパラメータ]]画面からも設定することが可能です。

設定方法の詳細については、「IM-Workflow　管理者操作ガイド」または「IM-Workflow　仕様書」を参照してください。

6 Appendix

6.1 テンプレート

ユーザプログラムおよび各リスナーのプログラムを作成する際のテンプレートが提供されています。

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/template/>

No	処理	物理名
1	案件開始処理	MatterStartProcess.js
2	案件終了処理	MatterEndProcess.js
3	アクション処理	ActionProcess.js
4	到達処理	ArriveProcess.js
5	分岐開始処理／分岐終了処理	RuleCondition.js
6	未完了案件削除処理リスナー	WorkflowActvMatterDeleteListener.js
7	完了案件削除処理リスナー	WorkflowCplMatterDeleteListener.js
8	過去案件削除処理リスナー	WorkflowArcMatterDeleteListener.js
9	案件退避処理リスナー	WorkflowMatterArchiveListener.js
10	処理対象者プラグイン	WorkflowAuthorityExecEventListener.js

■ JavaEE 開発モデル

JavaEE 開発モデル[java ファイル]のサンプルプログラムについては、製品メディアに保存されています。
また、製品最新情報ダウンロードページ（http://www.intra-mart.jp/download/product/index.html）から入手することもできます。

<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/workflow/template/>

No	処理	物理名
1	案件開始処理	MatterStartProcess.java
2	案件終了処理	MatterEndProcess.java
3	アクション処理	ActionProcess.java
4	到達処理	ArriveProcess.java
5	分岐開始処理／分岐終了処理	RuleCondition.java
6	未完了案件削除処理リスナー	WorkflowActvMatterDeleteListener.java
7	完了案件削除処理リスナー	WorkflowCplMatterDeleteListener.java
8	過去案件削除処理リスナー	WorkflowArcMatterDeleteListener.java
9	案件退避処理リスナー	WorkflowMatterArchiveListener.java
10	処理対象者プラグイン	WorkflowAuthorityExecEventListener.java
11	クローラ登録文書追加リスナー	WorkflowCrawlingAddListener.java

6.2 サンプルプログラム

IM-Workflow のインストール時”サンプルデータセットアップ”を行い、サンプルデータをインポートした場合に使用できるサンプルプログラムについて説明します。

サンプルプログラムは、スクリプト開発モデルと JavaEE 開発モデルのサンプルプログラムがあります。
開発モデルの違いはありますが、どちらのサンプルも「物品購買」の申請書であり、動作仕様は同一です。

6.2.1 画面

6.2.1.1 申請／一時保存／申請（起票案件）／再申請画面

PC 用画面とスマートフォン用画面について説明します。

6.2.1.1.1 PC用画面

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply.html>
<./jssp/src/sample/im_workflow/purchase/screen/apply.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------------


  <service>
    <service-id>apply</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceController</co
ntroller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceTransition</tra
nsition-class>
    <next-page>
      <page-path>/sample/im_workflow/purchase/apply.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.1.2 **スマートフォン用画面**

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/apply.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/apply.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>apply</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceController</co
ntroller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApplyServiceTransition</tra
nsition-class>
    <next-page>
      <page-path>/sample/im_workflow_smartphone/purchase/apply.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.2 処理画面

PC 用画面とスマートフォン用画面について説明します。

6.2.1.2.1 PC用画面

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/approve.html>
<./jssp/src/sample/im_workflow/purchase/screen/approve.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>approve</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceController</
controller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceTransition</t
ransition-class>
    <next-page>
      <page-path>/sample/im_workflow/purchase/approve.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.2.2 **スマートフォン用画面**

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/approve.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/approve.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>approve</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceController</
controller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ApproveServiceTransition</t
ransition-class>
    <next-page>
      <page-path>/sample/im_workflow_smartphone/purchase/approve.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.3 確認画面

PC 用画面とスマートフォン用画面について説明します。

6.2.1.3.1 PC用画面

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/confirm.html>
<./jssp/src/sample/im_workflow/purchase/screen/confirm.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>confirm</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceController</c
ontroller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceTransition</t
ransition-class>
    <next-page>
      <page-path>/sample/im_workflow/purchase/confirm.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.3.2 **スマートフォン用画面**

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/confirm.html>
<./jssp/src/sample/im_workflow_smartphone/purchase/screen/confirm.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow_smartphone-8.0.x-sample.jar/service-config-imw_sp_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------

  <service>
    <service-id>confirm</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceController</c
ontroller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.ConfirmServiceTransition</t
ransition-class>
    <next-page>
      <page-path>/sample/im_workflow_smartphone/purchase/confirm.jsp</page-path>
    </next-page>
  </service>
```

6.2.1.4 処理詳細／参照詳細／過去案件詳細／確認詳細画面

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/detail.html>
<./jssp/src/sample/im_workflow/purchase/screen/detail.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow-8.0.x-sample.jar/service-config-imw_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------

  <service>
    <service-id>detail</service-id>

<controller-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.DetailServiceController</controller-class>

<transition-class>jp.co.intra_mart.sample.workflow.purchase.controller.service.DetailServiceTransition</transition-class>
    <next-page>
      <page-path>/sample/im_workflow/purchase/detail.jsp</page-path>
    </next-page>
  </service>
```

処理詳細／参照詳細／過去案件詳細／確認詳細画面（以下、詳細画面）では、コンテンツ定義で定義した画面が表示されます。そのため、詳細画面に IM-Workflow の情報（案件名や添付ファイルなど）を表示する場合は、IM-Workflow が提供するタグライブラリを使用します。

案件の情報を表示するためのタグライブラリです。

案件番号	0000000010
案件名	物品購買
申請者	円山益男
申請基準日	2012/09/19

■ スクリプト開発モデル detail.html

```
43 | </header>
44 | <imart type="workflowMatterData" systemMatterId=$data.imwSystemMatterId
45 |                                  displayItem="matter_number,matter_name,apply_user,apply_base_date" />
46 | <table class="imui-form">
```

■ JavaEE 開発モデル detail.jsp

```
51 | </header>
52 | <workflow:workflowMatterData systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>'
53 |                              displayItem="matter_number,matter_name,apply_user,apply_base_date" />
54 | <table class="imui-form">
```

案件の添付ファイルを表示するためのタグライブラリです。

添付ファイル			
ファイル名	サイズ	登録者	登録日時
見積書	1 KB	円山益男	2012/09/19 19:49

■ スクリプト開発モデル detail.html

```
69 |   </table>
70 |   <imart type="workflowMatterFile" systemMatterId=$data.imwSystemMatterId />
71 | </div>
```

■ JavaEE 開発モデル detail.jsp

```
78 |   </table>
79 |   <workflow:workflowMatterFile systemMatterId='<%=(String)request.getAttribute("imwSystemMatterId")%>' />
80 | </div >
```

◆ 案件に添付ファイルがない場合は、表示されません。

これらの詳細画面は、スマートフォン版 IM-Workflow から画面遷移した際、PC 用の画面を新しいウィンドウで開きます。スマートフォンからの画面遷移で PC 用の画面を表示させたい場合は、明示的にクライアントタイプを PC に切り替える必要があります。

■ スクリプト開発モデル detail.js

```
19 | function init ( request ) {
20 |       ClientTypeSwitcher.oneTimeSwitchTo('pc');
21 |
```

■ JavaEE 開発モデル detail.jsp

```
 7 | <%
 8 | ClientTypeSwitcher.oneTimeSwitchTo("pc");
 9 | %>
```

ClientTypeSwitcher について、詳細は API リストを参照してください。

また、新しいウィンドウで表示する画面にグローバルナビやマイメニューを表示させないようにするには、以下のフィルターに画面のパスを追加する必要があります。

```
<./conf /theme-head-only-path-config.xml>
```

■ スクリプト開発モデル

```
 7 |  <path>/sample/im_workflow/purchase/screen/detail</path>
```

■ JavaEE 開発モデル

```
 8 |  <path>/imw_sample_purchase-detail.service</path>
```

6.2.2 ユーザプログラム

6.2.2.1 アクション処理プログラム

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/ActionProcess1.js>
```

■ JavaEE 開発モデル

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/ActionProcess1.java>
```

サンプルデータでは[ActionProcess1]を申請ノードのアクション処理として定義されています。

[ActionProcess1]では、下記の 2 つの処理を行っています。

■ ユーザアプリケーションのデータをテーブルに保存する。

- 申請または一時保存を行った場合に、画面に入力された情報をユーザアプリケーションで定義している独自のテーブルに登録／更新しています。

■ 案件番号を採番する。

- 案件番号は、申請のアクション処理で設定する必要があります。
- ここでは、IM-Workflow が提供する「WorkflowNumberingManager#getNumber()」で案件番号の採番を行っています。

