Redmine チケットプロバイダ

Version 1.0.0

ユーザーズ ガイド

July 2, 2019

備考：

【改版履歴】

バージョン	日付	内容
1.0.0	2019-07-02	初版.

目次

1. はじめに

Redmine チケットプロバイダは，Redmine との通信を REST を用いて行い，Redmine のチケットの取得・投稿を行う ORiN2 CAO プロバイダです.

Redmine チケットプロバイダを利用することで，Redmine のチケットを閲覧・投稿することができます.

本ドキュメントでは，Redmine チケット プロバイダの概要と，実装されている CAO インタフェース（関数仕様）について説明しています.

2. プロバイダの概要

2.1. インストール

Redmine チケットプロバイダモジュールは，下記の DLL で構成されています．ORiN2 SDK のインストーラでインストールした場合は，インストール作業は不要です．手動でインストールする場合は，表 2-1 のように実行してください．

RegistAsm.bat および UnregistAsm.bat は ORiN2SDK をインストールしたフォルダの下の DotNet¥BAT フォルダにあります．

表 2-1 Issues プロバイダ

ファイル名	CaoProvRedmineIssues.dll
ProgID	CaoProv.Redmine.Issues
レジストリ登録	RegistAsm.bat CaoProvRedmineIssues.dll
レジストリ登録の抹消	UnregistAsm.bat CaoProvRedmineIssues.dll

2.2. 概要

Redmine チケットプロバイダを利用することで，Redmine のチケットを閲覧・投稿することができます．

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

Controller クラスを作成し，ワークスペースに追加します.

書式　AddController (<bstrCtrlName:BSTR>, <bstrProvName:BSTR>, <bstrPcName:BSTR>, [<bstrOption:BSTR>])

< bstrCtrlName > : [in] コントローラ名
< bstrProvName > : [in] プロバイダ名. 固定値 =" CaoProv.Redmine.Issues"
<bstrPcName> : [in] プロバイダの実行マシン名（未使用）
<bstrOption> : [in] オプション文字列
以下のオプションをコンマ区切りで指定できます.

URL	:	接続先の URL を指定します.（必須） 例） URL=http://loalhost/redmine/
UserID	:	Redmine にログインするユーザ ID を指定します. 指定する場合は Password とセットで指定する必要があります.
Password	:	Redmine にログインするユーザのパスワードを指定します. 指定する場合は UserID とセットで指定する必要があります. 例） UserID=Test,Password=Test
AppKey	:	Redmine の REST API にアクセスするためのキーを指定します. AppKey を指定した場合，UserID および Password は無視されます.

2.3.2. CaoController::AddExtension() コマンド

Extension クラスを作成し, Controller クラスに追加します.
Extension クラスは Redmine のチケットに対応します.

書式　AddExtension(<VariableName>,<Option>)

<VariableName> : [in] 変数名(VT_BSTR)
<Option> : [in] オプション文字列
以下のいずれかのオプションを指定します.

ProjectID	:	投稿先のプロジェクトを指定して新規作成モードで Extension を作成します.
IssueID	:	取得するチケットの ID を指定して編集モードで Extension を作成します.

戻り値 : [out] 追加した Extension

使用例

```
CaoExtension ext = m_caoController.AddExtension("Ext",
“ProjectID=207”)
```

2.3.3. CaoExtension::AddVariable() コマンド

Variable クラスを作成し，Extension クラスに追加します.
Variable クラスは Redmine のチケットの各項目に対応します.

書式　AddVariable(<VariableName>,<Option>)

<VariableName> : [in] 変数名(VT_BSTR)
<Option> : [in] オプション文字列
システム変数以外の場合，以下のオプションを指定します.
CustomFieldID : 変数が表わすカスタムフィールドの ID を指定します.
ユーザ定義変数の場合は必須です.
戻り値 : [out] 追加した Variable

使用例

```
CaoVariable ext = m_caoExtension.AddVariable("@ID", null)
```

2.3.4. CaoController::GetVariableNames()コマンド

使用できるシステム変数の一覧を取得します.

