IM-ExApply for Accel Platform

プログラミングガイド

初版

2015 年 10 月 1 日

株式会社 NTT データイントラマート

＜＜　変更履歴　＞＞

変更年月日	変更内容
2015/10/1	初版

＜＜　目次　＞＞

1. はじめに

本ドキュメントは、IM-ExApply for Accel Platform への処理の追加手順及びプログラミング方法について記述しています。intra-mart の JavaEE 開発モデル（J2EE ベース開発モデル含む）については intra-mart マニュアルより「開発関連」を参照してください。
また、本ドキュメントにて記載する記述例の一部のファイルは以下の場所に格納されています。
｛パブリックストレージのパス｝/sprist/sep/sample/source/配下

1.1 用語

以下、本ドキュメントで使用する用語を定義します。

intra-mart AccelPlatform
以下、iAP と略します。
IM-ExApply for Accel Platform
以下、EX 申請システムと略します。
JavaEE 開発モデル（J2EE ベース開発モデル含む）
以下、JavaEE 開発モデルと略します。
スクリプト開発モデル（ページベース開発モデル含む）
以下、スクリプト開発モデルと略します。

1.2 処理の追加について

EX 申請システムには処理を追加する為の JAVA クラス、JavaScript ファイルがあらかじめ用意されています。これらの JAVA クラス、JavaScript に処理を記述していただくことにより、EX 申請システムにて標準で行う処理に加え、独自の処理を追加することが可能となります。
処理を記述することのできる対象を下表に示します。

表1.　処理対象

項番	処理対象	説明
1	登録前処理	様式の登録・訂正時の前処理を記述することができます。
2	ワークフロー登録前処理	ワークフローの承認処理時の前処理を記述することができます。 但し、ワークフローで追記を使用する場合のみ実行されます。
3	ワークフロー後処理	ワークフローの起票、承認、否認、差戻、引戻、取止、破棄処理時の後処理を記述することができます。
4	ワークフロー審議処理（追記）	承認時の追記登録で、登録ボタンをクリックしなくても審議ボタンをクリックした時に追記登録処理を行うことができます。
5	ワークフロー操作後処理（削除）	ワークフローの削除時に申請書データを削除しデータの同期を取ることができます。
6	登録データ検索削除後処理	登録データ検索の様式削除時の後処理で自由に処理を記述することができます。

2. 登録前処理

2.1 登録前処理・ワークフロー登録前処理の追加

EX 申請システムにて様式の登録処理を行う場合に、登録前処理を追加し実行することができます。
登録前処理が追加可能な登録処理を以下に示します。

表2.　追加可能登録処理

項番	追加可能登録処理
1	[サイトマップ] － [IM-ExApply] － [様式選択]からの登録処理
2	[サイトマップ] － [IM-ExApply] － [登録データ検索]からの訂正処理
3	intra-mart の標準ワークフロー機能の承認処理を行う場合の前処理

※上記項番 1 と項番 2 においては同一のプログラムの呼出しを行います。よって登録処理と訂正処理それぞれのプログラムを分けて作成することはできません。処理を分ける場合にはプログラム内にて分岐を行ってください。
※intra-mart の標準ワークフロー機能の承認処理を行う場合の前処理はワークフローにて追記を行う場合のみ実行される前処理となります。
※登録前処理の作成は JavaEE 開発モデルにて処理を記述した場合のみとなります。JavaScript では登録前処理を追加することはできません。

2.2 登録前処理の構成

(1) プログラム構成図

下図にイベントフレームワーク以降におけるプログラム構造図を示します。

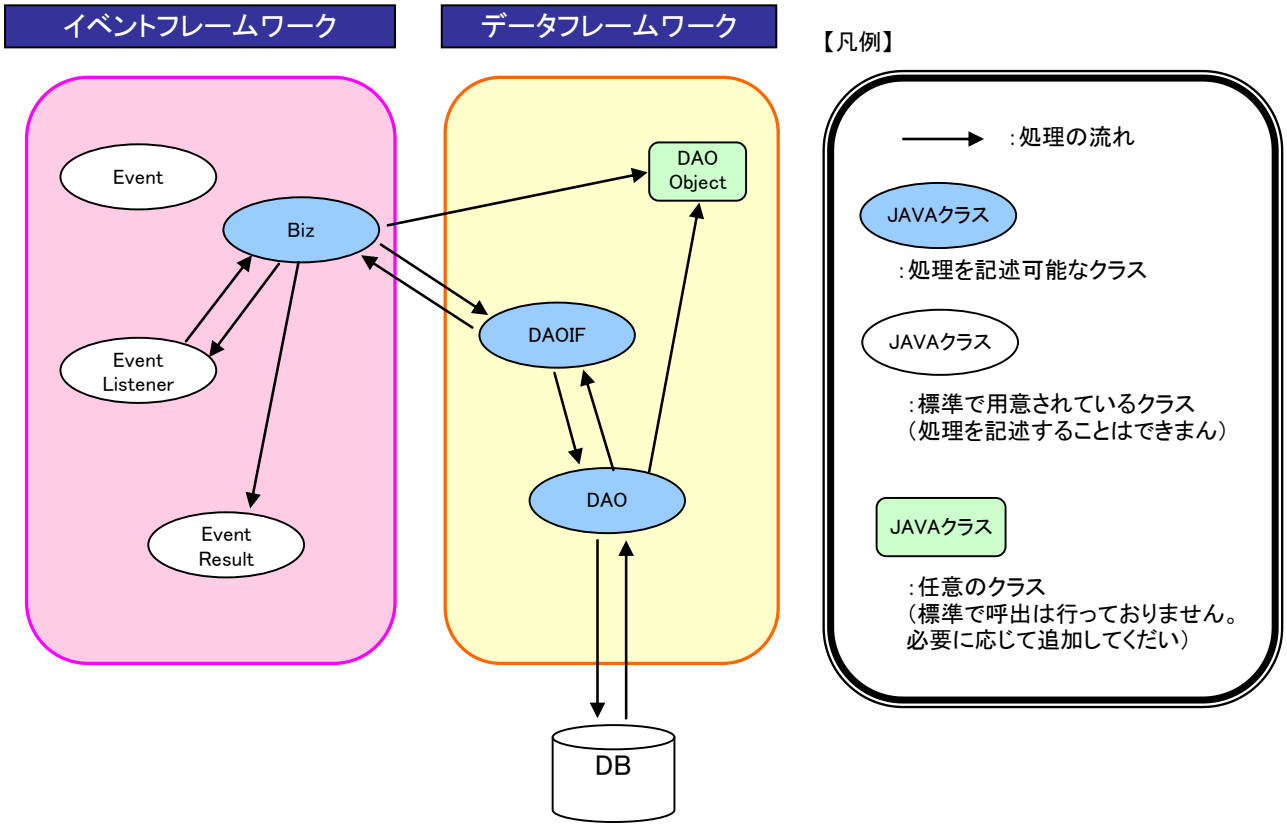


図 1.　登録前処理プログラム構成図

2.3 JavaEE 開発モデルで登録前処理を追加する

EX 申請システムでは Biz クラス、DAO クラス、DAOIF クラスの 3 つの JAVA クラスを用意しています。それ以外の JAVA クラスについては処理を記述することはできません。

2.4 作成する JAVA クラス

EX 申請システムの標準の登録処理から下表に示す JAVA クラスが呼び出され実行されます。
登録前処理を追加する場合、対象の処理に合わせて下表に示す JAVA クラスを作成します。

表3.　呼び出される JAVA クラス

作成する登録処理	JAVA クラス名	説明
登録・訂正処理	jp.co.nttdata_chugoku.sprist.sep.formselect.event.AddOnEntryBiz	実際の処理を記述する
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAO	データベースへのアクセス処理を記述する
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAOIF	DAO クラスのインターフェース
ワークフローの承認処理	jp.co.nttdata_chugoku.sprist.sep.imart.event.AddOnEntryBiz	実際の処理を記述する
	jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAO	データベースへのアクセス処理を記述する
	jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAOIF	DAO クラスのインターフェース

2.5 JAVA クラスの作成方法

2.5.1 登録・訂正処理の前処理を作成する

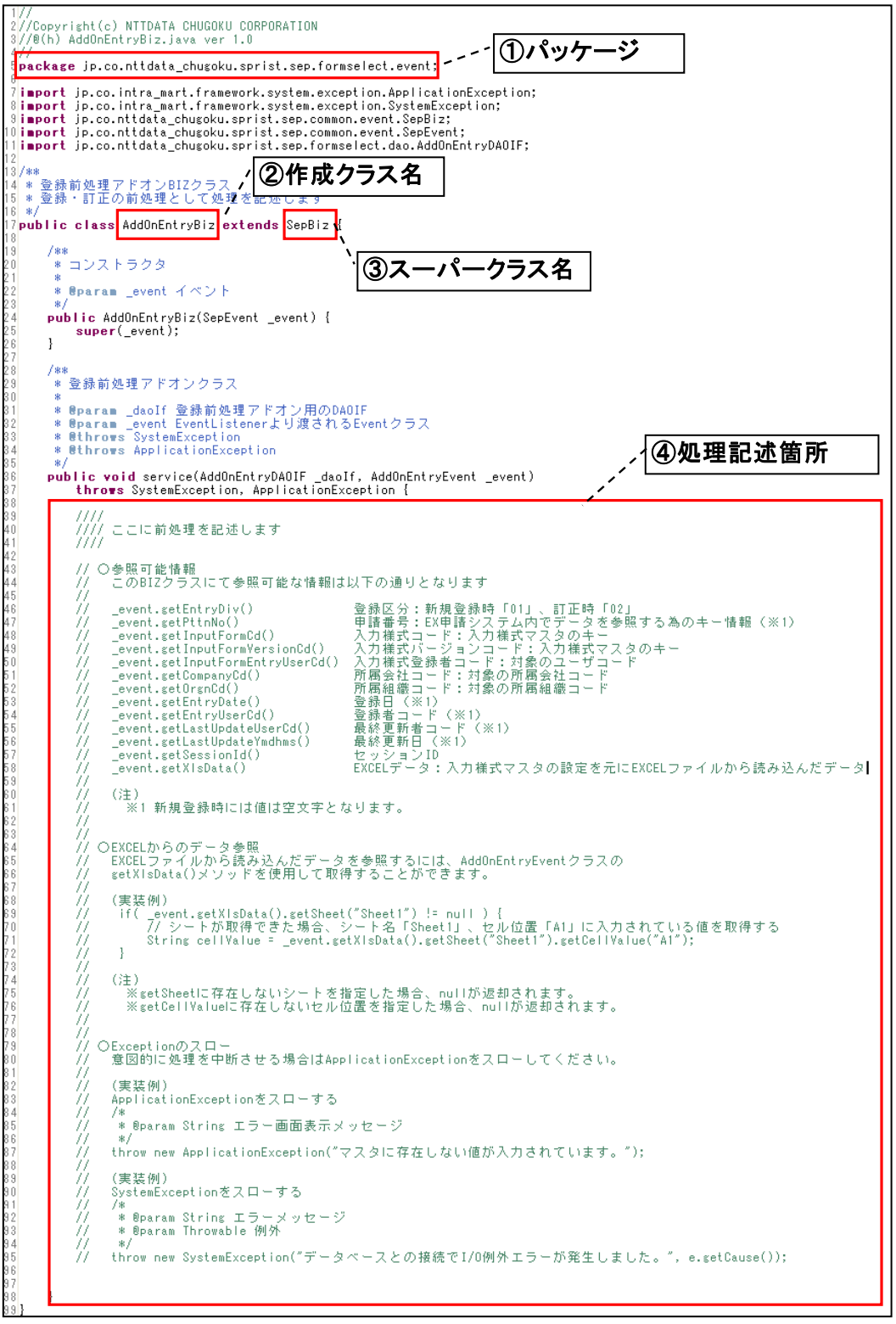
(1) Biz クラスの作成

EX 申請システムに用意された JAVA クラス「SepBiz」を継承し、Biz クラスを作成します。
Biz クラスは登録処理の EventListener 内から呼び出され、データベースへの登録前に実行されます。
この Biz クラスに登録前処理のロジックを記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]
クラス名：AddOnEntryBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.formselect.event
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz
説明：登録・訂正の前処理を記述するクラス