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/ActionProcess2.js>
```

■ JavaEE 開発モデル

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/ActionProcess2.java>
```

サンプルデータでは[ActionProcess2]を申請ノードのアクション処理として定義しています。

[ActionProcess2]では、画面から入力された「数量×金額」である”合計金額”を案件プロパティとして登録する処理を行っています。

6.2.2.2 案件終了処理プログラム

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/action/MatterEndProcess.js>
```

■ JavaEE 開発モデル

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/MatterEndProcess.java>
```

サンプルデータでは[MatterEndProcess]を案件終了処理として定義しています。

[MatterEndProcess]では、ユーザアプリケーションで定義している独自のテーブルの更新処理を行っています。

6.2.2.3 分岐開始処理プログラム

■ フロー定義“分岐ルート[スクリプト開発モデル]“で使用されている分岐開始処理プログラム

```
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition1.js>
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition2.js>
<./jssp/src/sample/im_workflow/purchase/action/RuleCondition3.js>
```

■ フロー定義“分岐ルート[JavaEE 開発モデル]“で使用されている分岐開始処理プログラム

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition1.java>
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition2.java>
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/action/RuleCondition3.java>
```

[RuleCondition1]では、”合計金額”が 10000 未満の場合に結果フラグとして成功（true）を返却します。

[RuleCondition2]では、”合計金額”が 10000 以上 50000 未満の場合に結果フラグとして成功（true）を返却します。

[RuleCondition3]では、”合計金額”が 50000 以上の場合に結果フラグとして成功（true）を返却します。

6.2.3 リスナー

6.2.3.1 未完了案件削除処理リスナー

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowActvMatterDeleteListener.js>
```

■ JavaEE 開発モデル

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowActvMatterDeleteListener.java>
```

[WorkflowActvMatterDeleteListener]では、下記の 2 つの処理を行っています。

■ ユーザアプリケーションのデータをテーブルから削除する。

- 申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。

■ 案件プロパティを削除する。

- 申請時に案件プロパティに登録した“合計金額”を案件プロパティから削除しています。

6.2.3.2 完了案件削除処理リスナー

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/listener/WorkflowCplMatterDeleteListener.js>
```

■ JavaEE 開発モデル

```
<%サンプルプログラムディレクトリ%/
        jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowCplMatterDeleteListener.java>
```

[WorkflowCplMatterDeleteListener]では、次の処理を行っています。

申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。

◆ 案件プロパティの情報は、案件削除のタイミングで IM-Workflow モジュールが自動的に削除しますので、個別の削除は不要です。

6.2.3.3 過去案件削除処理リスナー

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/purchase/listener/WorkflowArcMatterDeleteListener.js>

■ JavaEE 開発モデル

<%サンプルプログラムディレクトリ%/
jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowArcMatterDeleteListener.java>

[WorkflowArcMatterDeleteListener]では、次の処理を行っています。

申請時に登録したユーザアプリケーションのデータを案件削除と同タイミングで削除しています。

◆ 案件プロパティの情報は、案件削除のタイミングで IM-Workflow モジュールが自動的に削除しますので、個別の削除は不要です。

6.2.3.4 案件退避処理リスナー

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/purchase/listener/WorkflowMatterArchiveListener.js>

■ JavaEE 開発モデル

<%サンプルプログラムディレクトリ%/
jp/co/intra_mart/sample/workflow/purchase/listener/WorkflowMatterArchiveListener.java>

[WorkflowMatterArchiveListener]では、次の処理を行っています。

ユーザアプリケーションで定義している独自のテーブルの更新処理を行っています。

7 カスタマイズ

7.1 呼び出し画面の初期表示値指定

ここで記載している内容は、次の観点で共通です。

- 開発モデル
- クライアントタイプ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の呼び出し時に、呼び出し画面における初期表示値を外部指定する方法を説明します。

7.1.1 指定可能なパラメータ

「workflowOpenPage」タグの内部に下記パラメータを記述することにより、呼び出し画面における初期表示値を外部指定することが可能です。

No	パラメータ（物理名）	パラメータ（論理名）	呼び出し画面側の対応項目	動作対象呼び出し画面
1	imwMatterName	案件名	案件名	申請／一時保存／申請（起票案件）／再申請
2	imwComment	コメント	コメント	すべて
3	imwForcedParamFlag	強制パラメータフラグ	※動作制御用フラグ	-

また、下記のような条件のとき、「imwForcedParamFlag」（強制パラメータフラグ）の値に"1"を指定した場合のみ、初期表示値指定が反映されます。
「imwForcedParamFlag」（強制パラメータフラグ）の値に"1"を指定しない場合、または、「imwForcedParamFlag」（強制パラメータフラグ）を記述しない場合は、登録されている情報が優先されます。

No	呼び出し画面	条件
1	申請	一時保存からの申請時
2	一時保存	一時保存情報の再保存時
3	申請（起票案件）	-
4	再申請	-

7.1.2 実装例

サンプルとして提供されている「物品購買」の申請書において、申請画面で入力される「品名」を「案件名」に、「備考」を「コメント」に初期表示する例です。

なお、サンプルは PC 用画面のみ用意しています。
スマートフォン用画面の場合も全体の流れは同じです。実装中で使用するタグライブラリや Client-side JavaScript API が異なることに注意してください。

下記のプログラムが、初期表示を行うための処理が記述されたプログラムとなります。

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/purchase/screen/apply_display.html>

■ JavaEE 開発モデル

<(展開した war)/sample/im_workflow/purchase/apply_display.jsp>

これらのファイルを、以下のファイル名に変更し、上書き保存することで、申請画面において本機能の動作確認を行うことが出来ます。

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/purchase/screen/apply.html>

■ JavaEE 開発モデル

<(展開した war)/sample/im_workflow/purchase/apply.jsp>

以下のような処理を記述することで、初期表示を行うことが出来ます。

```
<imart type="head">
<title>
  <imart type="string" value=$msg.cap010 escapeXml="true" escapeJs="false" />
</title>

<imart type="workflowOpenPageCsjs" />
<script src="ui/libs/jquery-validation-1.9.0/jquery.validate.js"></script>
<script type="text/javascript">
・
・
・

function setParam() {
  $('#imwMatterName').val($('#item_name').val());
  $('#imwComment').val($('#item_comment').val());
  $('#imwForcedParamFlag').val('1');
}

$(function(){
・
・
・
  $('#openPage1').click(function(){
    setParam();
    workflowOpenPage('1');
  });
・
・
・
}
</script>
</imart>

<imart type="workflowOpenPage"
```

```
        name="workflowOpenPageForm"
        id="workflowOpenPageForm"
        method="POST"
        target="_top"
        imwUserDataId=$data.imwUserDataId
        imwSystemMatterId=$data.imwSystemMatterId
        imwAuthUserCode=$data.imwAuthUserCode
        imwApplyBaseDate=$data.imwApplyBaseDate
        imwNodeId=$data.imwNodeId
        imwFlowId=$data.imwFlowId
        imwCallOriginalParams=$data.imwCallOriginalParams
        imwNextScriptPath=$data.imwCallOriginalPagePath>

  <input type="hidden" name="imwMatterName" id="imwMatterName" />
  <input type="hidden" name="imwComment" id="imwComment" />
  <input type="hidden" name="imwForcedParamFlag" id="imwForcedParamFlag" />
.
.
.
</imart>
```

7.2 処理対象者プラグインの作成

IM-Workflow の各ノードに指定する「処理対象者」に、独自に作成した処理対象者を追加する方法を説明します。

IM-Workflow の処理対象者は、プラグインという形で機能を拡張できるようになっています。
プラグインを追加する場合には、拡張ポイントに応じた内容でプラグインの実装を作成し、対象の拡張ポイントへ Plugin するための設定ファイルを記述します。
拡張ポイントと、プラグインの関係は intra-mart Accel Platform の API である”PluginManager”によって管理されます。

7.2.1 対象ノード

処理対象者プラグインは、ノードの種類により「extension point」が決められています。

No	ノード	extension point
1	承認(※1)	jp.co.intra_mart.workflow.plugin.authority.node.approve
2	承認(※2)	jp.co.intra_mart.workflow.plugin.authority.node.approve.static
3	動的承認	jp.co.intra_mart.workflow.plugin.authority.node.dynamic
4	確認	jp.co.intra_mart.workflow.plugin.authority.node.confirm