書式　GetVariableNames(<Option>)

<Option> : [in] オプション(未使用)
戻り値 : [out] 使用できるシステム変数の一覧(VT_ARRAY | VT_BSTR)

使用例

```
string[] varNames = _controller.GetVariableNames(null)
```

2.4. Variable クラス

2.4.1. CaoVariable::get_Value()

変数の値を取得します.

書式　GetValue()

戻り値　：　[out] 変数の値(VT_VARIANT)

使用例

```
Object variableValue = val.Value
```

2.4.2. CaoVariable::put_Value()

変数の値を設定します.

書式　PutValue(<VariableValue>)

<VariableValue>　：　[in] 変数の値

使用例

```
val.Value = “1”
```

3.コマンドリファレンス

3.1. Controller クラス

表 3-1 CaoController::Execute コマンド一覧

コマンド	機能	ページ
GetProjects	ログインしたユーザがアクセスできるプロジェクトの一覧を取得します.	10
GetTrackers	トラッカの一覧を取得します.	11
GetStatuses	ステータスの一覧を取得します.	11
GetCategories	指定したプロジェクトに設定されているカテゴリの一覧を取得します.	12

3.1.1. CaoController::Execute(“GetProjects”) コマンド

ログインしたユーザがアクセスできるプロジェクトの一覧を取得します.

書式　GetProject()

戻り値　：　[out] プロジェクトの情報(VT_VARIANT | VT_ARRAY)
配列の各要素は VT_BSTR | VT_ARRAY であり, 第 1 要素がプロジェクト ID, 第 2 要素がプロジェクト名となっている.

使用例

```
m_caoController.Execute("GetProjects", null)
```

3.1.2. CaoController::Execute(“GetTrackers”) コマンド

トラッカの一覧を取得します.

書式　GetTrackers ()

戻り値　：[out] トラッカの情報(VT_VARIANT | VT_ARRAY)
配列の各要素は VT_BSTR | VT_ARRAY であり, 第 1 要素がトラッカ ID, 第 2 要素がトラッカ名となっている.

使用例

```
m_caoController.Execute("GetTrackers", null)
```

3.1.3. CaoController::Execute(“GetStauses”) コマンド

ステータスの一覧を取得します.

書式　GetStatuses()

戻り値　：[out] ステータスの情報(VT_VARIANT | VT_ARRAY)
配列の各要素は VT_BSTR | VT_ARRAY であり, 第 1 要素がステータス ID, 第 2 要素がステータス名となっている.

使用例

```
m_caoController.Execute("GetStatuses", null)
```

3.1.1. CaoController::Execute(“GetCategories”) コマンド

指定したプロジェクトに設定されているカテゴリの一覧を取得します.

書式　GetCategories(<projectId>)

<projectId>　:　[in] カテゴリを取得するプロジェクトの ID
戻り値　:　[out] カテゴリの情報(VT_VARIANT | VT_ARRAY)
配列の各要素は VT_BSTR | VT_ARRAY であり, 第 1 要素がカテゴリ ID, 第 2 要素がカテゴリ名となっている.

使用例

```
m_caoController.Execute("GetCategories", 207)
```

3.2. Extension クラス

表 3-2 CaoExtension::Execute コマンド一覧

コマンド	機能	ページ
Apply	Variable で指定した値を使用して Redmine に投稿を行います.	13
CreateNew	投稿するプロジェクトを指定して Extension を新規作成モードで初期化します.	14
Reload	編集するチケットを指定して Extension を編集モードで初期化します.	14
GetCustomFields	編集モードのとき，カスタムフィールドの一覧を取得します.	15
RegistAttachmentFile	添付ファイルを登録します.	16

3.2.1. CaoExtension::Execute(“Apply”)コマンド

Variable で指定した値を使用して Redmine に投稿を行います.
新規作成モードで投稿した場合，投稿したチケットの情報で変数の内容が更新され，自動的に編集モードに切り替わります.