[継承しなければならない JAVA クラス]
クラス名：SepBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.event
説明：EX 申請システムの Biz スーパークラス




```
//
//Copyright(c) NTTDATA CHUGOKU CORPORATION
//@(h) AddOnEntryBiz.java ver 1.0
//
package jp.co.nttdata_chugoku.sprist.sep.formselect.event;

import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepEvent;
import jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAOIF;

/**
 * 登録前処理アドオンBIZクラス
 * 登録・訂正の前処理として処理を記述します
 */
public class AddOnEntryBiz extends SepBiz {

    /**
     * コンストラクタ
     *
     * @param _event イベント
     */
    public AddOnEntryBiz(SepEvent _event) {
        super(_event);
    }

    /**
     * 登録前処理アドオンクラス
     *
     * @param _daoIf 登録前処理アドオン用のDAOIF
     * @param _event EventListenerより渡されるEventクラス
     * @throws SystemException
     * @throws ApplicationException
     */
    public void service(AddOnEntryDAOIF _daoIf, AddOnEntryEvent _event)
        throws SystemException, ApplicationException {

        ////
        //// ここに前処理を記述します
        ////

        // ○参照可能情報
        //   このBIZクラスにて参照可能な情報は以下の通りとなります
        //
        //   _event.getEntryDiv()              登録区分：新規登録時「01」、訂正時「02」
        //   _event.getPtnNo()                 申請番号：EX申請システム内でデータを参照する為のキー情報（※1）
        //   _event.getInputFormCd()           入力様式コード：入力様式マスタのキー
        //   _event.getInputFormVersionCd()    入力様式バージョンコード：入力様式マスタのキー
        //   _event.getInputFormEntryUserCd()  入力様式登録者コード：対象のユーザコード
        //   _event.getCompanyCd()             所属会社コード：対象の所属会社コード
        //   _event.getOrgnCd()                所属組織コード：対象の所属組織コード
        //   _event.getEntryDate()             登録日（※1）
        //   _event.getEntryUserCd()           登録者コード（※1）
        //   _event.getLastUpdateUserCd()      最終更新者コード（※1）
        //   _event.getLastUpdateYmdhms()      最終更新日（※1）
        //   _event.getSessionId()             セッションID
        //   _event.getXlsData()               EXCELデータ：入力様式マスタの設定を元にEXCELファイルから読み込んだデータ
        //
        //   (注)
        //     ※1 新規登録時には値は空文字となります。
        //
        //
        // ○EXCELからのデータ参照
        //   EXCELファイルから読み込んだデータを参照するには、AddOnEntryEventクラスの
        //   getXlsData()メソッドを使用して取得することができます。
        //
        //   (実装例)
        //    if( _event.getXlsData().getSheet("Sheet1") != null ) {
        //        // シートが取得できた場合、シート名「Sheet1」、セル位置「A1」に入力されている値を取得する
        //        String cellValue = _event.getXlsData().getSheet("Sheet1").getCellValue("A1");
        //    }
        //
        //   (注)
        //     ※getSheetに存在しないシートを指定した場合、nullが返却されます。
        //     ※getCellValueに存在しないセル位置を指定した場合、nullが返却されます。
        //
        //
        // ○Exceptionのスロー
        //   意図的に処理を中断させる場合はApplicationExceptionをスローしてください。
        //
        //   (実装例)
        //   ApplicationExceptionをスローする
        //   /*
        //    * @param String エラー画面表示メッセージ
        //    */
        //   throw new ApplicationException("マスタに存在しない値が入力されています。");
        //
        //   (実装例)
        //   SystemExceptionをスローする
        //   /*
        //    * @param String エラーメッセージ
        //    * @param Throwable 例外
        //    */
        //   throw new SystemException("データベースとの接続でI/O例外エラーが発生しました。", e.getCause());


    }
}
```

図 2.　Biz クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.formselect.event」を指定する

② JAVA クラス名を「AddOnEntryBiz」とする

③ スーパークラス名に「SepBiz」を指定する

④ 実際の処理を記述する。AddOnEntryEvent クラスから取得可能なパラメータについては、下表参照

表4.　AddOnEntryEvent クラスから取得できる情報

メソッド名		取得する値	備考
メソッド名	戻り値のデータ型		
getEntryDiv()	String	登録区分	新規登録時「01」、訂正時「02」
getPttnNo()	String	申請番号	登録時に採番される一意な番号 ※1
getAutoNo()	String	採番番号	採番マスタから採番した番号 ※1
getInpurFormCd()	String	入力様式コード	入力様式マスタ内での一意なコード
getInputFormVersionCd()	String	入力様式バージョンコード	-
getInputFormEntryUserCd()	String	入力様式登録者コード	入力様式の登録対象者ユーザ
getCompanyCd()	String	会社コード	入力様式登録者の会社コード
getOrgnCd()	String	所属部門コード	入力様式登録者コードの所属部門コード
getEntryDate()	String	登録日	- ※1
getEntryUserCd()	String	登録者コード	- ※1
getLastUpdateUserCd()	String	最終更新者コード	- ※1
getLastUpdateYmdhms()	String	最終更新日	- ※1
getSessionId()	String	セッションＩＤ	-
getLoginUserCd()	String	ログインユーザコード	-
getLoginGroupCd()	String	ログイングループコード	-

※１　新規登録時には空文字が返却されます

(2) DAOIF クラスの作成

EX 申請システムに用意されたクラス「SepDAOIF」を継承し、DAOIF クラスを作成します。
DAOIF クラスは登録処理の EventListener 内にて生成され、Biz クラスの引数として渡されます。
また、このクラスは「AddOnEntryDAO」クラスのインターフェースクラスとなります。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnEntryDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.formselect.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAOIF
説明：DAO クラスのインターフェースクラス

[継承しなければならない JAVA クラス]

クラス名：SepDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAOIF スーパークラス

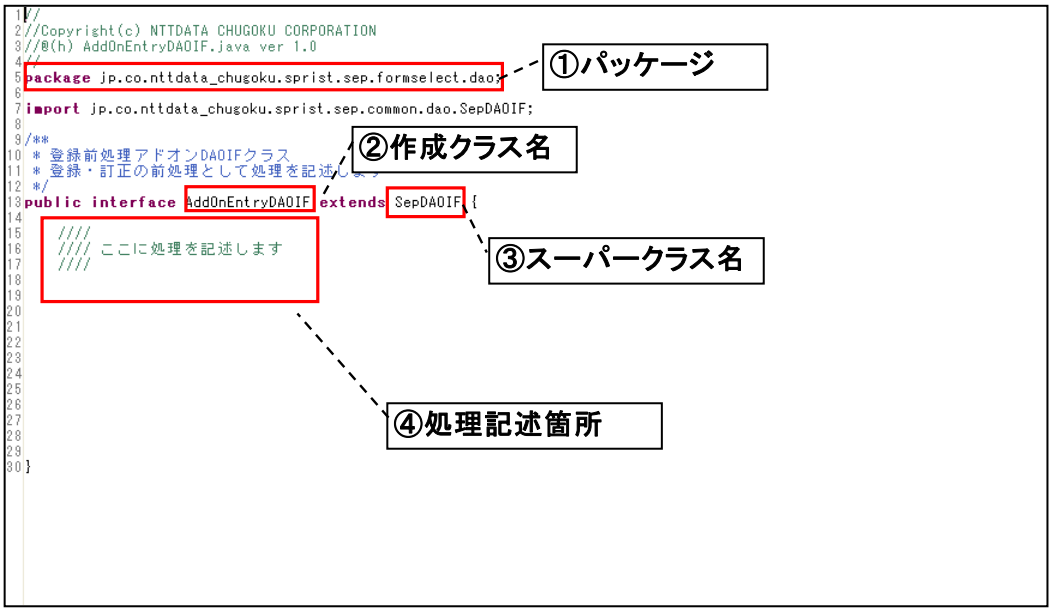


図 3.　DAOIF クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.formselect.dao」を指定する
② クラス名を「AddOnEntryDAOIF」とする
③ スーパークラス名に「SepDAOIF」を指定する
④ 実際の処理を記述する。

(3) DAO クラスの作成

EX 申請システムに用意されたクラス「SepDAO」を継承し、DAO クラスを作成します。
この DAO クラスにデータベースへのアクセス処理を記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnEntryDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.formselect.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAO
説明：データベースに対するアクセス処理を記述するクラス

[継承しなければならないクラス]

クラス名：SepDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAO スーパークラス

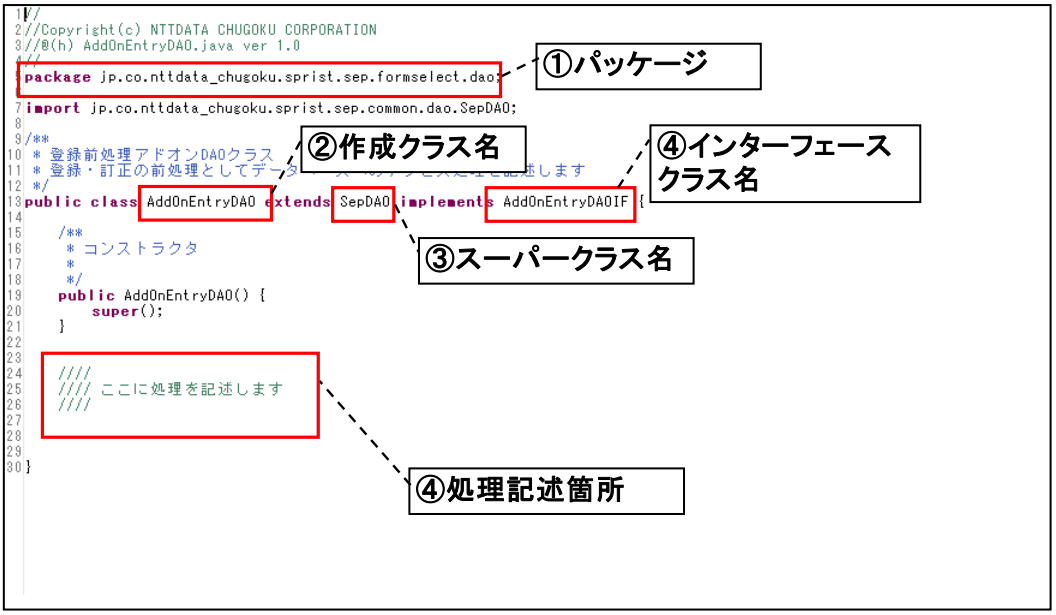


図 4.　DAO クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.formselect.dao」を指定する
② クラス名を「AddOnEntryDAO」とする
③ スーパークラス名に「SepDAO」を指定する
④ インターフェースクラス名に「AddOnEntryDAOIF」を指定する
⑤ 実際の処理を記述する。