※1 前ノードが、”申請ノード”または”承認ノード”の場合
※2 前ノードが、”申請ノード”または”承認ノード”以外の場合

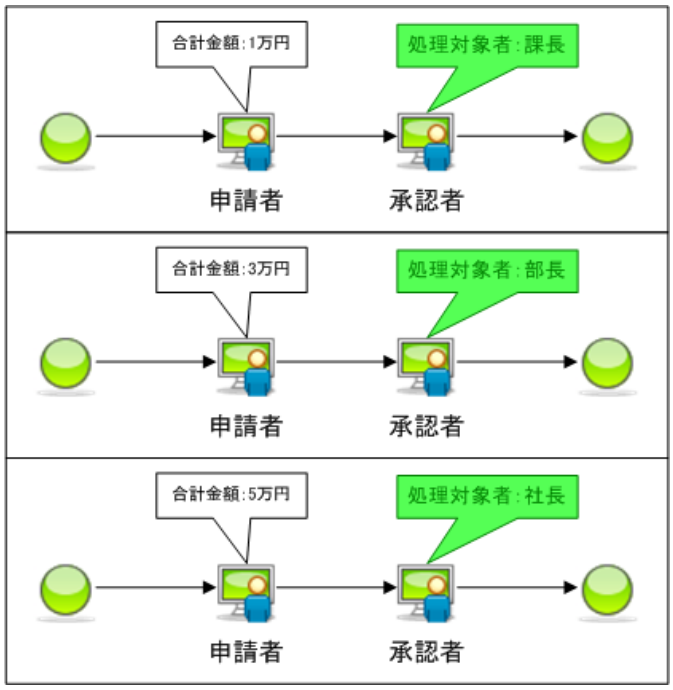
7.2.2 サンプルの説明

サンプルで提供する“処理対象者プラグイン”は、同じくサンプルで提供されている“物品購買“の画面と連携しています。

“物品購買“の画面で入力された「数量」と「金額」からの「合計金額」により、次の承認者を決定します。
具体的には、「合計金額」により、

- 1 万円未満
  - 課長
- 1 万円以上かつ 5 万円未満
  - 部長
- 5 万円以上
  - 社長

と、処理対象者に役職が割り当てられます。



7.2.3 サンプルの実行準備

ここでは、承認ノードに対して、「合計金額」で処理対象者を決めるプラグインを使用してみます。

下記のファイルを編集します。

<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve/plugin.xml>

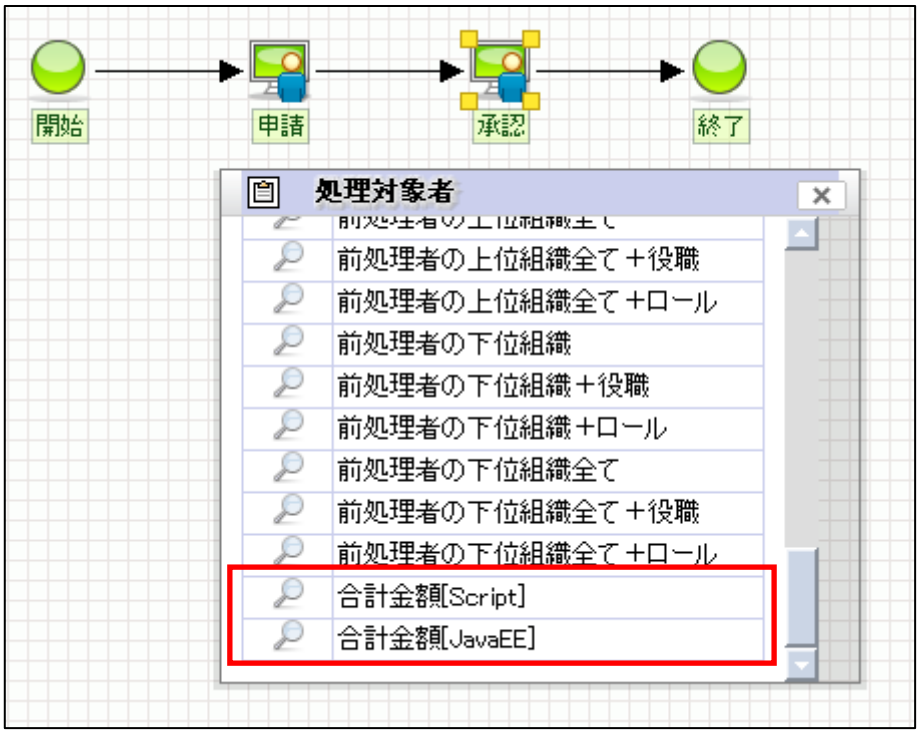
スクリプト開発モデル

JavaEE 開発モデル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
    <extension point="jp.co.intra_mart.workflow.plugin.authority.node.approve" >

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            version="7.2.0"
            rank="910"
            enable="true">
            <configPage>
                <script pagePath="sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
                </script>
            </configPage>
            <extend>
                <script
            file="sample/im_workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            version="7.2.0"
            rank="920"
            enable="true">
            <configPage>
                <javaee applicationId="imw_sample_purchase" serviceId="authority_item_total">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.031" />
                </javaee>
            </configPage>
            <extend>
                <java
        class="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

    </extension>
</plugin>
```

上記ファイルを編集後、サーバを再起動します。

[ルート定義]画面より、次のようなルートを作成します。
承認ノードの処理対象者の検索を行うと、下記のように「合計金額[Script]」および「合計金額[JavaEE]」が表示されます。



「合計金額[Script]」および「合計金額[JavaEE]」は、実装方法（開発言語）の違いによるもので、処理内容に関して違いはありません。

「合計金額[Script]」または「合計金額[JavaEE]」を選択し、ルートを作成します。

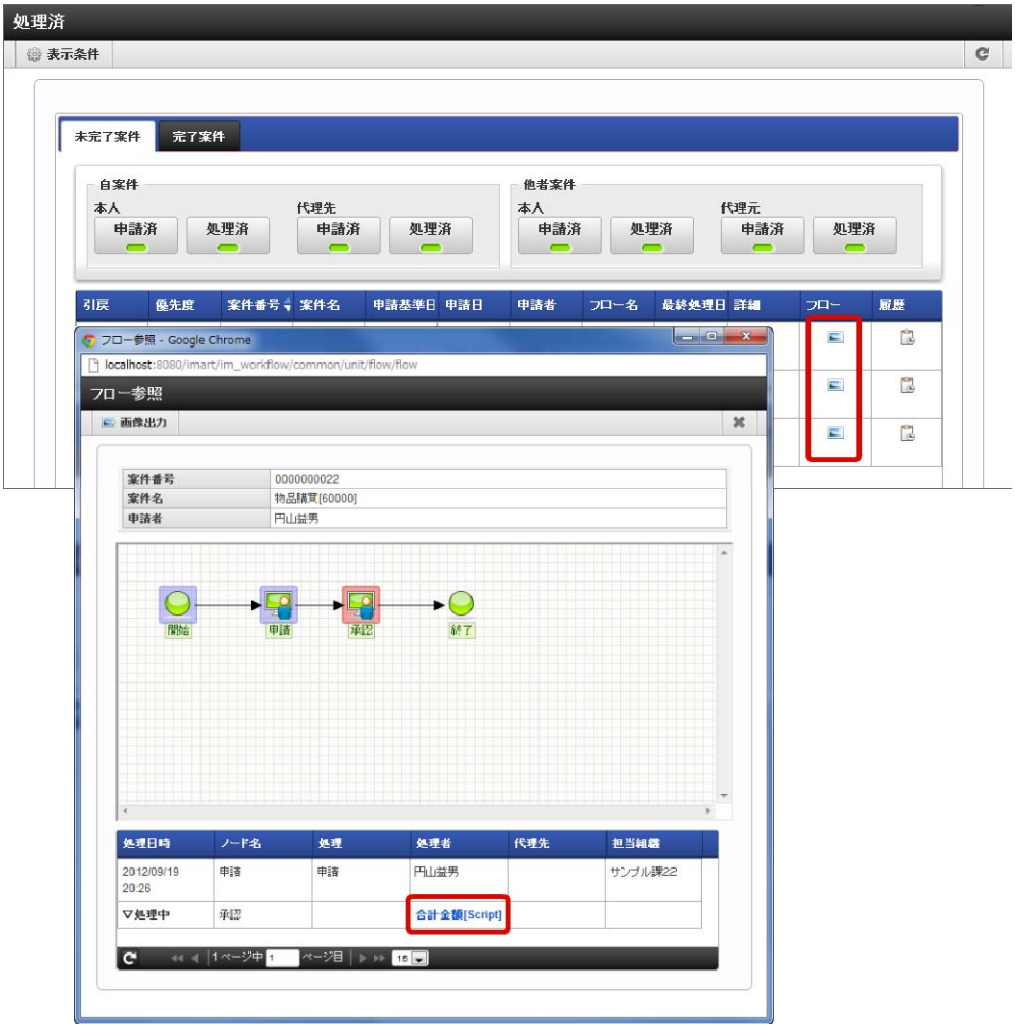
次に、[フロー定義]画面より、上記で作成したルート定義を使用したフロー定義を作成します。
この時、コンテンツは、サンプルで提供されている「スクリプト開発モデル」または、「JavaEE 開発モデル」を選択してください。

7.2.4 サンプルの実行

「7.2.3 サンプルの実行準備」で作成したフロー定義で申請を行ないます。

入力した「数量」と「金額」からの「合計金額」により、承認ノードの処理対象者が変わることを確認します。

[処理済]一覧画面より、申請を行った案件のフローを参照します。



■ 合計金額が 1 万円未満の場合

■ 合計金額が 1 万円以上かつ 5 万円未満の場合

■ 合計金額が 5 万円以上の場合

「合計金額」により、処理対象者が違うことを確認します。

7.2.5 処理対象者プラグインについて

処理対象者プラグインを作成するには、次の 3 ファイルを作成する必要があります。

7.2.5.1 「plugin.xml」

「plugin.xml」は、”PluginManager”によって管理されるファイルです。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
    <extension point="jp.co.intra_mart.workflow.plugin.authority.node.approve" >

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.script"
            version="7.2.0"
            rank="910"
            enable="true">
            <configPage>
                <script pagePath="sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.030" />
                </script>
            </configPage>
            <extend>
                <script
            file="sample/im_workflow/purchase/plugin/authority/item_total/WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

        <authority
            name="%jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            id="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.item_total.javaee"
            version="7.2.0"
            rank="920"
            enable="true">
            <configPage>
                <javaee applicationId="imw_sample_purchase" serviceId="authority_item_total">
                    <parameter key="pluginName" value="SAMPLE.IMW.CAP.031" />
                </javaee>
            </configPage>
            <extend>
                <java
    class="jp.co.intra_mart.sample.workflow.purchase.plugin.authority.item_total.WorkflowAuthorityExecEventListener" />
            </extend>
        </authority>

    </extension>
</plugin>
```