書式　Apply(<Comment>)

<Comment>　:　[in] 投稿するコメント. 新規作成モードの場合は無視されます
戻り値　:　[out] なし

使用例

```
_extension.Execute(“Apply”,”コメント”)
```

3.2.2. CaoExtension::Execute(“CreateNew”)コマンド

投稿するプロジェクトを指定して Extension を新規作成モードで初期化し，AddExtension で ProjectID を指定したときと同様の状態に戻します.
追加した変数は削除されず，値のみが初期値に戻ります.

書式　CreateNew(<ProjectID>)

<ProjectID>　：　[in] 投稿を行うプロジェクトの ID
戻り値　：　[out] なし

使用例

```
_extension.Execute(“CreateNew”, 207)
```

3.2.3. CaoExtension::Execute(“Reload”)コマンド

編集するチケットを指定して Extension を編集モードで初期化し，AddExtension で IssueID を指定したときと同様の状態に戻します.
追加した変数は削除されず，値のみが初期値に戻ります.

書式　Reload(<IssueID>)

<IssueID>　：　[in] 編集を行うチケットの ID
戻り値　：　[out] なし

使用例

```
_extension.Execute(“Reload”, 1000)
```

3.2.4. CaoExtension::Execute(“GetCustomFields”)コマンド

カスタムフィールドの一覧を取得します.
編集モード（チケットの情報がロードされた状態）の時のみ値が取得できます.

書式　GetCustomFields(<Option>)

<Option>　：　[in] オプション（未使用）
戻り値　：　[out] (VT_VARIANT | VT_ARRAY)でカスタムフィールドの情報を戻します.
各要素は 3 要素の配列（VT_VARIANT | VT_ARRAY）になっており，それぞれ以下の意味を持ちます.
第 1 要素：カスタムフィールドの ID（VT_I4）
第 2 要素：カスタムフィールド名（VT_BSTR）
第 3 要素：カスタムフィールドの値（VT_BSTR）

使用例

```
object customFields = _extension.Execute(“GetCustomFields”, null)
```

3.2.5. CaoExtension:: Execute(“RegistAttachmentFile”)コマンド

チケットに添付するファイルを登録します.
登録したファイルは Apply を実行したときにアップロードされ, チケットに添付されます.

書式　RegistAttachmentFile(<Option>)

<Option> : [in] (VT_VARIANT | VT_ARRAY)で以下のデータを渡してください.

第 1 要素：ファイルイメージ（VT_UI1 | VT_ARRAY）
第 2 要素：ファイル名（VT_BSTR）
第 3 要素：ファイルの説明（VT_BSTR, 省略可）
第 4 要素：コンテントタイプ（VT_BSTR, 省略可）

コンテントタイプを省略した場合は指定したファイル名の拡張子に従って自動的にコンテントタイプを決定します.

戻り値 : [out] なし

使用例

```
_extension.Execute(“RegistAttachmentFile”, new object[] {data, name})
```

4. 変数について

4.1. 概要

Redmine チケットプロバイダの変数は Redmine に標準で準備されている項目に対応するシステム変数と，カスタムフィールドに対応するユーザ定義変数の 2 種類の変数があります.
プロバイダは変数の put_Value で設定した値を記録しており，Apply コマンド実行時には記録された値のみが Redmine に送信されます.
Apply コマンド実行後は記録した値は削除され，次に Apply コマンドを実行した際には前回分の値は送信されません.

4.2. システム変数

Redmine チケットプロバイダで使用できるシステム変数の一覧を以下に示します.
システム変数には書込みができる投稿する値を設定する変数と書込みができない参照専用の変数があります.
書込み可の項目を省略した場合,新規作成モードの時はデフォルト値が設定されている場合はデフォルト値が使用され，編集モードの時は値が変更されません.
また，ID と名称が組になっている項目の名称は編集モードで，かつ，ID に対して put_Value で値が設定されていない時のみ取得でき，それ以外の時は値が空になります.