2.5.2 ワークフローの承認処理の登録前処理を作成する

(1) Biz クラスの作成

EX 申請システムに用意された JAVA クラス「SepBiz」を継承し、Biz クラスを作成します。
Biz クラスは承認処理の EventListener 内から呼び出され、データベースへの登録前に実行されます。
この Biz クラスにワークフロー登録前処理のロジックを記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]
クラス名：AddOnEntryBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.imart.event
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz
説明：登録・訂正の前処理を記述するクラス

[継承しなければならない JAVA クラス]
クラス名：SepBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.event
説明：EX 申請システムの Biz スーパークラス


```
1//
2//Copyright(c) NTTDATA CHUGOKU CORPORATION
3//@(h) AddOnEntryBiz.java ver 1.0
4//
5package jp.co.nttdata_chugoku.sprist.sep.imart.event;
6
7import jp.co.intra_mart.framework.system.exception.ApplicationException;
8import jp.co.intra_mart.framework.system.exception.SystemException;
9import jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz;
10import jp.co.nttdata_chugoku.sprist.sep.common.event.SepBpwEvent;
11import jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAOIF;
12
13/**
14 * 登録前処理(ワークフロー追記時処理)アドオンクラス
15 * 登録(ワークフロー追記時処理)の前処理として処理を記述します
16 */
17public class AddOnEntryBiz extends SepBiz {
18
19    /**
20     * コンストラクタ
21     *
22     * @param _event イベント
23     */
24    public AddOnEntryBiz(SepBpwEvent _event) {
25        super(_event);
26    }
27
28    /**
29     * 登録前処理(ワークフロー追記時処理)アドオンクラス
30     *
31     * @param _daoIf 登録前処理(ワークフロー追記時処理)アドオン用のDAOIF
32     * @param _event EventListenerより渡されるEventクラス
33     * @throws SystemException
34     * @throws ApplicationException
35     */
36    public void service(AddOnEntryDAOIF _daoIf, AddOnEntryEvent _event)
37        throws SystemException, ApplicationException {
38
39        ////
40        //// ここに前処理を記述します
41        ////
42
43        // ○参照可能情報
44        //   このBIZクラスにて参照可能な情報は以下の通りとなります
45        //
46        //   _event.getDataEntryDiv()          データ登録区分：データ登録時「01」、データ取消時「02」
47        //   _event.getPttnNo()                申請番号：EX申請システム内でデータを参照する為のキー情報（※1）
48        //   _event.getInputFormCd()           入力様式コード：入力様式マスタのキー
49        //   _event.getInputFormVersionCd()    入力様式バージョンコード：入力様式マスタのキー
50        //   _event.getInputFormEntryUserCd()  入力様式登録者コード：対象のユーザコード
51        //   _event.getCompanyCd()             会社コード：対象の会社コード
52        //   _event.getOrgnCd()                所属コード：対象の所属コード
53        //   _event.getEntryDate()             登録日（※1）
54        //   _event.getEntryUserCd()           登録者コード（※1）
55        //   _event.getLastUpdateUserCd()      最終更新者コード（※1）
56        //   _event.getLastUpdateYmdhms()      最終更新日（※1）
57        //   _event.getSessionId()             セッションID
58        //   _event.getDataEntryDiv()          データ登録区分(00：登録、01：取消)
59        //   _event.getXlsData()               EXCELデータ：入力様式マスタの設定を元にEXCELファイルから読み込んだデータ
60        //
61        //   (注)
62        //     ※1 新規登録時には値は空文字となります。
63        //
64        //
65        // ○EXCELからのデータ参照
66        //   EXCELファイルから読み込んだデータを参照するには、AddOnEntryEventクラスの
67        //   getXlsData()メソッドを使用して取得することができます。
68        //
69        //   (実装例)
70        //    if( _event.getXlsData().getSheet("Sheet1") != null ) {
71        //        // シートが取得できた場合、シート名「Sheet1」、セル位置「A1」に入力されている値を取得する
72        //        String cellValue = _event.getXlsData().getSheet("Sheet1").getCellValue("A1");
73        //    }
74        //
75        //   (注)
76        //     ※getSheetに存在しないシートを指定した場合、nullが返却されます。
77        //     ※getCellValueに存在しないセル位置を指定した場合、nullが返却されます。
78        //
79        //
80        // ○Exceptionのスロー
81        //   意図的に処理を中断させる場合はApplicationExceptionをスローしてください。
82        //   但し、ワークフロー追記の場合の登録前処理にてApplicationExceptionをスローしても
83        //   アプリケーションエラー画面は表示されません。イントラマートの標準ワークフロー機能での
84        //   エラー画面が表示されます。
85        //
86        //   (実装例)
87        //   ApplicationExceptionをスローする
88        //   /*
89        //    * @param String エラー画面表示メッセージ
90        //    */
91        //   throw new ApplicationException("マスタに存在しない値が入力されています。");
92        //
93        //   (実装例)
94        //   SystemExceptionをスローする
95        //   /*
96        //    * @param String エラーメッセージ
97        //    * @param Throwable 例外
98        //    */
99        //   throw new SystemException("データベースとの接続でI/O例外エラーが発生しました。", e.getCause());
100       //
101       //   (注)
102       //   ページベースの後処理を使用する場合、AddOnEntryEventからログインユーザコード等の情報を取得することはできません。
103       //   ログインユーザコード・ログイングループIDを取得するには、AddOnEntryEventクラスの以下のメソッドを使用してください。
104       //    getLoginUserCd()
105       //    getLoginGroupCd()
106
107   }
108}
```

図 5.　Biz クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.imart.event」を指定する

② JAVA クラス名を「AddOnEntryBiz」とする

③ スーパークラス名に「SepBiz」を指定する

④ 実際の処理を記述する。AddOnEntryEvent クラスから取得可能なパラメータについては、下表参照

表5.　AddOnEntryEvent クラスから取得できる情報

メソッド名		取得する値	備考
メソッド名	戻り値のデータ型		
getEntryDiv()	String	登録区分	新規登録時「01」、訂正時「02」
getPttnNo()	String	申請番号	登録時に採番される一意な番号 ※1
getAutoNo()	String	採番番号	採番マスタから採番した番号 ※1
getInpurFormCd()	String	入力様式コード	入力様式マスタ内での一意なコード
getInputFormVersionCd()	String	入力様式バージョンコード	-
getInputFormEntryUserCd()	String	入力様式登録者コード	入力様式の登録対象者ユーザ
getOrgnCd()	String	所属部門コード	入力様式登録者コードの所属部門コード
getEntryDate()	String	登録日	- ※1
getEntryUserCd()	String	登録者コード	- ※1
getLastUpdateUserCd()	String	最終更新者コード	- ※1
getLastUpdateYmdhms()	String	最終更新日	- ※1
getSessionId()	String	セッションＩＤ	-
getLoginUserCd()	String	ログインユーザコード	-
getLoginGroupCd()	String	ログイングループコード	-

※１　新規登録時には空文字が返却されます

(2) DAOIF クラスの作成

EX 申請システムに用意されたクラス「SepDAOIF」を継承し、DAOIF クラスを作成します。
DAOIF クラスは承認処理の EventListener 内にて生成され、Biz クラスの引数として渡されます。
また、このクラスは「AddOnEntryDAO」クラスのインターフェースクラスとなります。

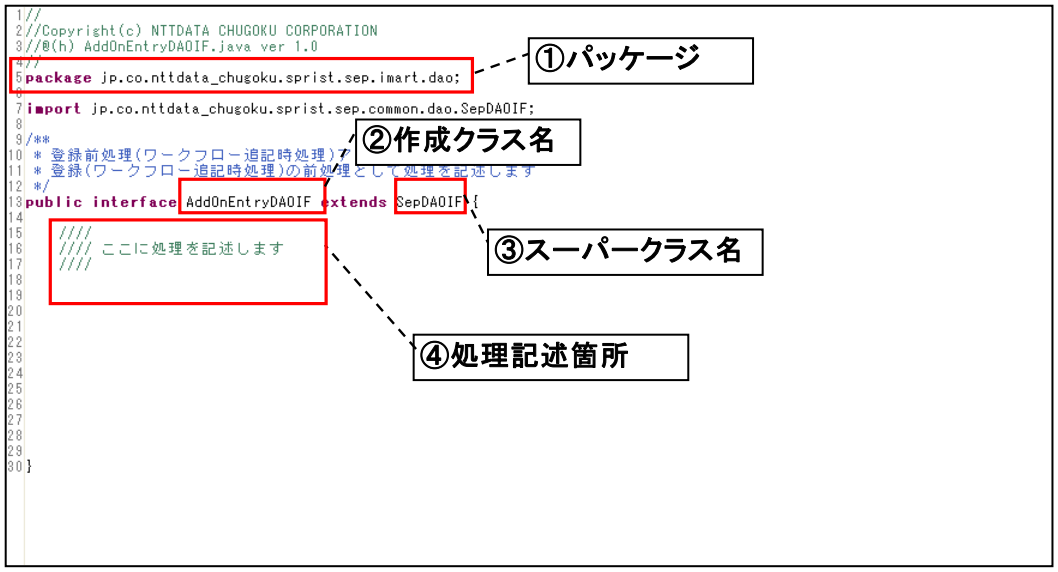
以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnEntryDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.imart.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAOIF
説明：DAO クラスのインターフェースクラス

[継承しなければならない JAVA クラス]

クラス名：SepDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAOIF スーパークラス




```
1 //
2 //Copyright(c) NTTDATA CHUGOKU CORPORATION
3 //@(h) AddOnEntryDAOIF.java ver 1.0
4 //
5 package jp.co.nttdata_chugoku.sprist.sep.imart.dao;
6
7 import jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAOIF;
8
9 /**
10  * 登録前処理(ワークフロー追記時処理)で
11  * 登録(ワークフロー追記時処理)の前処理として処理を記述します
12  */
13 public interface AddOnEntryDAOIF extends SepDAOIF {
14
15     ////
16     //// ここに処理を記述します
17     ////
18
19
20
21
22
23
24
25
26
27
28
29
30 }
```