処理対象者プラグインで重要になるのは、下記の要素です。

<table>
<tr><td colspan="2"><extension point></td><td>処理対象者プラグインを差し込むノードの種類により、<extension point>が変わります。
差し込みたいノードの<extension point>を指定します。</td></tr>
<tr><td rowspan="2"><configPage></td><td><script pagePath></td><td rowspan="2"><configPage>は、[ルート定義]画面において、ノードに設定する処理対象者の一覧画面から、処理対象者プラグインが選択されたときに呼ばれるプログラムです。

このプログラムは、スクリプト開発モデルおよび、JavaEE 開発モデルで記述することが可能です。

スクリプト開発モデルで、このプログラムを作成する場合は、<script pagePath>にパスを指定します。

JavaEE 開発モデルで、このプログラムを作成する場合は、applicationId および serviceId を指定します。</td></tr>
<tr><td><javaee applicationId serviceId></td></tr>
<tr><td rowspan="2">< extend ></td><td><script file></td><td rowspan="2">< extend > に指定するプラグラムは、処理対象者を決定するプログラムとなります。

このプログラムは、スクリプト開発モデルおよび、JavaEE 開発モデルで記述することが可能です。

スクリプト開発モデルで、このプログラムを作成する場合は、<script file>にパスを指定します。

JavaEE 開発モデルで、このプログラムを作成する場合は、<java class>にパッケージを指定します。</td></tr>
<tr><td><java class></td></tr>
</table>

サンプルは、下記のようになります。

- 承認ノード

<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve/plugin.xml>

- 承認ノード

<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.approve.static/plugin.xml>

- 動的承認ノード

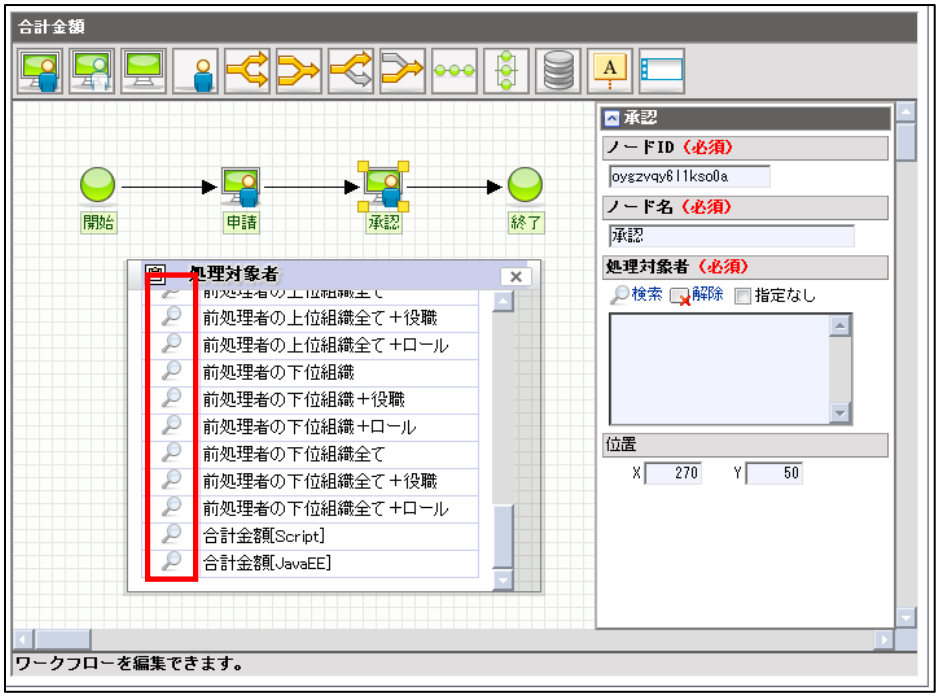
<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.dynamic/plugin.xml>

- 確認ノード

<./plugin/jp.co.intra_mart.sample.workflow.purchase.plugin.authority.node.confirm/plugin.xml>

7.2.5.2 <configPage>に指定するプログラム

[ルート定義]画面において、ノードに設定する処理対象者の一覧画面から、処理対象者プラグインが選択されたときに呼ばれるプログラムです。



選択された対象者プラグインの情報を、[ルート定義]画面に引き渡します。

サンプルプログラムは、下記のようになります。

■ スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig.html>
<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/itemTotalConfig.js>
```

■ JavaEE 開発モデル

```
<./lib/im_workflow-8.0.0-sample.jar/service-config-imw_sample_purchase.xml>
-----------------------------------------------------------------------------------------------------------------------

