# フローを利用

# 1. 機能の概要:

フローテンプレートを選択して申請フローを起票し、申請を行う画面です。

# 2. 目的:

承認の過程で必要な要素（承認位置, 承認者, 添付ファイル, 選択されたユーザーやグループ）を効率的に選択し, 適切な承認処理を進行できるようにします。

# 3. 画面構想（後で記載）

1.新規申請

2…..

# 4. データオブジェクト一覧 ・ビジネスロジック

## ①機能：新規申請

関連クラスとアクション：

モデルクラス：FlowRequestData

アクション：CreateFlowRequest

説明：[新規申請]ボタンをクリックして申請フロー選択画面に遷移します。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | パラメータ |  | ダミーデータ | 説明 |
|  | flow\_request\_id | 申請フローID | SKT001… |  |
|  | flow\_request\_name | 申請フロー名/概要 | 稟議フロー |  |
|  | flow\_id | フローID | F1 |  |
|  | form\_id | 帳票ID | T1 |  |
|  | form\_name | 帳票名 | 稟議書 |  |
|  | node\_id | 承認者位置 | R1 |  |
|  | user\_id | 申請フロー作成者ID | 32430… |  |
|  | user\_name | 申請フロー作成者名 | クリエイト ユーザ03 |  |
|  | user\_group\_id | ユーザーグループId | 2… |  |
|  | user\_group\_name | ユーザーグループ名 | IT人材 |  |
|  | user\_position | 役職 | 一般 |  |
|  | process\_function | 行った機能 | 申請する | 保存、申請、中止、削除・・・ |

## ビジネスロジック：新規申請

関連クラスとアクション：

モデルクラス：FlowRequestData

アクション：CreateFlowRequest

説明：[新規申請]ボタンをクリックして申請フロー選択画面に遷移します。［フロー一覧］から、起票するフローテンプレートを選択します。申請フローID・申請フロー名が変更できる場合や、申請フロー作成グループが選択できる場合は、 申請フロー情報指定ダイアログが表示されますので、各項目を入力して［申請する］ボタンをクリ ックします。自動的に次の帳票画面または申請指示画面に進みます。

**申請フローID・申請フロー名はフローテンプレート作成の設計書の　パラメータ説明の採番ルールと命名規則の通り設定できます。**

フォームから受け取ったデータを格納するオブジェクト作成

クラス:FlowRequest

アクション: CreateFlowRequest（FlowRequestData inputData）

// 入力データを整形し、必要な形式に変換

        var flowRequestData= new FlowRequestData

        {

flow\_request\_id=inputdata.flow\_request\_id

flow\_request\_name=inputdata.flow\_request\_name

flow\_id=inputdata.flow\_id

flow\_name=inputdata.flow\_name

form\_id=inputdata.form\_id

form\_name= inputdata.form\_name

user\_id=inputdata.user\_id

user\_name=inputdata.user\_name

user\_group\_id=inputdata.group\_id

user\_position=inputdata.user\_position

process\_function=inputdata.process\_function

        };

戻り値：flowRequestData

flowRequestDataをflow\_requestテーブルにデータを挿入します。

DBに接続処理

データベース操作を行っています。

挿入query

**※同時に申請フロー状態もflow\_approval\_resultテーブル挿入にします。**

## ②機能：フロー状態一覧

関連クラスとアクション：

モデルクラス：FlowApprovalResultData

アクション：FlowApprovalResult

説明：申請フローの中止、中止取消、削除や処理済の承認位置へなどのフロー状態のメンテナンスを行います。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | パラメータ |  | ダミーデータ | 説明 |
|  | flow\_request\_id | 申請フローID | SKT001… |  |
|  | node\_id | ノードID | R1 |  |
|  | approve\_type | 承認者タイプ | 申請フロー作成者 | グループ・ユーザー |
|  | require\_position | 役職 | 一般 |  |
|  | approve\_id | 承認者ID | 324301.. |  |
|  | status | 状態 | 申請する・申請待ち | 申請者が行った機能によって状態を設定します。 |
|  | comment | コメント | 申請待ち |  |
|  | status\_count | 状態をカウント | １ | 申請したフローに対して申請者が行った機能をカウント |
|  | delete\_flag | 削除フラグ | ０ | １：中止　２：削除 |

## ビジネスロジック：フロー状態一覧

クラス:FlowRequest

アクション:FlowApprovalResult（string flow\_request\_id , string user\_id）

// 入力データを整形し、必要な形式に変換

        var flowApprovalResultData= new FlowApprovalRequestData

        {

flow\_request\_id= flow\_request\_id

node\_id=”R1”

approve\_type= “申請フロー作成者“

require\_position=”一般“

approve\_id=user\_id

status=申請する・申請待ち

comment=申請待ち

status\_count=１

delete\_flag = 0

        };

戻り値：flowApprovalResultData

flowApprovalResutlDataをflow\_approval\_resultテーブルにデータを挿入します。

DBに接続処理

## ③機能：保存・申請・中止・削除

### **保存**

クラス:FlowRequest

アクション:FlowRequestSave（）

帳票に関するデータを保存します。※帳票利用設計書を参照。

### **申請**

クラス:FlowRequest

アクション:FlowRequestApply（string flow\_request\_id , string node\_id）

説明： フローを申請しました。flow\_approval\_resultテーブルのstatusを”申請“に更新します。

更新query

UPDATE flow\_approval\_result

SET status= “申請“, comment = ”SKT001…を申請しました。“

WHERE flow\_request\_id = 'SKT001…' and node\_id = ‘R1’