図 6.　DAOIF クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.imart.dao」を指定する
② クラス名を「AddOnEntryDAOIF」とする
③ スーパークラス名に「SepDAOIF」を指定する
④ 実際の処理を記述する。

(3) DAO クラスの作成

EX 申請システムに用意されたクラス「SepDAO」を継承し、DAO クラスを作成します。
この DAO クラスにデータベースへのアクセス処理を記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnEntryDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.imart.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAO
説明：データベースに対するアクセス処理を記述するクラス

[継承しなければならないクラス]

クラス名：SepDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAO スーパークラス

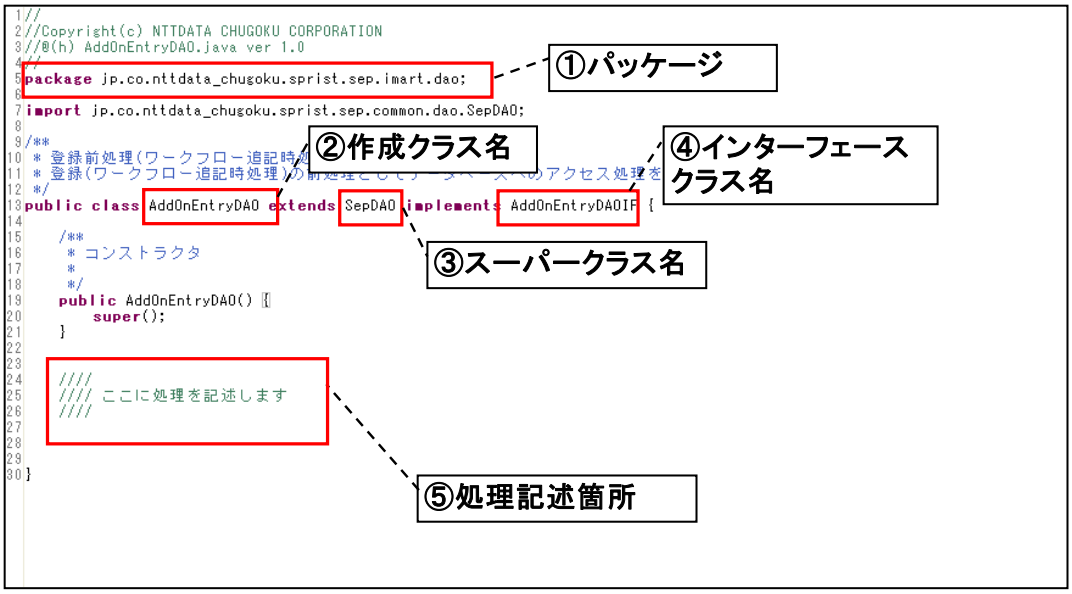


図 7.　DAO クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.imart.dao」を指定する
② クラス名を「AddOnEntryDAO」とする
③ スーパークラス名に「SepDAO」を指定する
④ インターフェースクラス名に「AddOnEntryDAOIF」を指定する
⑤ 実際の処理を記述する。

3. ワークフロー後処理

3.1 ワークフロー後処理の追加

intra-mart の標準ワークフロー機能への起票、承認、否認、差戻、引戻、取止、破棄の処理を行う場合に、ワークフロー後処理を作成し実行することができます。

(1) EX 申請システムの標準のワークフロー後処理

EX 申請システムであらかじめ用意されている標準の後処理では、ワークフローで行われた処理（起票、承認、否認、差戻、引戻、取止、破棄）に伴い、その処理内容を登録データへ反映する処理が行われます。

プロセス定義を作成する際に、指定の後処理を設定することにより、ワークフローと登録データが連携することができます。（設定方法についてはシステム管理者編ワークフローを参照）

(2) ワークフロー後処理の作成

EX 申請システムの特定の箇所に処理を記述することにより、後処理を追加することができます。

ワークフロー後処理の作成は JavaEE 開発モデル、JavaScript のいずれにおいても処理を記述することができます。

3.2 ワークフロー後処理の構成

プロセス定義に設定されたタスク定義の種類により、実行するワークフロー後処理が異なります。

後処理の種類は申請後処理、第一承認後処理、最終承認後処理、中間承認後処理、第一最終承認後処理の 5 種類があります。

また、ワークフローで行われた処理により実行される処理（メソッド）が異なります。

(1) タスクの種類

表6.　タスクの種類

タスク	設定する内容
申請	申請（起票）を行うタスク
第一承認	申請（起票）を行ったタスクの次に（一番目に）承認処理を行うタスク
中間承認	第一承認者と最終承認者の間に承認処理を行うタスク（第一でも最終承認者でもないタスク）
最終承認	最後の承認処理を行うタスク（承認を行うとワークフローが完了するタスク）
第一最終承認	第一承認者と最終承認者の両方を兼ねた承認処理タスク（承認タスクが一つしかない場合）

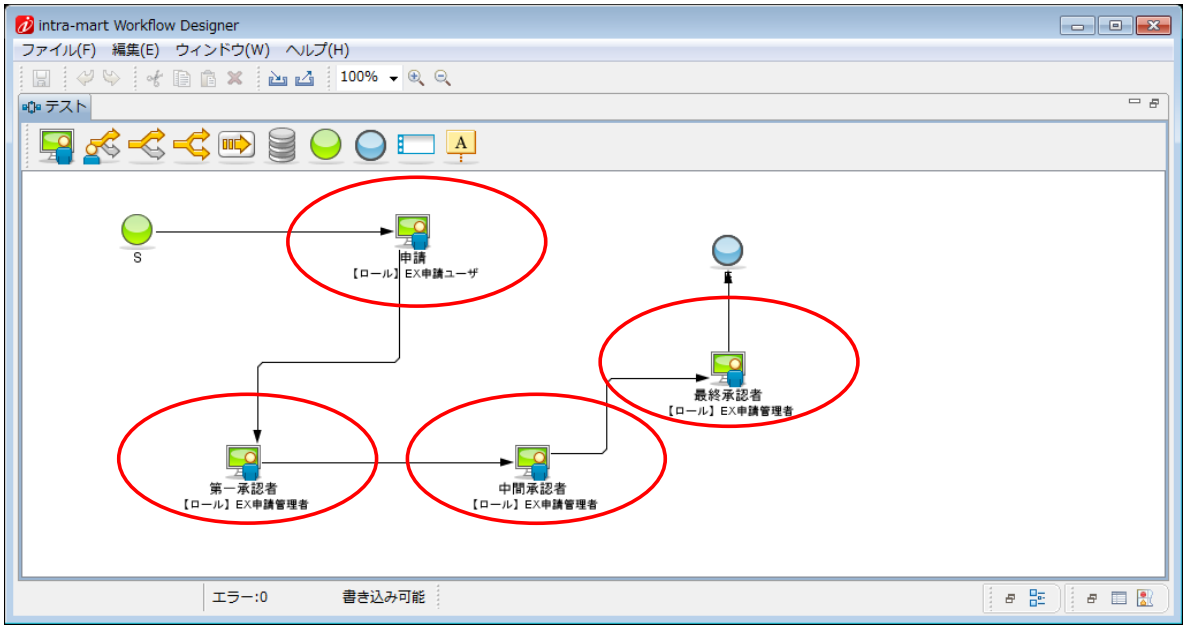


図 8.　プロセス定義（承認者３人）

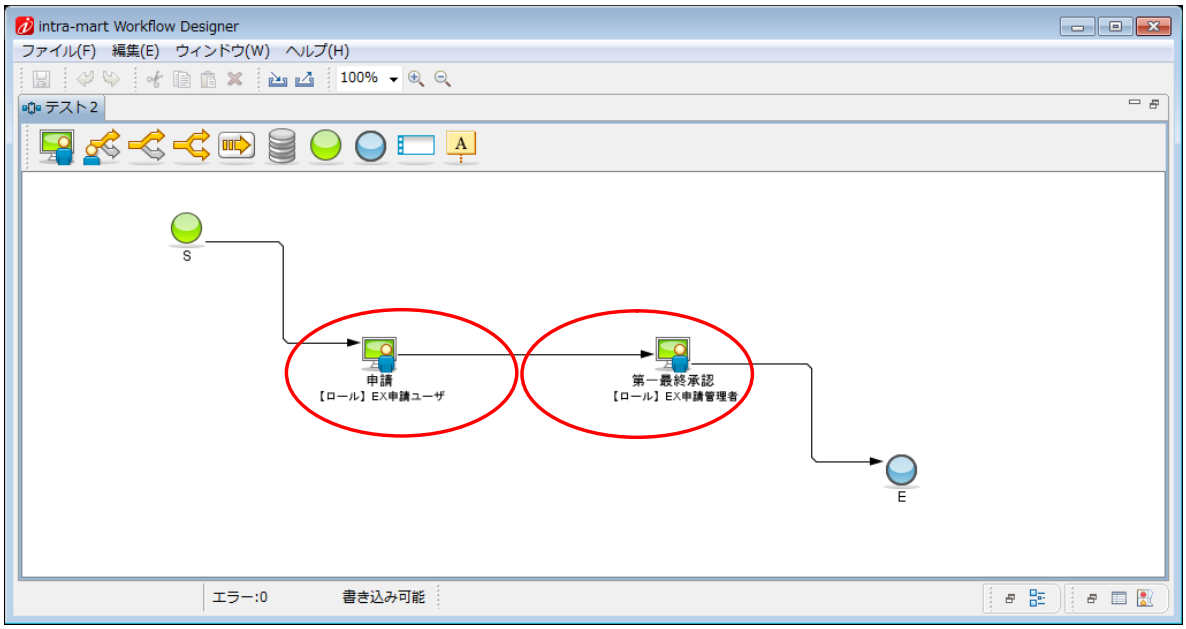


図 9. プロセス定義（承認者１人）

(2) ワークフロー承認時の処理の種類

申請者や承認者が行う処理の種類です。

表7. 処理の種類

処理	設定する内容
申請/承認	申請・再申請・承認を行った場合に実行されるメソッド
引戻	引戻を行った場合に実行されるメソッド
承認者へ差戻	承認者へ差戻を行った場合に実行されるメソッド
申請者へ差戻	申請者へ差戻を行った場合に実行されるメソッド
取止/ここで終了	取止/ここで終了を行った場合に実行されるメソッド
否認	否認を行った場合に実行されるメソッド