<service>
    <service-id>authority_item_total</service-id>
        <controller-class>jp.co.intra_mart.sample.workflow.purchase.plugin.authority.
                        item_total.controller.service.ItemTotalConfigServiceController</controller-class>
    <transition-class>jp.co.intra_mart.sample.workflow.purchase.plugin.authority.
                                item_total.controller.service.ItemTotalConfigTransition</transition-class>
    <next-page>
       <page-path>/sample/workflow/purchase/plugin/authority/item_total/itemTotalConfig.jsp</page-path>
    </next-page>
  </service>
```

7.2.5.3 < extend >に指定するプログラム

処理対象者を決定する際に実行されるプログラムとなります。
ここで指定するプログラムには、次の3つのメソッドを実装する必要があります。

メソッド	概要
execute	処理対象者を取得するメソッド 対象のノードに案件が到達したときに実行されます。
getTargetUserList	処理対象ユーザの一覧を取得するメソッド [案件操作]-[ノード編集]画面の「状況確認」ボタン押下時に表示される[対象者状況確認]画面で使用されます。
getDisplayName	処理対象者プラグインの名称を取得するメソッド プラグインの名称を表示するため使用されます。

サンプルプログラムは、下記のようになります。

■ スクリプト開発モデル

<./jssp/src/sample/im_workflow/purchase/plugin/authority/item_total/
WorkflowAuthorityExecEventListener.js>

■ JavaEE 開発モデル

<%サンプルプログラムディレクトリ%/jp/co/intra_mart/sample/
workflow/purchase/plugin/authority/item_tota/WorkflowAuthorityExecEventListener.java>

7.3 画面入力情報の保持

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

申請画面、一時保存画面、申請（起票案件）画面、再申請画面、処理画面、確認画面において、「閉じる」リンク（PC 用画面）もしくは「戻る」リンク（スマートフォン用画面）によって各画面を閉じた後に画面の再表示を行ったとき、入力内容を保持した状態で画面表示されます。

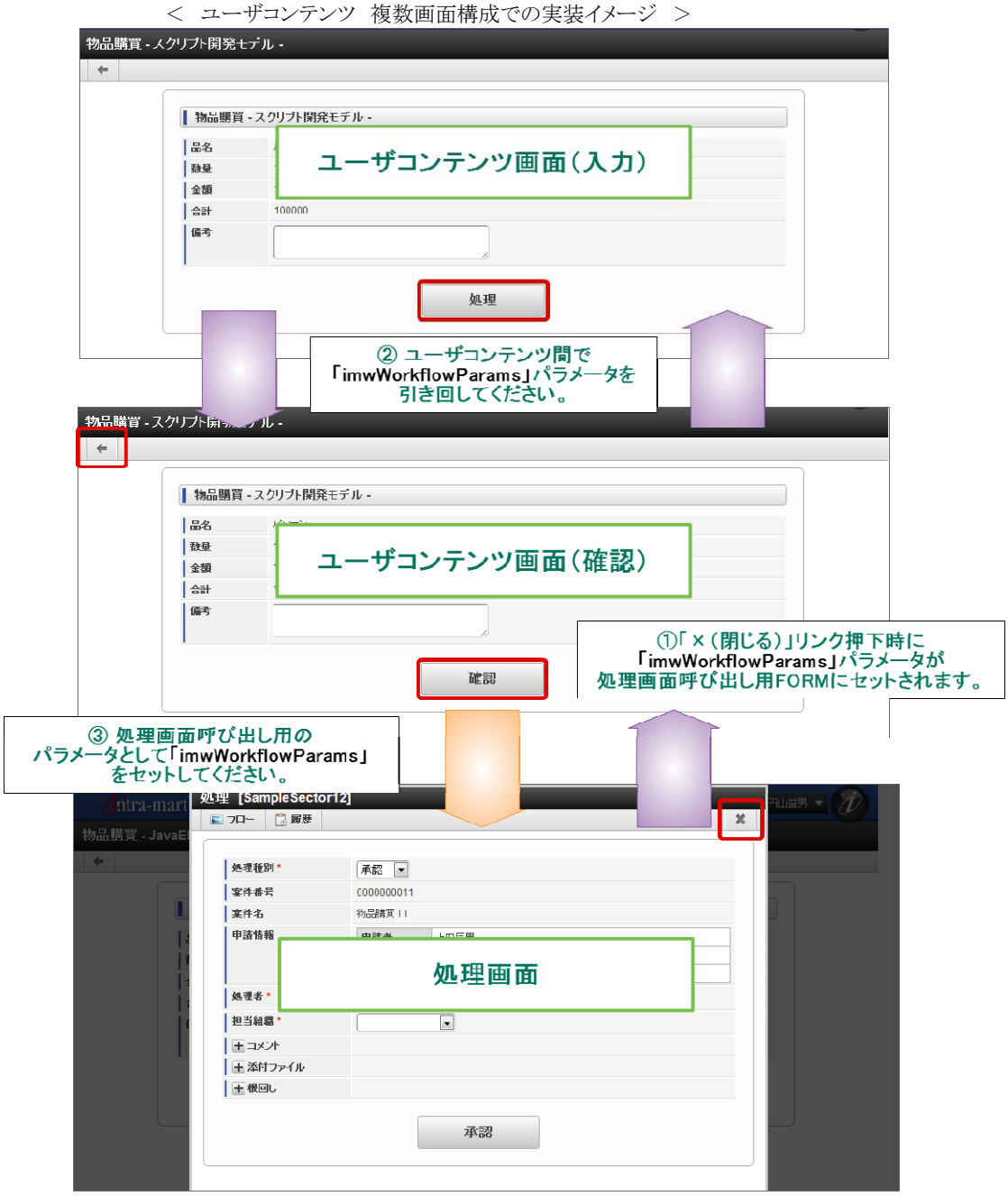
当機能の仕様概要は以下の通りです。

- 各処理画面の「閉じる」「戻る」リンク押下時に、呼出元ユーザコンテンツ内の画面呼出用タグライブラリによって生成された FORM に対して「imwWorkflowParams」というパラメータ名の hidden タグを追加し、そのタグに入力情報を格納
- 再度画面表示した際にリクエストパラメータとして「imwWorkflowParams」が含まれている場合、画面の初期表示処理で保持情報による復元表示を実行

リクエストパラメータの受け渡しによって入力情報再表示が行われるため、ユーザコンテンツが単一画面構成の場合は意識する必要がありませんが、複数画面で構成されている場合は以下対応が必要です。

各処理画面を閉じてからユーザコンテンツ間の画面遷移が行われ、その後入力内容を保持した状態で各処理画面の再表示を行う必要がある場合、「imwWorkflowParams」パラメータをユーザコンテンツ間で引き回し、各処理画面表示用のタグライブラリのコンテンツ内に「imwWorkflowParams」パラメータを hidden タグで明示的に記述してください。

＜ ユーザコンテンツ 複数画面構成での実装イメージ ＞



7.4 呼び出し画面からのコールバック関数の指定

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

申請画面、一時保存画面、申請（起票案件）画面、再申請画面、処理画面、確認画面において、「閉じる」リンク（PC用画面） もしくは 「戻る」リンク（スマートフォン用画面）によって各画面を閉じる際のコールバック関数を指定可能です。 またコールバック関数は、「7.5 処理完了後の画面遷移」 に記載のパラメータ（imwNext～）の指定を行っていない場合、IM-Workflowで提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理完了後にも実行されます。

呼出元のユーザコンテンツ画面の関数を実行する方法について説明します。

7.4.1 実装例

サンプルとして提供されている「物品購買」の申請書において、GreyBox で表示される申請画面の閉じる処理が実行された際に、「物品購買」の申請書で定義された関数をコールバック関数として実行する例です。

なお、サンプルは PC 用画面のみ用意しています。
スマートフォン用画面の場合も全体の流れは同じです。実装中で使用するタグライブラリや Client-side JavaScript API が異なることに注意してください。

下記のプログラムが、コールバック関数の実行を行うための処理が記述されたプログラムとなります。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply_callback.html>
```

- JavaEE 開発モデル

```
<（展開した war）/sample/im_workflow/purchase/apply_callback.jsp>
```

上記ファイルを、以下のファイル名に変更し、上書き保存することで、申請画面において本機能の動作確認を行うことが出来ます。

- スクリプト開発モデル

```
<./jssp/src/sample/im_workflow/purchase/screen/apply.html>
```

- JavaEE 開発モデル

```
<（展開した war）/sample/im_workflow/purchase/apply.jsp>
```

以下のような処理を記述することで、コールバック関数の実行を行うことが出来ます。

```
<imart type="head">

<imart type="workflowOpenPageCsjs" />
<script type="text/javascript">
function onClickOpenPage(pageType) {
	if (pageType != "1") {
		if(!inputCheck()) {
			return;
		}
	}

	workflowOpenPage(pageType, callbackFnc);
}

function callbackFnc() {
	alert("Callback function is executed.");
}

.
.
.

<imart type="form" name="backForm" method="POST" page=$data.imwCallOriginalPagePath>
	<imart type="hidden" imwCallOriginalParams=$data.imwCallOriginalParams />
</imart>
```

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理完了後にコールバック関数が実行された場合、コールバック関数は処理された案件の情報を引数として受け取ることができます。

```
function callbackFnc(result) {
	alert("Callback function is executed.");
	alert( result.imwSystemMatterId );	// システム案件 ID
	alert( result. imwUserDataId );	// ユーザデータ ID
}
```

処理種別と受け取ることのできる情報の関係は以下の通りです。

処理種別	システム案件 ID imwSystemMatterId	ユーザデータ ID imwUserDataId
申請	○	-
再申請	○	-
申請（起票案件）	○	-
一時保存	-	○
処理	○	-
確認	○	-

＜「○」：取得可能 ／「-」：取得不可能＞

7.4.2 標準画面を非同期で実行する場合の注意点

IM-Workflow バージョン 8.0.4 より標準画面の処理を非同期に行う機能が追加されました。
この機能が有効の場合、標準画面の呼び出し元画面で指定されたコールバック関数の振る舞いが異なります。

IM-Workflow で提供する各処理が非同期として受付された時点で処理完了を各処理画面に通知します。ほぼ処理開始の時点で通知するイメージです。
従いまして、標準画面の呼び出し元画面で指定されたコールバック関数が実行された時点では各処理が完了していない可能性が高いです。そのため処理種別が申請の場合は、システム案件 ID を受け取ることはできません。