申請後は次の承認者を決定します。④の承認を参照

### **中止**

クラス:FlowRequest

アクション:FlowRequestStop（string flow\_request\_id , string node\_id）

説明：フロー全体を終了にします。flow\_approval\_resultテーブルのdelete\_flagを１に更新します。

更新query

UPDATE flow\_approval\_result

SET delete　= 1, status\_count = 現状のカウント+1, comment = ”SKT001…を中止しました。“

WHERE flow\_request\_id = 'SKT001…' and node\_id = ‘R1’

### **削除**

クラス:FlowRequest

アクション:FlowRequestDelete（string flow\_request\_id , string user\_id）

説明：削除フラグを更新して一覧から隠すします。flow\_approval\_resultテーブルのdelete\_flagを２に更新します。

更新query

UPDATE flow\_approval\_result

SET delete　= ２, status\_count = 現状のカウント+1、, comment = ”SKT001…を削除しました。“

WHERE flow\_request\_id = 'SKT001…' and node\_id = ‘R1’

## ④機能：承認

if(user\_leve<8){ 飛び先　＝　A1}

else\_if(user\_leve=8){ 飛び先　＝　A2}

else\_if(user\_leve=10){ 飛び先　＝　A3}

else\_if(user\_leve=12){ 飛び先　＝　A4}

else{ 飛び先＝A5}

申請者

if(最終決裁者＝責任者){ 飛び先＝A10}

else{ 飛び先　＝　A5}

else\_if(user\_leve=10){ 飛び先　＝　A3}

else\_if(user\_leve=12){ 飛び先　＝　A4}

else{ 飛び先＝A5}

if(最終決裁者＝副担当役員)

{ 飛び先＝A6}

else{ 飛び先　＝　A10}

else{ 飛び先　＝　A5}

else\_if(user\_leve=10){ 飛び先　＝　A3}

else\_if(user\_leve=12){ 飛び先　＝　A4}

else{ 飛び先＝A5}

if(最終決裁者＝責任者){ 飛び先＝A10}

else{ 飛び先　＝　A9}

else\_if(user\_leve=10){ 飛び先　＝　A3}

else\_if(user\_leve=12){ 飛び先　＝　A4}

else{ 飛び先＝A5}

if(最終決裁者＝責任者){ 飛び先＝A10}

else{ 飛び先＝A8}

if(修＝修繕){ if (金額>＝50000001){ 飛び先　＝　A11} else{飛び先　＝　A12}}

else{ if (金額>=20000001){ 飛び先　＝　A11} else{飛び先　＝　A12}}

## ビジネスロジック：承認

関連クラスとアクション：

クラス：FlowApprove

アクション：NextApproverDecision（string flow\_id, string\_node\_id）

説明：承認者決定処理

参照テーブル： flow\_request\_client, flow\_condition\_setting, flow\_request

1. flow\_request\_clientの　condition\_setting\_flagを取得します。

取得sql (flow\_id=F1, node\_id = R1)

select frc.condition\_setting\_flag from flow\_request\_client frc left join flow\_request fr on fr.flow\_id = frc.flow\_id , fr.node\_id = frc.node\_id

1. condition\_setting\_flag がtrue (R1ノードに条件があること)　の場合だけflow\_condition\_settingテーブルを参照します。

flow\_condition\_settingテーブルからnode\_id = R1に関する　条件情報を取得します。

テーブル

AI によって生成されたコンテンツは間違っている可能性があります。

取得sql

select condition\_id from flow\_condition\_setting where flow\_id = F1 and node\_id = R1。

condition\_idを基にしてflow\_logicalテーブルと紐づけます。詳細の論理和を取得します。

貰ったデータをリスト変換しまます。

if(user\_level<8){ 飛び先　＝　A1}

else\_if(user\_level=8){ 飛び先　＝　A2}

else\_if(user\_level=10){ 飛び先　＝　A3}

else\_if(user\_level=12){ 飛び先　＝　A4}

else{ 飛び先＝A5}

foreach (var condition in flowLogicals)

{

// Use the logical framework to check conditions

if (condition.LogicalItemType == "user\_level" && condition.LogicalItemId == "<")

{

if (int.TryParse(condition.LogicalValue, out int value) && userLevel < value)

{

nextNodeId = condition.NextNodeId;

break;

}

}

else if (condition.LogicalItemType == "user\_level" && condition.LogicalItemId == "=")

{

if (int.TryParse(condition.LogicalValue, out int value) && userLevel == value)

{

nextNodeId = condition.NextNodeId;

break;

}

}

else if (condition.LogicalItemType == "user\_level" && condition.LogicalItemId == "null")

{

if (string.IsNullOrEmpty(condition.LogicalValue))

{

nextNodeId = condition.NextNodeId;

break;

}

}

まずは、申請者のユーザーレベルを見ます。申請者のユーザーレベルをみて次の

1. condition\_setting\_flag がfalse は条件なしで次の処理を進みます。

条件に合っている場合は次の承認者として判断します。