(3) システム構成図

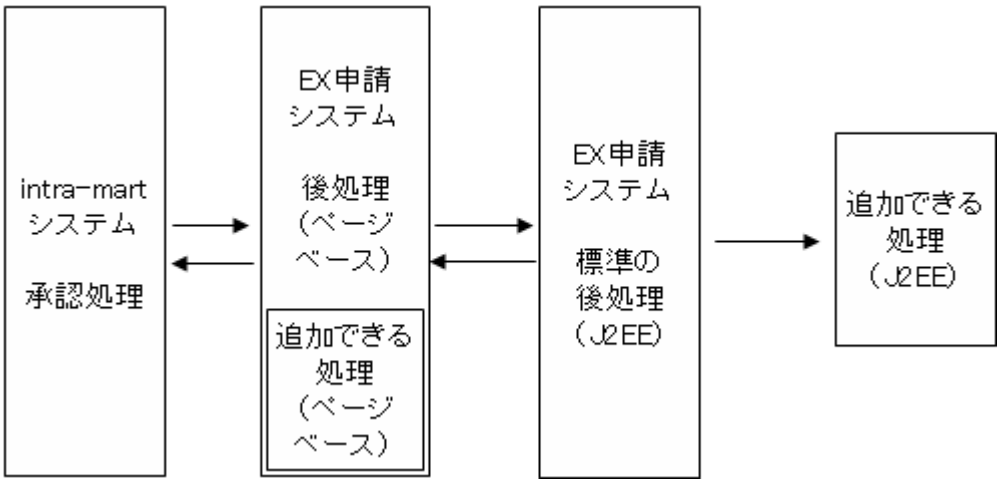


図 10.　ワークフロー後処理構成図

3.3 JavaEE 開発モデルでワークフロー後処理を追加する

EX 申請システムでは EventListener クラスを用意しています。

それ以外の JAVA クラスについては処理を記述することはできません。

3.3.1 作成する JAVA クラス

EX 申請システムの標準のワークフロー後処理から下表に示す JAVA クラスが呼び出され実行されます。
登録前処理を作成する場合、対象の処理に合わせて下表に示す JAVA クラスを作成します。

表8.　呼び出される JAVA クラス

JAVA クラス名	説明
jp.co.nttdata_chugoku.sprist.sep.imart.event.BPWAddOnEventListener	ワークフローの後処理を記述する

3.3.2 JAVA クラスの作成方法

(1) EventListener を作成する

EX 申請システムに用意されたクラス「SepEventListener」を継承し、EventListener を作成します。

EventListener クラスはワークフロー後処理の BPW 内から呼び出され実行されます。

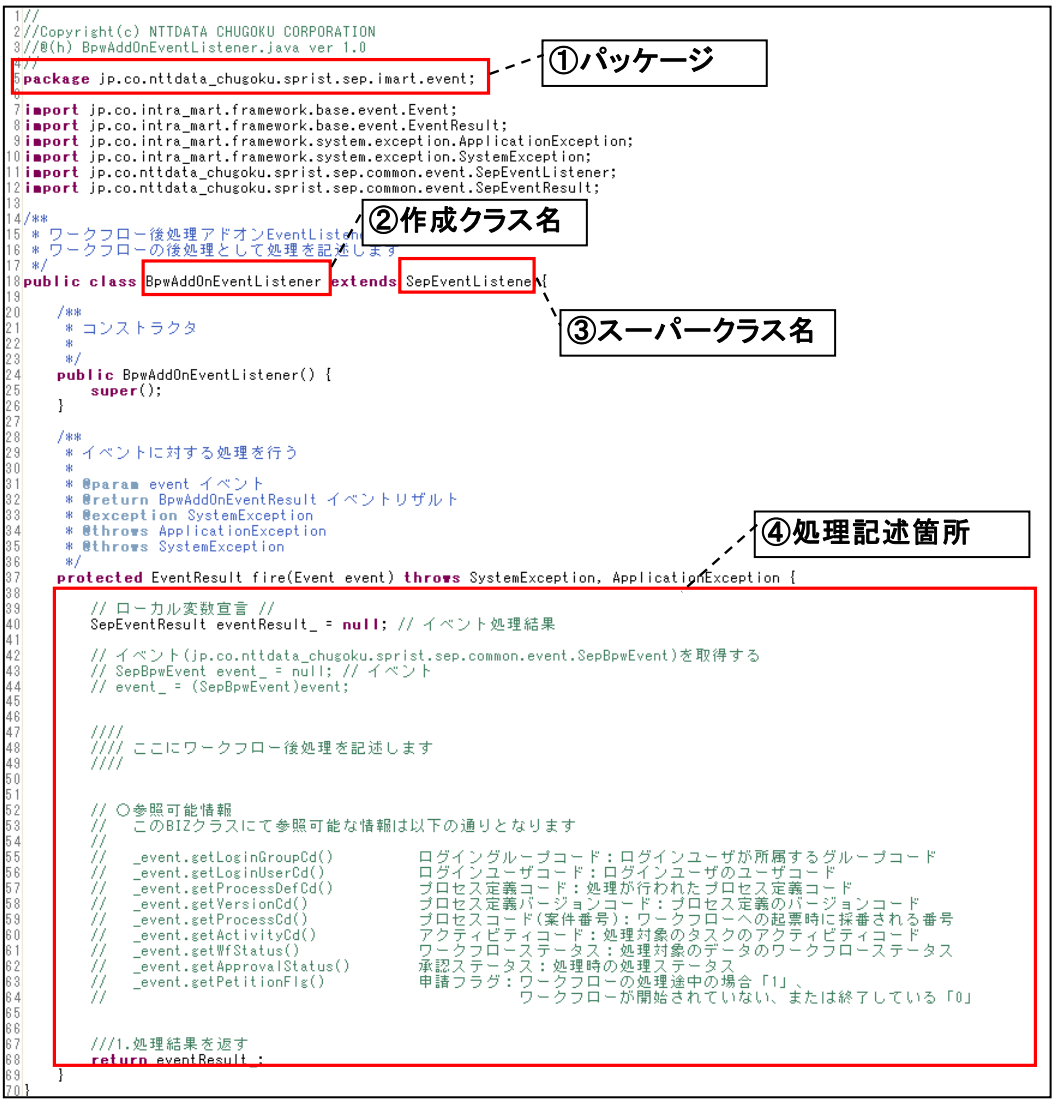
以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：BPWAddOnEventListener
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.imart.event
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepEventListener
説明：ワークフローの後処理を記述するクラス

[継承しなければならないクラス]

クラス名：SepEventListener
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.event
説明：EX 申請システムの EventListener スーパークラス




```
//
//Copyright(c) NTTDATA CHUGOKU CORPORATION
//@(h) BpwAddOnEventListener.java ver 1.0
//
package jp.co.nttdata_chugoku.sprist.sep.imart.event;

import jp.co.intra_mart.framework.base.event.Event;
import jp.co.intra_mart.framework.base.event.EventResult;
import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepEventListener;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepEventResult;

/**
 * ワークフロー後処理アドオンEventListener
 * ワークフローの後処理として処理を記述します
 */
public class BpwAddOnEventListener extends SepEventListener{

    /**
     * コンストラクタ
     *
     */
    public BpwAddOnEventListener() {
        super();
    }

    /**
     * イベントに対する処理を行う
     *
     * @param event イベント
     * @return BpwAddOnEventResult イベントリザルト
     * @exception SystemException
     * @throws ApplicationException
     * @throws SystemException
     */
    protected EventResult fire(Event event) throws SystemException, ApplicationException {

        // ローカル変数宣言 //
        SepEventResult eventResult_ = null; // イベント処理結果

        // イベント(jp.co.nttdata_chugoku.sprist.sep.common.event.SepBpwEvent)を取得する
        // SepBpwEvent event_ = null; // イベント
        // event_ = (SepBpwEvent)event;


        ////
        //// ここにワークフロー後処理を記述します
        ////


        // ○参照可能情報
        //   このBIZクラスにて参照可能な情報は以下の通りとなります
        //
        //   _event.getLoginGroupCd()          ログイングループコード：ログインユーザが所属するグループコード
        //   _event.getLoginUserCd()           ログインユーザコード：ログインユーザのユーザコード
        //   _event.getProcessDefCd()          プロセス定義コード：処理が行われたプロセス定義コード
        //   _event.getVersionCd()             プロセス定義バージョンコード：プロセス定義のバージョンコード
        //   _event.getProcessCd()             プロセスコード(案件番号)：ワークフローへの起票時に採番される番号
        //   _event.getActivityCd()            アクティビティコード：処理対象のタスクのアクティビティコード
        //   _event.getWfStatus()              ワークフローステータス：処理対象のデータのワークフローステータス
        //   _event.getApprovalStatus()        承認ステータス：処理時の処理ステータス
        //   _event.getPetitionFlg()           申請フラグ：ワークフローの処理途中の場合「1」、
        //                                                 ワークフローが開始されていない、または終了している「0」
        //


        ///1.処理結果を返す
        return eventResult_;
    }
}
```

図 11.　EventListener クラス実装例

① **パッケージ名**「jp.co.nttdata_chugoku.sprist.sep.imart.event」**を指定する**

② JAVA **クラス名を**「BpwAddOnEventListener」**とする**

③ **スーパークラス名に**「SepEventListener」**を指定する**

④ **実際の処理を記述する。**SepBPWEvent **から取得できる情報については下表参照**

※引数として取得する Event クラス

パッケージ「jp.co.nttdata_chugoku.sprist.sep.common.event」

クラス名「SepBpwEvent」

Event クラスから情報を取得する場合、event を「SepBpwEvent」にキャストして取得してください。

表9.　SepBpwEvent から取得できる情報

メソッド名	取得する値
getLoginGroupCd	ログイングループコード
getLoginUserCd	ログインユーザコード
getProcessDefCd	プロセス定義コード
getProcessCd	プロセスコード
getVersionCd	プロセス履歴コード
getActivityCd	アクティビティコード
getWfStatus	ワークフローステータス
getApprovalStatus	承認ステータス
getPetitionFlg	申請フラグ

3.4 スクリプト開発モデルで後処理を追加する

EX 申請システムでは標準の後処理、追加する後処理の全ての JAVA クラスを JavaScript から呼出し実行しています。この JavaScript に処理を追加することによりスクリプト開発モデルでの後処理の追加ができます。

※用意している JavaScript は intra-mart EX 申請の標準のワークフロー後処理を呼出し実行しています。
この標準のワークフロー後処理の呼び出しを行っている箇所は削除しないでください。万が一削除した場合、EX 申請システムのワークフロー後処理は正常に動作しません。