7.4.3 特記事項

7.4.3.1 IM-Workflowバージョン 8.0.2 における改善

IM-Workflow バージョン 8.0.2 から、連続処理／連続確認中のコールバック呼び出しの動作仕様を改善しています。

- IM-Workflow バージョン 8.0.1 までの動作仕様
  - コールバック関数の指定有無に関わらず、コールバック関数は実行されません。
- IM-Workflow バージョン 8.0.2 以降の動作仕様
  - 「7.5 処理完了後の画面遷移」に記載のパラメータ（imwNext～）を指定しない場合には各処理完了後にコールバック関数が実行されます。

※ IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、上記を意識する必要はありません。

7.5 処理完了後の画面遷移

ここで記載している内容は、次の観点において共通です。

- 開発モデル
- クライアントタイプ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の処理後に、任意の画面に遷移することが可能です。

7.5.1 遷移先を指定するためのパラメータ

IM-Workflow で提供する各処理（申請／再申請／申請（起票案件）／一時保存／処理／確認）画面の呼び出し時、「workflowOpenPage」タグの属性に下記パラメータを記述すると、処理完了後の遷移先を指定する事ができます。

No	パラメータ（物理名）	省略	説明
1	imwNextScriptPath	可	処理完了後に遷移する画面のスクリプトパス 処理後の遷移先がスクリプト開発画面の場合に指定が必要です。
2	imwNextApplicationId	可	処理完了後に遷移する画面のアプリケーション ID 処理後の遷移先が javaEE 開発画面の場合に指定が必要です。
3	imwNextServiceId	可	処理完了後に遷移する画面のサービス ID 処理後の遷移先が javaEE 開発画面の場合に指定が必要です。
4	imwNextPagePath	可	処理完了後に遷移する画面のページパス 処理後の遷移先が JSP or Servlet の場合に指定が必要です。

実現したい画面遷移によって指定する属性を決定してください。

- 処理後にユーザコンテンツの呼出元一覧画面に遷移したい場合
  - 「imwNextScriptPath」に、一覧から渡された「imwCallOriginalPagePath」を指定してください。
    ※連続処理、連続確認の場合は、次の案件ノードがあれば、該当のユーザコンテンツに遷移します。
    次の案件ノードがなければ、呼出元一覧画面に遷移します。
- 処理後に任意の画面に遷移したい場合
  - 「imwNext～」に、遷移先の画面パスを指定してください。
- 処理後にユーザコンテンツ独自のコールバック関数を実行して処理画面を閉じる、もしくは処理画面を閉じることのみ実行したい場合
  - 「imwNext～」に何も設定しないでください。

7.5.2 遷移先画面が受け取ることのできるリクエストパラメータ

遷移元の処理画面の種類によって、遷移先では下記の情報をリクエストパラメータとして受け取る事ができます。

No	遷移元処理画面	パラメータ(物理名)	パラメータ(論理名)	備考
1	申請／再申請／申請(起票案件)／処理／確認	imwSystemMatterId	システム案件 ID	-
2	一時保存	imwUserDataId	ユーザデータ ID	-
3	すべて	imwCallOriginalParams	呼出元パラメータ	ユーザコンテンツが一覧画面からリクエストパラメータとして受け取ることのできる値と同じ値が受け取れます。
4	すべて ※連続処理／連続確認中の場合のみ	imwCallOriginalPagePath	呼出元ページパス	ユーザコンテンツが一覧画面からリクエストパラメータとして受け取ることのできる値と同じ値が受け取れます。

7.5.3 特記事項

7.5.3.1 IM-Workflowバージョン 8.0.2 における改善

IM-Workflow バージョン 8.0.2 から、連続処理／連続確認中の画面遷移仕様を改善しています。

- IM-Workflow バージョン 8.0.1 までの動作仕様
  - 処理完了後の遷移先指定は無視されます。
  - 処理完了後の遷移先指定の有無に関わらず、処理完了後は次の案件のユーザコンテンツが表示されます。
- IM-Workflow バージョン 8.0.2 以降の動作仕様
  - 処理完了後の遷移先指定が行われている場合、処理完了後は指定された画面に遷移します。

※ IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、上記を意識する必要はありません。

7.6 ユーザコンテンツと連続処理／連続確認の連携方法

ここで記載している内容は、次の観点において共通です。

■ 開発モデル

「workflowOpenPage」タグの属性「imwNext～」を指定して IM-Workflow 処理後に任意の画面（呼出元一覧画面以外の画面）に遷移した場合、もしくは「workflowOpenPage」タグの属性「imwNext～」を指定せずに IM-Workflow 処理後の画面遷移を行わない場合の、ユーザコンテンツと連続処理／連続確認の連携方法について説明します。

なお、IM-Workflow スマートフォン では、連続処理／連続確認機能が存在しないため、ここで記載の事項を意識する必要はありません。

7.6.1 連続処理／連続確認を継続実行する

連続処理、連続確認を継続し、次の案件ノードに対応するユーザコンテンツ画面に遷移するためには、次の実装を行ってください。

- 一覧から渡された「imwCallOriginalPagePath」が指し示す画面に遷移してください。
- 一覧から渡された「imwCallOriginalParams」を遷移先画面へのリクエストパラメータとして設定してください。

7.6.2 連続処理／連続確認を中断する

連続処理、連続確認を中断し、一覧から渡された「imwCallOriginalPagePath」が指し示す画面に遷移するためには、「imwCallOriginalPagePath」への画面遷移の前に、セッションからクライアント固有情報を削除してください。
セッションキーは **”IMW_LAST_PROCESSED_MATTER_INFO_IN_SERIAL”** です。

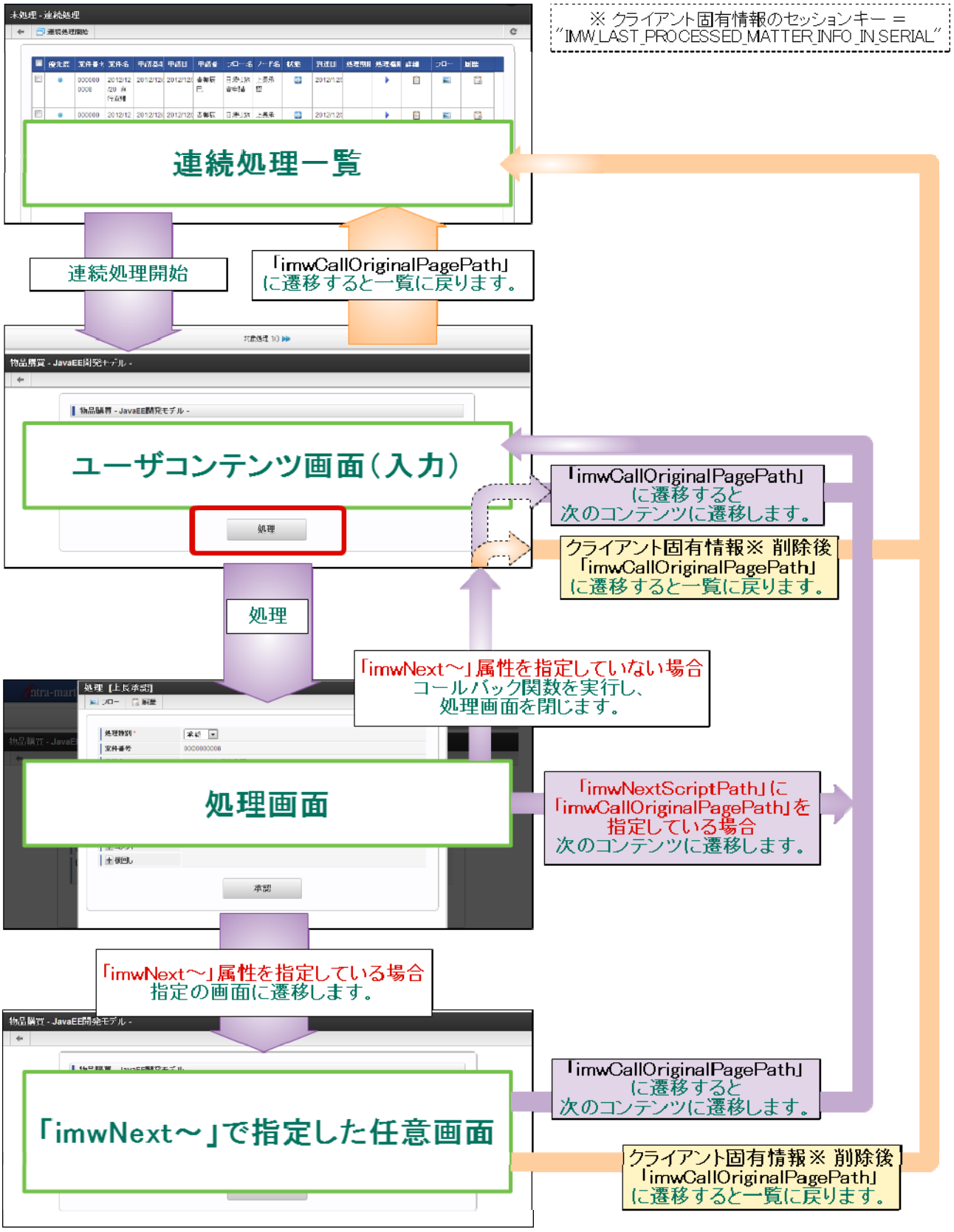
スクリプト開発モデルでセッションからクライアント固有情報を削除する場合は、次のメソッドを利用します。

```
Client.remove(Strign key)
```

javaEE 開発モデルでセッションからクライアント固有情報を削除する場合は、次のメソッドを利用します。

```
HttpSession.removeAttribute(java.lang.String name)
```