表 4-1 システム変数一覧

システム変数	意味	書込み	備考
@ID	チケット ID	不可	
@Subject	タイトル	可	必須
@ProjectID	プロジェクト ID	可	
@ProjectName	プロジェクト名称	不可	
@TrackerID	トラッカ ID	可	編集時に VT_NULL を設定したときは Apply 実行時に値が変化しない
@TrackerName	トラッカ名称	不可	
@StatusID	ステータス ID	可	編集時に VT_NULL を設定したときは Apply 実行時に値が変化しない
@StatusName	ステータス名称	不可	
@PriorityID	優先度 ID	可	編集時に存在しない優先度 ID を設定したときは Apply 時にエラーになる

@PriorityName	優先度名称	不可	
@AuthorID	投稿者 ID	不可	
@AuthorName	投稿者名称	不可	
@AssignedToID	担当者 ID	可	編集時に VT_NULL を設定したときは Apply 実行時に値が変化しない
@AssignedToName	担当者名称	不可	
@CategoryID	カテゴリ ID	可	
@CategoryName	カテゴリ名称	不可	
@ParentIssueID	親チケット ID	可	編集時に VT_NULL を設定したときは Apply 実行時に削除される
@Description	説明	可	
@StartDate	開始日	可	編集時に VT_NULL を設定したときは Apply 実行時に削除される
@DueDate	期日	可	編集時に VT_NULL を設定したときは Apply 実行時に削除される
@DoneRatio	進捗率	可	パーセント単位の整数 編集時に VT_NULL を設定したときは Apply 実行時に 0 に設定される
@EstimatedHours	予定工数	可	実数 編集時に VT_NULL を設定したときは Apply 実行時に削除される
@CreatedOn	作成日時	不可	
@UpdatedOn	更新日時	不可	
@IsNewIssue	新規作成モードフラグ	不可	新規作成モードの時は True, 編集モードの時は False

4.3. ユーザ定義変数

ユーザ定義変数はチケットに設定されているカスタムフィールドに対応します.
ユーザ定義変数はすべて書込みが可能です.
ユーザ定義変数を追加する際はオプション文字列に”CustomFieldID={カスタムフィールド ID}”を必ず指定してください.
カスタムフィールド ID が重複するユーザ定義変数は追加することができません.

5. サンプルプログラム

Redmine チケットプロバイダと C# を使った簡単なサンプルプログラムを紹介します.

サンプルプログラム

```
try
{
    _ctrl = _engine.Workspaces[0].AddController(
        "Redmine",
        "CaoProv.Redmine.Issues",
        null,
        string.Format( "Url={0},
                        UserID={1},Password={2}",
                        txtUrl.Text,
                        txtUserId.Text,
                        txtPassword.Text));

    _extension = _ctrl.AddExtension(    "Issue",
                                        string.Format("ProjectId={0}",
                                        txtProject.Text));
    _extension.Variables.Add("@ID", null);

    foreach (Control ctrl in pnlEdit.Controls)
    {
        RedmineEditControl editCtrl = ctrl as RedmineEditControl;
        if (editCtrl == null)
        {
            continue;
        }

        _extension.Variables.Add(editCtrl.VariableName, editCtrl.Option);
    }

    CreateNew();

    txtProject.Enabled = false;
```

```
    txtPassword.Enabled = false;
    txtUserId.Enabled = false;
    txtUrl.Enabled = false;
    btnConnect.Enabled = false;
    pnlEdit.Visible = true;
}
catch (Exception ex)
{
    if (_ctrl != null)
    {
        _engine.Workspaces[0].Controllers.Remove(_ctrl.Name);
    }
    MessageBox.Show(    this,
                        ex.ToString(),
                        this.Text,
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
}
```