3.4.1 JavaScript の作成方法

タスクの種類により、5 種類の後処理が用意されています。

表10.　スクリプト開発モデル後処理用 JavaSctipst ファイル

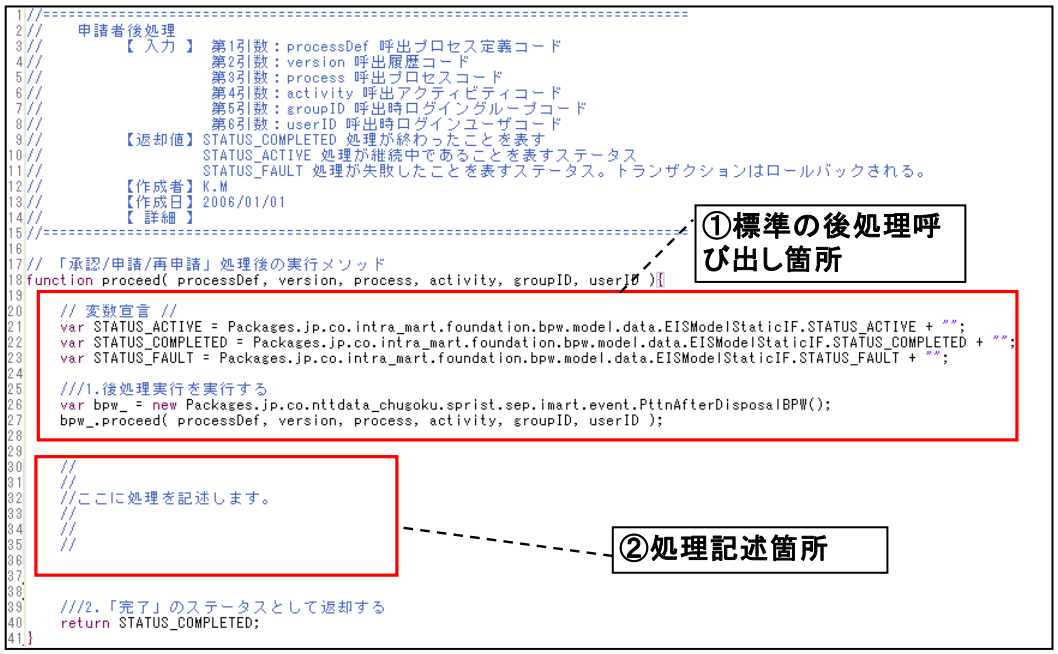
タスク	後処理 JavaScript ファイル名
申請	pttn_after_disposal_bpw.js
第一承認	first_approval_after_disposal_bpw.js
中間承認	middle_approval_after_disposal_bpw.js
最終承認	last_approval_after_disposal_bpw.js
第一最終承認	first_last_approval_after_disposal_bpw.js

(1) ワークフロー後処理の記述

ワークフロー後処理の追加を行う各タスク用の後処理用 JavaScript ファイルのメソッドに処理を記述します。

表11.　メソッド

処理	メソッド名
承認・申請	proceed
引戻	pullDraft
承認者へ差戻	returnDraft
申請者へ差戻	returnStart
取止/ここで終了	compulsion
否認	rejection




```
//=============================================================================
//    申請者後処理
//          【 入力 】  第1引数：processDef 呼出プロセス定義コード
//                      第2引数：version 呼出履歴コード
//                      第3引数：process 呼出プロセスコード
//                      第4引数：activity 呼出アクティビティコード
//                      第5引数：groupID 呼出時ログイングループコード
//                      第6引数：userID 呼出時ログインユーザコード
//          【返却値】STATUS_COMPLETED 処理が終わったことを表す
//                      STATUS_ACTIVE 処理が継続中であることを表すステータス
//                      STATUS_FAULT 処理が失敗したことを表すステータス。トランザクションはロールバックされる。
//          【作成者】K.M
//          【作成日】2006/01/01
//          【 詳細 】
//=============================================================================

// 「承認/申請/再申請」処理後の実行メソッド
function proceed( processDef, version, process, activity, groupID, userID ){

    // 変数宣言 //
    var STATUS_ACTIVE = Packages.jp.co.intra_mart.foundation.bpw.model.data.EISModelStaticIF.STATUS_ACTIVE + "";
    var STATUS_COMPLETED = Packages.jp.co.intra_mart.foundation.bpw.model.data.EISModelStaticIF.STATUS_COMPLETED + "";
    var STATUS_FAULT = Packages.jp.co.intra_mart.foundation.bpw.model.data.EISModelStaticIF.STATUS_FAULT + "";

    ///1.後処理実行を実行する
    var bpw_ = new Packages.jp.co.nttdata_chugoku.sprist.sep.imart.event.PttnAfterDisposalBPW();
    bpw_.proceed( processDef, version, process, activity, groupID, userID );


    //
    //
    //ここに処理を記述します。
    //
    //
    //


    ///2.「完了」のステータスとして返却する
    return STATUS_COMPLETED;
}
```

図 12.　ワークフロー後処理 JavaScript ファイル実装例

① EX 申請システムの標準のワークフロー後処理を実行箇所です。

② 実際の処理を記述します。

4. ワークフロー審議処理（追記）

4.1 ワークフロー審議処理（追記）の追加

ワークフローで審議ボタンをクリックする際に呼ばれる JavaScript から追記登録を行うワークフロー審議処理(追記）API を呼ぶことで、追記登録ボタンをクリックしなくても登録処理を行うことができます。

以下に用意する API について示します。

[用意する API]

JavaScript ファイル名 ：<imAR インストール先>/sprist/sep/csjs/imart/pttn_particular_head.js
関数：onClickApprovalEntry
説明：ワークフロー審議処理（追記）

(1) ワークフロー審議処理（追記）の作成

審議処理を行っている JSP ファイル（AcknowledgeBody.jsp）の審議ボタンクリック時に呼んでいる JavaScript で、ワークフロー審議処理（追記）の API を呼びます。

例

```
<script language="javascript">
function approvalEntry(){
    try{
        // 登録処理を実行します（申請詳細画面が開かれているウィンドウの「onClickApprovalEntry()」を実行します）
        if( parent.DetailAcknowledge_FOD.onClickApprovalEntry()){
            // 登録処理が正常に終了した場合の処理を記述します
            handleEvent('','','save','');
        }
    }catch(e){
        // 登録処理が異常終了した場合の処理を記述します
        handleEvent('','','save','');
    }
}
</script>
        :
        :
        :
        <tr>
                <td class="button_padding">
                        <img src="images/standard/button_left.gif"></td>
                <td class="button_bg">
                        <input name="save" type="button" class="button_bg"
                                value='<fw_tag:Message application="bpw"
                                key="DetailAcknowledge.DetailInfoButtonApprove"></fw_tag:Message>'
                                onClick="approvalEntry()">
                </td>
                <td class="button_padding">
                        <img src="images/standard/button_right.gif"></td>
        </tr>
```

①APIを呼ぶ
②登録処理後の処理記述箇所
③異常処理後の処理記述箇所
④審議ボタンクリック時にjavascriptを呼ぶ

図 13.　ワークフロー審議処理（追記）JSP ファイル実装例

5. ワークフロー操作後処理（削除）

5.1 ワークフロー操作後処理（削除）の追加

ワークフローの削除時に申請書データを削除しデータの同期を取ることができます。

申請書データ削除を行うには、プロセス操作画面にて実施される処理に申請書データ削除の機能を呼び出す JavaScript を追加します。

(1) ワークフロー機能のプロセス操作画面の編集

プロセス操作画面の JSP ファイル

[インストール先]/doc/imart/bpw/business/DeleteOldTask/DeleteOldTaskFrame.jsp

のフレーム構成にフレームセットを１つ追加します。

```
<HTML>
    <HEAD>
        <TITLE>
        </TITLE>
    </HEAD>
    <FRAMESET ROWS="100,*,0" frameborder="0">
            <!-- ヘッダ -->
            <fw_tag:Frame application="bpw" service="delete_old_task_head_call" name="DeleteOldTask_HED">
            </fw_tag:Frame>
            <!-- ボディ -->
            <fw_tag:Frame application="bpw" service="delete_old_task_body_call" name="DeleteOldTask_BOD">
            </fw_tag:Frame>
            <frame src="" name="DeleteOldTask_EX" />
    </FRAMESET>
</HTML>
```

①フレームサイズの追加箇所

②フレームの追加箇所

図 14.　プロセス操作画面 JSP ファイル実装例

(2) ワークフロー操作後処理（削除）の追加

プロセス操作処理を行っている JSP ファイルを編集します。

[インストール先]/doc/imart/bpw/business/DeleteOldTask/DeleteOldTaskBody.jsp

①案件削除ボタンクリック時に使用している JavaScript にワークフロー削除後処理（削除）を追加します。

```
        //================================================
        //    案件の削除
        //          【入力】 なし
        //          【返却】 なし
        //          【作成】
        //          【更新】
        //          【概要】
        //================================================
        function deleteTask()
        {
            if( deleteFlag == "true" )
            {
                if( confirm(document.alertForm.ConDeleteOldTask.value) )
                {
                    // フラグの更新
                    deleteFlag = "false";

                    // EX申請　申請書データ削除
                    document.deleteExData.submit();

                    // 削除(成功:一覧に戻る 失敗：再表示)
                    document.deleteForm.submit();

                }
                else
                {
                    return;
                }
            }
        }
```

①申請書データ削除の
リクエスト処理記述箇所

図 15.　ワークフロー操作後処理（削除）JSP ファイル実装例①

②ワークフロー削除後処理（削除）機能のフォームを追加します。

```
144         <!--削除フォーム-->↓
145         <fw_tag:Form application="bpw" service="delete_active_activity" name="deleteForm" method="POST">↓
146         </fw_tag:Form>↓
147         <!--アラートフォーム-->↓
148         <FORM name="alertForm">↓
149             <!--確認-->↓
150             <INPUT type="hidden" name="ConDeleteOldTask" value='<%=dele                    DeleteOldTask.C
151             <INPUT type="hidden" name="ConUpdateOldTask" value='<%=delete_old_task.getMessage( DeleteOldTask.C
152         </FORM>↓
153         <!-- EX申請フォーム -->↓
154         <fw_tag:Form application="sep_imart" service="input_form_data_manager_sid" ↓
155             name="deleteExData" method="POST" target="DeleteOldTask_EX">↓
156             <input type="hidden" name="hdnINPUT_PROCESS_CD" value="<%= delete_old_task.getProcessCd() %>">↓
157         </fw_tag:Form>↓
158     </BODY>↓
159 </HTML>↓
```

①申請書データ削除の
リクエスト設定記述箇所

図 16.　ワークフロー操作後処理（削除）JSP ファイル実装例②

6. 登録データ検索削除後処理

6.1 登録データ検索削除後処理の追加

(1) Biz クラスの作成

EX 申請システムに用意された JAVA クラス「SepBiz」を継承し、Biz クラスを作成します。
Biz クラスは登録処理の EventListener 内から呼び出され、データベースへの登録前に実行されます。
この Biz クラスに登録前処理のロジックを記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]
クラス名：AddOnEntryBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.event
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz
説明：登録・訂正の前処理を記述するクラス