連続処理の場合の画面遷移を図示します。なお、連続確認の場合も画面遷移は同様です。



7.7 PC版ユーザコンテンツをスマートフォン用画面としても利用する

ここで記載している内容は、次の観点において共通です。

- 開発モデル

PC 版ユーザコンテンツとして作成した画面を、スマートフォン用画面として動作させる方法を説明します。
この方法を採用すると、PC 版ユーザコンテンツとスマートフォン版ユーザコンテンツをひとつの画面でまかなうことも可能です。

ただし、スマートフォン端末で PC 版ユーザコンテンツを表示した場合、さまざまな制限事項があります。
そのため、PC 版のユーザコンテンツとスマートフォン版のユーザコンテンツは、それぞれ独自に実装することを推奨します。

詳しくは「intra-mart Accel Platform リリースノート」の制限事項を参照してください。

7.7.1 必要な作業

マスタ設定と、実装の修正を行う必要があります。

7.7.1.1 マスタ定義のスマートフォン用画面設定を行う

「サイトマップ」-「マスタ定義-コンテンツ定義」より、設定対象のコンテンツ定義に対し「画面」を選択し
スマートフォン用の画面定義を新規作成、または編集してください。
画面パスとして、スマートフォン用の画面として利用する PC 版ユーザコンテンツを指定してください。

以降、必要に応じて、フロー定義の個別設定などを行ってください。

マスタ定義の新規作成、編集手順は「IM-Workflow 管理者操作ガイド」を参照してください。

以上を行ったうえで、スマートフォン端末で対象のフローの申請画面を表示すると、PC 版ユーザコンテンツが表示されるようになります。

ただし、この状態では PC 版ユーザコンテンツにスマートフォン用の画面テーマが適用されてしまい、レイアウトが崩れてしまう場合があります。

そこで、PC 版ユーザコンテンツの実装に対して修正を行います。

7.7.1.2 クライアントタイプをPCに切り替える

ユーザコンテンツの実装において、クライアントタイプを PC に切り替える必要があります。

画面表示を行う際のサーバサイドロジックにおいて、ClientTypeSwitcher.oneTimeSwitchTo を利用し、ユーザコンテンツとして表示する画面のクライアントタイプを無条件で PC に切り替えてください。

```
ClientTypeSwitcher.oneTimeSwitchTo("pc");
```

ClientTypeSwitcher について、詳細は API リストを参照してください。

実装の修正を行う対象は、スマートフォン用画面として動作させる PC 版ユーザコンテンツすべてとなります。

以上を行うことで、レイアウトが崩れることなく PC 版ユーザコンテンツをスマートフォン端末で表示することができるようになります。

この状態で、IM-Workflow が提供する案件の各処理画面（GreyBox 上に表示される画面）が正常に表示されない（画面が表示されない、画面レイアウトが崩れる）場合のみ、以降の作業を行ってください。

7.7.1.3 補足修正

ワークフロー処理を実行する画面を表示するための Client-side JavaScript API「workflowOpenPage」の引数として、各種一覧画面からリクエストパラメータとして受け取った「画面種別（imwPageType）」を **そのまま** 受け渡している場合、修正が必要です。

クライアントタイプがスマートフォンの場合、各種一覧からは画面種別としてスマートフォン用画面の値が受け渡されます。
「workflowOpenPage」の引数には、PC 用の画面種別の値を受け渡してください。

画面種別のクライアントタイプ別対応は下表のとおりです。

画面種別 クライアントタイプ	申請	一次保存	申請 （起票案件）	再申請	処理	確認
PC	0	1	2	3	4	5
スマートフォン	10	11	12	13	14	15

7.8 ユーザコンテンツ画面への不正な直接アクセスを抑止する

ここで記載している内容は、次の観点で共通です。

■ クライアントタイプ

IM-Workflow 標準機能では、IM-Workflow の各種一覧画面からユーザコンテンツ画面に遷移することができます。
この場合、IM-Workflow の標準機能は、ログインユーザが対象のコンテンツ画面の表示権限を保持しているか判定を行い、権限がない場合はエラー画面を表示します。

上記の通常遷移時以外の場合、IM-Workflow の標準機能によるユーザコンテンツ画面の表示権限の判定が行われません。
例えば、ユーザコンテンツ画面のURLに直接アクセスが行われた場合、IM-Workflow の標準機能による表示権限の判定が行われないため、ユーザコンテンツ画面のつくりによっては、表示権限を持たないユーザにユーザコンテンツ画面の内容を閲覧されてしまう可能性があります。

上記の状態でも、ユーザコンテンツ画面の表示後に各種処理（申請、承認など）を実行するタイミングでは、IM-Workflow の標準機能による処理権限の判定が行われるため、不正な処理が実行されてしまうことはありません。

ただし、表示権限のないユーザにユーザコンテンツ画面を閲覧されてしまうことが運用上の問題となる場合には、以降の対応を行うことにより、ユーザコンテンツ画面への不正な直接アクセスを抑止することが可能です。

7.8.1 対象者

以下の対応を検討している方を対象としています。

- ユーザコンテンツ画面へのアクセス権限について、セキュリティ強化を図りたい方
- intra-mart Accel Platform の認可機構を利用し、ユーザコンテンツ画面の表示権限を制御したい方
- すでに実施済みのセキュリティ対応について、IM-Workflow 標準の方法に切替えたい方

7.8.2 対象パス種別

ユーザコンテンツ定義の画面定義において、以下のパス種別として登録する画面を対象としています

- javaEE 開発モデル
- JPS or Servlet

パス種別が「スクリプト開発モデル」であるユーザコンテンツ画面については、スクリプト開発のセキュアな機構で直接のアクセスが抑止されているため、対応の必要はありません。

7.8.3 対応方法

対応方法としては、以下のいずれかを選択可能です。

1. 認可設定
2. ユーザコンテンツ画面の追加開発（カスタマイズ）

どちらの方法を選択すべきかは、下表を参照してください。

要件	推奨する対応方法
ユーザコンテンツ画面の実装を改修することができない	認可設定
アクセス権設定を認可機構で統一的に扱いたい	
IM-Workflow の標準機能と同等のユーザコンテンツ画面表示権限判定を実行したい	ユーザコンテンツ画面の追加開発（カスタマイズ）

以降では、それぞれの対応方法の詳細について説明します。
運用形態や影響範囲を考慮の上、適当な方法を選択してください。

7.8.3.1 認可設定

認可機構により、ユーザコンテンツ画面を「リソース」として登録し、アクセス権設定を行います。
認可の仕様については「認可仕様書」を参照してください。

認可設定による対応の特徴は以下の通りです。

- ユーザの権限を認可機構で集約して管理することが可能です。
- ユーザコンテンツ画面の実装の改修は不要です。
- IM-Workflow のルート定義で設定される処理対象者を包含する範囲で認可設定を行う必要があります。
  - 例として、同一の申請用ユーザコンテンツ画面を、フローA とフローB で流用している場面を想定します。
    - フローA はルート A を利用しており、申請ノードの処理対象者は「サンプル課11」です。
    - フローB はルート B を利用しており、申請ノードの処理対象者は「サンプル部門02」です。
  - この場合、申請用ユーザコンテンツ画面の認可設定としては、「サンプル課11」と「サンプル部門02」からの実行を許可する設定を行う必要があります。

以降では、IM-Workflow のコンテンツ定義における「パス種別」ごとに、認可設定を行う際の参考となるドキュメントを紹介します。

7.8.3.1.1 パス種別「javaEE開発モデル」の場合

「移行ガイド の 個別対応（im-JavaEE Framework）」の認可設定部分を参照してください。

7.8.3.1.2 パス種別「JSP or Servlet」の場合

SAStruts フレームワークを利用して実装している場合、「SAStruts+S2JDBC プログラミングガイド の 認可」を参照してください。

TERASOLUNA Global Framework を利用して実装している場合、「TERASOLUNA Global Framework プログラミングガイド の 認可」を参照してください。