[継承しなければならない JAVA クラス]
クラス名：SepBiz
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.event
説明：EX 申請システムの Biz スーパークラス


```
//
//Copyright(c) NTTDATA CHUGOKU CORPORATION
//@(h) AddOnInputFormDeleteBiz.java ver 1.0
//
package jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.event;

import jp.co.intra_mart.framework.system.exception.ApplicationException;
import jp.co.intra_mart.framework.system.exception.SystemException;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepBiz;
import jp.co.nttdata_chugoku.sprist.sep.common.event.SepEvent;
import jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteDAOIF;


/**
 * 削除後処理(登録データ検索削除処理)アドオンBizクラス
 * 登録データ検索の削除処理の後処理として処理を記述します
 */
public class AddOnInputFormDeleteBiz extends SepBiz {

    /**
     * コンストラクタ
     *
     * @param _event イベント
     */
    public AddOnInputFormDeleteBiz(SepEvent _event) {
        super(_event);
    }

    /**
     * 削除後処理(登録データ検索削除処理)アドオンクラス
     *
     * @param _daoIf 削除前処理アドオン用のDAOIF
     * @param _event EventListenerより渡されるEventクラス
     * @throws SystemException
     * @throws ApplicationException
     */
    public void service(AddOnInputFormDeleteDAOIF _daoIf, AddOnInputFormDeleteEvent _event)
        throws SystemException, ApplicationException {

        ////
        //// ここに処理を記述します
        ////

        // ○参照可能情報
        //   このBIZクラスにて参照可能な情報は以下の通りとなります
        //
        //   _event.getDataEntryDiv()          データ登録区分：データ登録時「01」、データ取消時「02」
        //   _event.getPttnNo()                申請番号：EX申請システム内でデータを参照する為のキー情報
        //   _event.getInputFormCd()           入力様式コード：入力様式マスタのキー
        //   _event.getInputFormVersionCd()    入力様式バージョンコード：入力様式マスタのキー
        //   _event.getInputFormEntryUserCd()  入力様式登録者コード：対象のユーザコード
        //   _event.getCompanyCd()             会社コード：対象の会社コード
        //   _event.getOrgnCd()                所属コード：対象の所属コード
        //   _event.getEntryDate()             登録日
        //   _event.getEntryUserCd()           登録者コード
        //   _event.getLastUpdateUserCd()      最終更新者コード
        //   _event.getLastUpdateYmdhms()      最終更新日
        //   _event.getSessionId()             セッションID
        //   _event.getDataEntryDiv()          データ登録区分(00：登録、01：取消)
        //
        //
        // ○Exceptionのスロー
        //   意図的に処理を中断させる場合はApplicationExceptionをスローしてください。
        //   但し、ワークフロー追記の場合の登録前処理にてApplicationExceptionをスローしても
        //   アプリケーションエラー画面は表示されません。イントラマートの標準ワークフロー機能での
        //   エラー画面が表示されます。
        //
        //   (実装例)
        //   ApplicationExceptionをスローする
        //   /*
        //    * @param String エラー画面表示メッセージ
        //    */
        //   throw new ApplicationException("マスタに存在しない値が入力されています。");
        //
        //   (実装例)
        //   SystemExceptionをスローする
        //   /*
        //    * @param String エラーメッセージ
        //    * @param Throwable 例外
        //    */
        //   throw new SystemException("データベースとの接続でI/O例外エラーが発生しました。", e.getCause());
        //
        //   (注)
        //   ページベースの後処理を使用する場合、AddOnEntryEventからログインユーザコード等の情報を取得することはできませ
        //   ログインユーザコード・ログイングループIDを取得するには、AddOnEntryEventクラスの以下のメソッドを使用してくだ
        //    getLoginUserCd()
        //    getLoginGroupCd()

    }

}
```

図 17.　Biz クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.event」を指定する

② JAVA クラス名を「AddOnInputFormDeleteBiz」とする

③ スーパークラス名に「SepBiz」を指定する

④ 実際の処理を記述する。AddOnInputFormDeleteEvent クラスから取得可能なパラメータについては、下表参照

表12.　AddOnInputFormDeleteEvent クラスから取得できる情報

メソッド名		取得する値	備考
メソッド名	戻り値のデータ型		
getDataEntryDiv()	String	登録区分	データ登録時「01」、データ取消時「02」
getPttnNo()	String	申請番号	登録時に採番される一意な番号 ※1
getInpurFormCd()	String	入力様式コード	入力様式マスタ内での一意なコード
getInputFormVersionCd()	String	入力様式バージョンコード	-
getInputFormEntryUserCd()	String	入力様式登録者コード	入力様式の登録対象者ユーザ
getCompanyCd()	String	会社コード	入力様式登録者の会社コード
getOrgnCd()	String	所属部門コード	入力様式登録者コードの所属部門コード
getEntryDate()	String	登録日	- ※1
getEntryUserCd()	String	登録者コード	- ※1
getLastUpdateUserCd()	String	最終更新者コード	- ※1
getLastUpdateYmdhms()	String	最終更新日	- ※1
getSessionId()	String	セッションＩＤ	-

※１　新規登録時には空文字が返却されます

(2) DAOIF クラスの作成

EX 申請システムに用意されたクラス「SepDAOIF」を継承し、DAOIF クラスを作成します。
DAOIF クラスは登録処理の EventListener 内にて生成され、Biz クラスの引数として渡されます。
また、このクラスは「AddOnInputFormDeleteDAO」クラスのインターフェースクラスとなります。

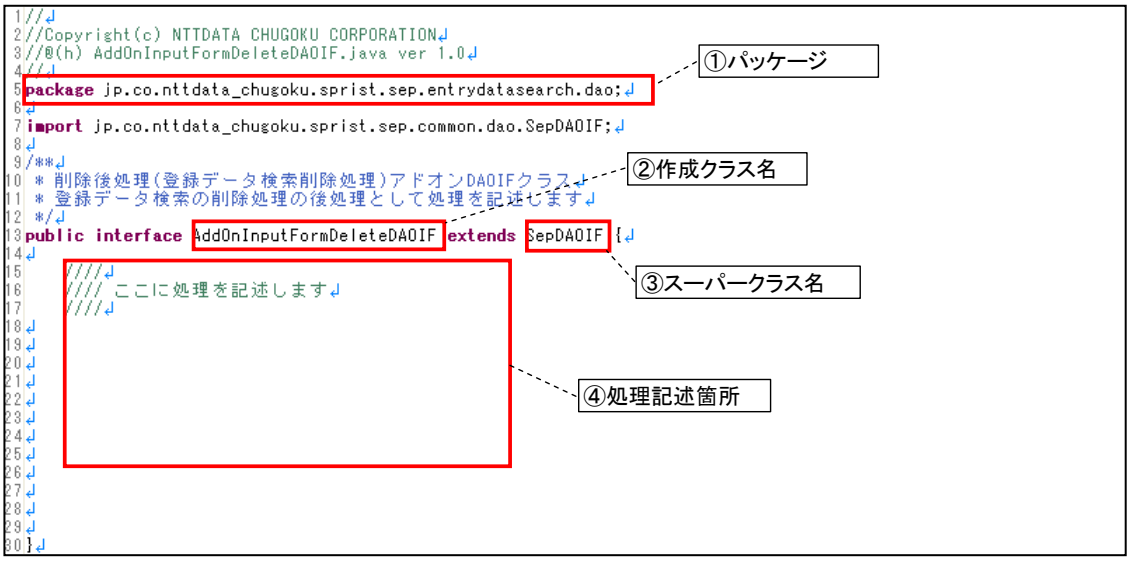
以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnInputFormDeleteDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAOIF
説明：DAO クラスのインターフェースクラス

[継承しなければならない JAVA クラス]

クラス名：SepDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAOIF スーパークラス




```
1 //
2 //Copyright(c) NTTDATA CHUGOKU CORPORATION
3 //@(h) AddOnInputFormDeleteDAOIF.java ver 1.0
4 //
5 package jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao;
6
7 import jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAOIF;
8
9 /**
10  * 削除後処理(登録データ検索削除処理)アドオンDAOIFクラス
11  * 登録データ検索の削除処理の後処理として処理を記述します
12  */
13 public interface AddOnInputFormDeleteDAOIF extends SepDAOIF {
14
15     ////
16     //// ここに処理を記述します
17     ////
18
19
20
21
22
23
24
25
26
27
28
29
30 }
```

図 18.　DAOIF クラス実装例

② パッケージ名「jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao」を指定する
③ クラス名を「AddOnInputFormDeleteDAOIF」とする
④ スーパークラス名に「SepDAOIF」を指定する
⑤ 実際の処理を記述する。

(3) DAO クラスの作成

EX 申請システムに用意されたクラス「SepDAO」を継承し、DAO クラスを作成します。
この DAO クラスにデータベースへのアクセス処理を記述します。

以下に作成する JAVA クラスについて示します。

[作成する JAVA クラス]

クラス名：AddOnInputFormDeleteDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao
継承クラス：jp.co.nttdata_chugoku.sprist.sep.common.dao.SepDAO
説明：データベースに対するアクセス処理を記述するクラス

[継承しなければならないクラス]

クラス名：SepDAO
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAO スーパークラス

[実装しなければならないインターフェースクラス]

クラス名：AddOnInputFormDeleteDAOIF
パッケージ名：jp.co.nttdata_chugoku.sprist.sep.common.dao
説明：EX 申請システムの DAOIF 実装クラス

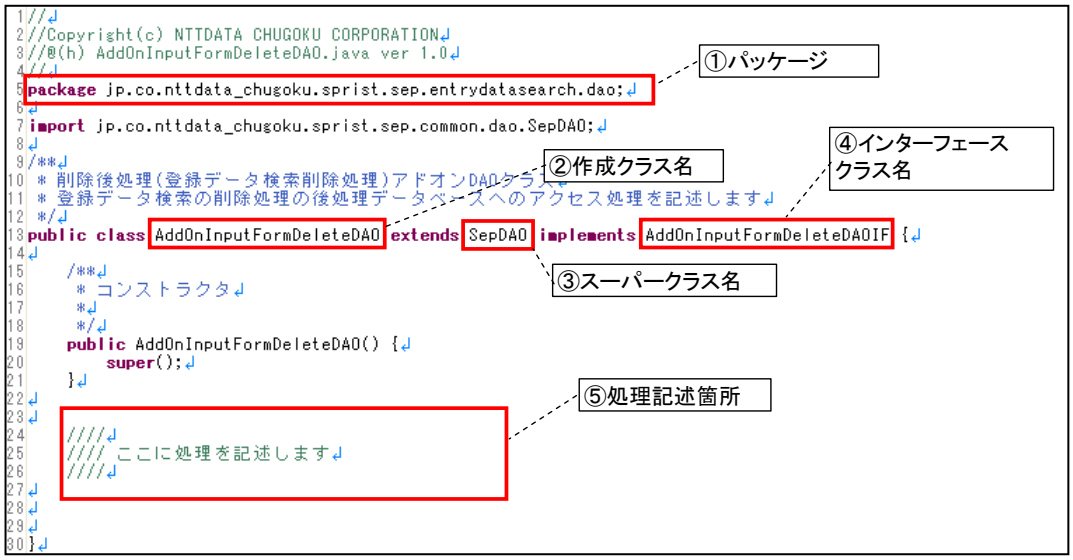


図 19.　DAO クラス実装例

① パッケージ名「jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao」を指定する
② クラス名を「AddOnInputFormDeleteDAO」とする
③ スーパークラス名に「SepDAO」を指定する
④ インターフェースクラス名に「AddOnInputFormDeleteDAOIF」を指定する
⑤ 実際の処理を記述する。

7. エラー発生時の記述方法

7.1 Exception について

Biz に記述している処理内において意図的にエラー画面を表示させる場合、Exception をスローする記述を行ってください。Exception をスローすることにより画面にエラー通知メッセージを表示させ、ユーザに再入力を促すなどの処理を記述することができます。以下にスローする Exception を示します。

表13.　スローする Exception

Exception 名	説明	遷移先画面	
		登録・訂正処理	ワークフロー承認処理
ApplicationException	アプリケーションエラーが発生した場合にスローします。 例） マスタとの整合性チェックエラー時	EX 申請システム標準のアプリケーションエラー表示画面 ※下図「アプリケーションエラー表示画面」参照	intra-mart の標準ワークフロー機能のエラー表示画面
SystemException	システムエラーが発生した場合にスローします。 例） データベースとの接続エラー時	EX 申請システム標準のシステムエラー表示画面 ※下図「システムエラー表示画面」参照	intra-mart の標準ワークフロー機能のエラー表示画面

また、Exception をスローする場合、以下に示す JAVA クラスのインポートを行ってください。

表14.　インポートするパッケージ

Exception 名	パッケージ
ApplicationException	jp.co.intra_mart.framework.system.exception.ApplicationException
SystemException	jp.co.intra_mart.framework.system.exception.SystemException

図 20.　アプリケーションエラー表示画面

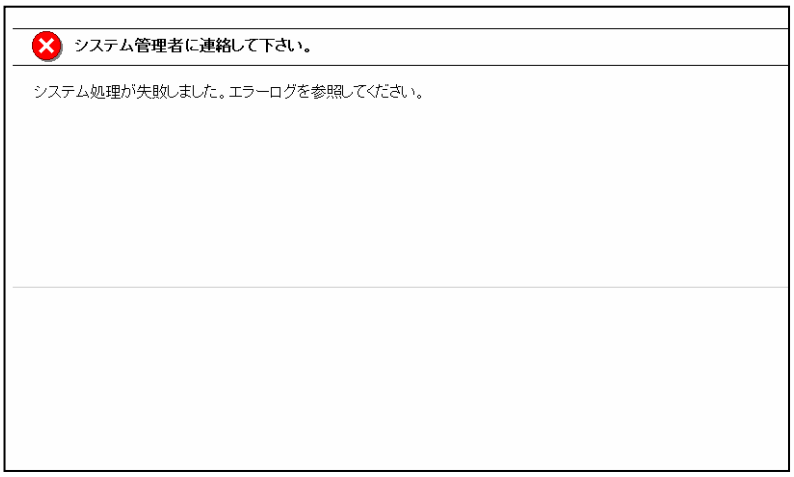


図 21.　システムエラー表示画面

図 22.　intra-mart の標準ワークフロー機能のエラー表示画面

7.2 Exception の記述方法

Exception をスローする場合、下図のような記述を行ってください。

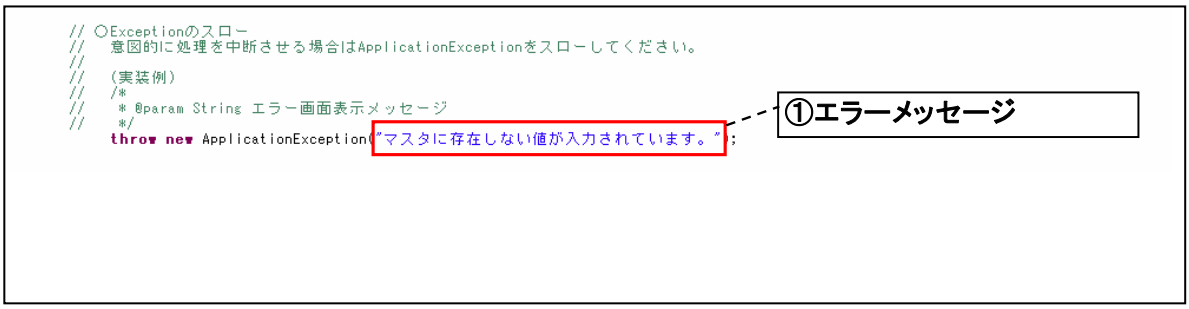


図 23. ApplicationException のスロー記述例

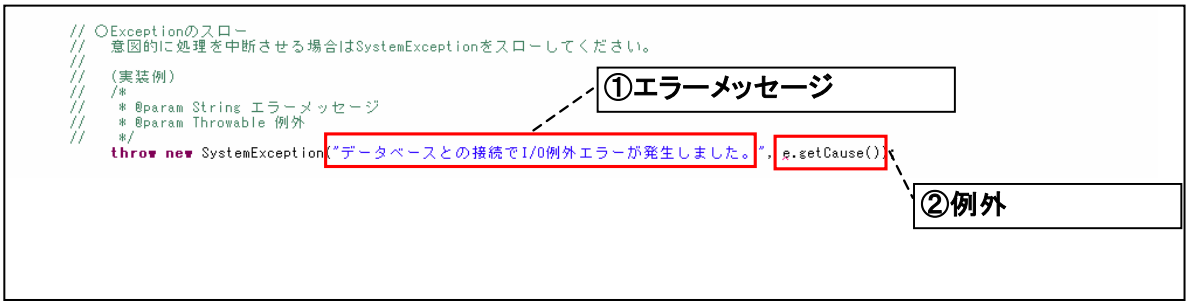


図 24. SystemException のスロー記述例

8. 作成したファイルの配置

JavaEE 開発モデルにて処理を記述したファイルはEX 申請システムがインストールされているサーバの特定の場所に配置する必要があります。以下に配置が必要なファイルを示します。

表15.　配置が必要なファイル

処理名	JAVA クラス名
登録・訂正	jp.co.nttdata_chugoku.sprist.sep.formselect.event.AddOnEntryBiz
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAO
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAOIF
ワークフロー承認処理	jp.co.nttdata_chugoku.sprist.sep.imart.event.AddOnEntryBiz
	jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAO
	jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAOIF
ワークフロー後処理	jp.co.nttdata_chugoku.sprist.sep.imart.event.BPWAddOnEventListener
登録データ検索削除後処理	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteBiz
	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteDAO
	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteDAOIF

※JavaScript ファイルについては、インストール後の配置場所を変更しないでください。配置場所を変更した場合、ワークフローの後処理は正常に動作しません。

8.1 クラスファイルを配置する

作成した JAVA ファイルは配置する前にあらかじめコンパイルしておいてください。
以下に JAVA クラスとその配置場所を示します。

表16.　JAVA クラス名と配置場所

処理名	JAVA クラス名	配置場所
登録・訂正	jp.co.nttdata_chugoku.sprist.sep.formselect.event.AddOnEntryBiz	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep /formselect/event/ 配下
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAO	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep /formselect/dao/ 配下
	jp.co.nttdata_chugoku.sprist.sep.formselect.dao.AddOnEntryDAOIF	
ワークフロー承認処理	jp.co.nttdata_chugoku.sprist.sep.imart.event.AddOnEntryBiz	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep/imart/event/ 配下
	Jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAO	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep/imart/dao/ 配下
	jp.co.nttdata_chugoku.sprist.sep.imart.dao.AddOnEntryDAOIF	
ワークフロー後処理	jp.co.nttdata_chugoku.sprist.sep.imart.event.BPWAddOnEventListener	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep/imart/event/ 配下
登録データ検索削除後処理	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteBiz	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep /entrydatasearch/event/ 配下

	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteDAO	%RESIN_HOME%/webapps/{アプリケーション名} /WEB-INF/classes/jp/co/nttdata_chugoku/sprist/sep /entrydatasearch/dao/ 配下
	jp.co.nttdata_chugoku.sprist.sep.entrydatasearch.dao.AddOnInputFormDeleteDAOIF	

9. 補足資料

後処理の開発を行う際の補足資料です。

9.1 ステータス・フラグ

EX 申請システム内で使用しているワークフローとの連携用のステータスとフラグです。

(1) ワークフローステータス

ステータス名	システム内部でのコード
未申請	01
承認待	02
差戻	03
引戻	04
承認	11
否認	12
取止	13

(2) 承認ステータス

ステータス名	システム内部でのコード
申請	01
承認	02
引戻	03
差戻	04
申請者へ差戻	05
否認	06
ここで終了（承認）	07
ここで終了（否認）	08
取止	09

(3) 申請フラグ

ステータス名	システム内部でのコード
申請中	1
未申請	0

9.2 後処理とステータス・フラグ

ワークフロー後処理から EX 申請システムで登録したデータに反映するステータス・フラグの内容です。

(1) 申請後処理

処理	ワークフロー ステータス	承認ステータス	申請フラグ
承認・申請	承認待	申請	申請中
引戻	引戻	引戻	申請中
承認者へ差戻	-	-	-
申請者へ差戻	-	-	-
取止/ここで終了	取止	取止	未申請
否認	-	-	-

(2) 第一承認後処理

処理	ワークフロー ステータス	承認ステータス	申請フラグ
承認・申請	承認待	承認	申請中
引戻	承認待	引戻	申請中
承認者へ差戻	差戻	差戻	申請中
申請者へ差戻	差戻	申請者へ差戻	申請中
取止/ここで終了	承認	ここで終了（承認）	未申請
否認	否認	否認	未申請

(3) 中間承認後処理

処理	ワークフロー ステータス	承認ステータス	申請フラグ
承認・申請	承認待	承認	申請中
引戻	承認待	引戻	申請中
承認者へ差戻	承認待	差戻	申請中
申請者へ差戻	差戻	申請者へ差戻	申請中
取止/ここで終了	承認	ここで終了（承認）	未申請
否認	否認	否認	未申請

(4) 最終承認後処理

処理	ワークフロー ステータス	承認ステータス	申請フラグ
承認・申請	承認	承認	未申請
引戻	-	-	-
承認者へ差戻	承認待	差戻	申請中
申請者へ差戻	差戻	申請者へ差戻	申請中
取止/ここで終了	承認	ここで終了（承認）	未申請
否認	否認	否認	未申請

(5) 第一最終承認後処理

処理	ワークフロー ステータス	承認ステータス	申請フラグ
承認・申請	承認	承認	未申請
引戻	-	-	-
承認者へ差戻	差戻	差戻	申請中
申請者へ差戻	差戻	申請者へ差戻	申請中
取止/ここで終了	承認	ここで終了（承認）	未申請
否認	否認	否認	未申請

9.3 処理対象データのキー情報の取得

intra-mart の標準ワークフロー機能のテーブル内に保存されたキー情報を取得することができます。EX 申請システムで登録されたデータのキー情報（申請番号、入力様式コード）が保存されています。

これにより、ワークフロー後処理から EX 申請システムで登録されたデータへアクセスすることができます。

パラメータの取得は以下のテーブルから行うことができます。

取得項目：パラメータ番号（parameter_no）、パラメータコード（parameter_cd）
対象テーブル：b_bpw_t_user_application_key
条件：プロセス定義コード、プロセス履歴コード、プロセスコード

パラメータ番号＝１　・・・　入力様式コード
パラメータ番号＝２　・・・　申請番号
パラメータ番号＝３　・・・　最終更新日時