7.8.3.2 ユーザコンテンツ画面の追加開発（カスタマイズ）

IM-Workflow が提供するタグライブラリ、もしくは API を利用し、ユーザコンテンツ画面の表示権限を判定します。

ユーザコンテンツ画面の表示権限とは、特定の案件を処理、もしくは参照する場合に利用されるユーザコンテンツ画面を、IM-Workflow 標準の各種一覧画面（フロー一覧、未処理一覧など）から表示することのできる権限のことを指します。

ユーザコンテンツ画面の追加開発による対応の特徴は以下の通りです。

- IM-Workflow の標準機能と同等のユーザコンテンツ表示権限判定を行うことが可能です
- ユーザコンテンツ画面の実装の改修が必要です

対応方法としては、以下のいずれかを選択可能です。

- タグライブラリによる対応
- APIによる対応

7.8.3.2.1 タグライブラリによる対応

クライアントタイプ別で、ユーザコンテンツ画面の表示権限判定用タグライブラリが用意されています。

- クライアントタイプ＝PC
  - 「workflowUserCnotentsAuth」
- クライアントタイプ＝スマートフォン
  - 「spWorkflowUserCnotentsAuth」

ユーザコンテンツ画面で上記のタグライブラリを利用するのみで、ユーザコンテンツ画面の表示権限の判定を行うことが可能です。
表示権限がない場合、HTTP403 エラーとなります。

タグライブラリによる対応を行う場合は、API リストを併せて参照してください。

7.8.3.2.1.1 推奨実装

以下のルールで実装を行うことを推奨します。

1. ユーザコンテンツ画面に「2.2 リクエストパラメータ」として受け渡されたパラメータを、すべてリクエストスコープの属性として格納します。
2. タグライブラリを引数省略の形式で利用します。

ユーザコンテンツ画面が複数画面構成の場合、追加で下記実装を行うことを推奨します。

3. 「2.2 リクエストパラメータ」としてユーザコンテンツ画面に受け渡されたパラメータを、ユーザコンテンツ画面間を遷移する際に引き回します。そのうえで、上記の 1、2 の実装を各ユーザコンテンツ画面で行います。

上記のルールを採用することにより、以下の実装上のメリットがあります。

- タグライブラリを統一的な手法で組み込むことが可能です。
- ひとつのユーザコンテンツ画面が複数の画面種別に対応した実装となっている場合でも、画面種別の差異によってタグライブラリに指定するパラメータを切り替える必要がなくなります。

7.8.3.2.1.2 実装例

IM-Workflow の JavaEE 開発モデルの以下のサンプルをもとに、推奨実装の 1、2 の例を紹介します。

- クライアントタイプ＝PC の場合
  - アプリケーション ID ： imw_sample_purchase
  - サービス ID ： apply
- クライアントタイプ＝スマートフォンの場合
  - アプリケーション ID ： imw_sp_sample_purchase
  - サービス ID ： apply

このサンプルでは、以下の画面種別に対応しています。

- 申請画面
- 一時保存画面
- 申請（起票案件）画面
- 再申請画面

それでは、順を追って実装例を示します。

1. ユーザコンテンツ画面に「2.2 リクエストパラメータ」として受け渡されたパラメータを、すべてリクエストスコープの属性として格納します

HttpServletRequest#setAttribute(String, String) を利用し、リクエストスコープの属性にパラメータを格納します。
※サンプルではあらかじめ実装されています。

<%サンプルプログラムディレクトリ%/
jp/co/intra_mart/sample/workflow/purchase/controller/service/ApplyServiceTransition.java>

```
package jp.co.intra_mart.sample.workflow.purchase.controller.service;

import javax.servlet.http.HttpServletRequest;

import jp.co.intra_mart.framework.base.service.DefaultTransition;
import jp.co.intra_mart.framework.base.service.ServicePropertyException;
import jp.co.intra_mart.framework.base.service.TransitionException;

public class ApplyServiceTransition extends DefaultTransition {

    @Override
    public String getNextPage() throws ServicePropertyException, TransitionException {
        final ApplyServiceResult serviceResult = (ApplyServiceResult) getResult();
        return getNextPagePath(serviceResult.getNextPageServiceId());
    }

    @Override
    public void setInformation() throws TransitionException {
        final HttpServletRequest request = getRequest();
        final ApplyServiceResult serviceResult = (ApplyServiceResult) getResult();
        request.setAttribute("imwGroupId", serviceResult.getImwGroupId());
        request.setAttribute("imwUserCode", serviceResult.getImwUserCode());
        request.setAttribute("imwPageType", serviceResult.getImwPageType());
        request.setAttribute("imwUserDataId", serviceResult.getImwUserDataId());
        request.setAttribute("imwSystemMatterId", serviceResult.getImwSystemMatterId());
        request.setAttribute("imwNodeId", serviceResult.getImwNodeId());
        request.setAttribute("imwArriveType", serviceResult.getImwArriveType());
        request.setAttribute("imwAuthUserCode", serviceResult.getImwAuthUserCode());
        request.setAttribute("imwApplyBaseDate", serviceResult.getImwApplyBaseDate());
        request.setAttribute("imwContentsId", serviceResult.getImwContentsId());
```

```
            request.setAttribute("imwContentsVersionId", serviceResult.getImwContentsVersionId());
            request.setAttribute("imwRouteId", serviceResult.getImwRouteId());
            request.setAttribute("imwRouteVersionId", serviceResult.getImwRouteVersionId());
            request.setAttribute("imwFlowId", serviceResult.getImwFlowId());
            request.setAttribute("imwFlowVersionId", serviceResult.getImwFlowVersionId());
            request.setAttribute("imwCallOriginalParams", serviceResult.getImwCallOriginalParams());
            request.setAttribute("imwCallOriginalPagePath", serviceResult.getImwCallOriginalPagePath());
            request.setAttribute("item_name", serviceResult.getItemName());
            request.setAttribute("item_amount", serviceResult.getItemAmount());
            request.setAttribute("item_price", serviceResult.getItemPrice());
            request.setAttribute("item_total", serviceResult.getItemTotal());
            request.setAttribute("item_comment", serviceResult.getItemComment());
        }
}
```

2. タグライブラリを引数省略の形式で利用します

タグライブラリを画面実装に追加します。

- クライアントタイプ＝PC の場合

＜(展開した war)/sample/im_workflow/purchase/apply.jsp＞

```
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>
<%@ taglib prefix="imartj2ee" uri="http://www.intra-mart.co.jp/taglib/core/framework" %>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui" %>
<%@ taglib prefix="imart" uri="http://www.intra-mart.co.jp/taglib/core/standard" %>
<%@ taglib prefix="workflow" uri="http://www.intra-mart.co.jp/taglib/imw/workflow" %>
<imartj2ee:HelperBean id="bean" class="jp.co.intra_mart.sample.workflow.purchase.controller.view.CommonHelperBean"/>

<workflow:workflowUserContentsAuth />

<imui:head>
    .
    .
    .
```

- クライアントタイプ＝スマートフォンの場合

＜(展開した war)/sample/im_workflow_smartphone/purchase/apply.jsp＞

```
<%@ page contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>
<%@ taglib prefix="imartj2ee" uri="http://www.intra-mart.co.jp/taglib/core/framework" %>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui" %>
<%@ taglib prefix="imart" uri="http://www.intra-mart.co.jp/taglib/core/standard" %>
<%@ taglib prefix="workflow" uri="http://www.intra-mart.co.jp/taglib/imw/workflow-smartphone" %>
<imartj2ee:HelperBean id="bean" class="jp.co.intra_mart.sample.workflow.purchase.controller.view.CommonHelperBean"/>

<workflow:spWorkflowUserContentsAuth />

<imui:head>
    .
    .
    .
```

7.8.3.2.2 APIによる対応

タグライブラリによる対応では実現できない要件がある場合は、ユーザコンテンツ画面の表示権限の判定 API を利用することで、任意の動作をさせることが可能です。
具体的には、次のような場合を想定します。

- 業務ロジックのとの兼ね合いで、タグライブラリを利用することができない場合
- 表示権限がないと判定された際、HTTP403 エラーではなく任意の処理を行いたい場合

対応する API は「jp.co.intra_mart.foundation.workflow.util.auth.WorkflowAuthUtil」です。
権限判定の結果は boolean 値で返却されるため、結果をうけて任意の処理を行うことが可能です。
詳細は API リストを参照してください。

intra-mart Accel Platform
IM-Workflow プログラミングガイド

2014/04/01 第7版

TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/