目次

改訂情報

変更年月日	変更内容
2018-04-01	初版

はじめに

本書の目的

本書は、IM-Repository for Accel Platform（以下 IM-Repository）におけるそれぞれの機能を拡張する仕組の詳細および、基本的な使用方法も併せて説明します。

説明範囲は以下のとおりです。

- 辞書項目APIの使用方法について
- 列挙型APIの使用方法について

対象読者

本書では以下のユーザを対象としています。

- IM-Repositoryを利用して処理を実装したい
- IM-Repositoryと連携した機能を実装したい

サンプルコードについて

本書に掲載されているサンプルコードは可読性を重視しており、性能面や保守性といった観点において必ずしも適切な実装ではありません。
開発においてサンプルコードを参考にされる場合には、上記について十分に注意してください。

本書の構成

- *辞書項目API*

  辞書項目APIについて説明します。

- *列挙型API*

  列挙型APIについて説明します。

辞書項目API

辞書項目APIとは。

IM-Repository上で管理される辞書項目は、バージョン管理および、項目内に用途や制約といった様々な定義が存在します。
本項ではこの辞書項目の情報を取得するAPIの利用方法を説明します。

最新バージョン

項目
- 最新バージョンの辞書を取得する

最新バージョンの辞書を取得する

REST-API

メソッド	GET
URI	%ベースURL%/api/repository/dictionary/current

JavaEE開発モデル

```
// 最新バージョンの辞書を取得
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// 辞書のバージョンを確認する
dictionary.getVersion();

// 辞書項目を検索する
Criteria criteria = new Criteria();
criteria.setName("%表示名%");
dictionary.search(criteria);
```

辞書項目

項目
- 辞書項目を取得する
- 辞書項目のエイリアスを取得する

辞書項目を取得する

JavaEE開発モデル

```
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// 辞書項目を取得する
Item item = dictionary.find(ItemId.of("%辞書項目のID%"));
if (item != null) {
   // 辞書項目名を取得する
   item.getName();
}
```

辞書項目のエイリアスを取得する

JavaEE開発モデル

```
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// 指定された辞書項目のエイリアスを取得する
List<Alias> alias = dictionary.findAlias(ItemId.of("%辞書項目のID%"));
```

エイリアス

項目

- エイリアスを取得する

エイリアスを取得する

JavaEE開発モデル

```
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// エイリアスを取得する
Alias alias = dictionary.find(AliasId.of("%エイリアスのID%"));
if (alias != null) {
   // エイリアス名を取得する
   alias.getName();
}
```

用途と制約

辞書項目やエイリアスそのものは、メタデータとして存在するだけで利用価値はありません。 これらに用途を指定することで、単純なメタデータではなく様々なアプリケーションから 利用することができる存在に成ります。

現在用意している用途は

- データ
- 制約

の2種類です。

データ用途は、対象のメタデータの形式、例えばデータベースのスキーマを生成するための情報などを管理することができます。

制約用途は、対象のメタデータのバリデーションを行うための情報を管理することができます。

用途の参照

REST-API

メソッド	GET
URI	%ベースURL%/api/repository/dictionary/usages

JavaEE開発モデル

```
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// 辞書項目、エイリアスの順に用途を取得する
Item item = dictionary.find(ItemId.of("%辞書項目のID%");
Alias alias = dictionary.find(AliasId.of("%エイリアスのID%");

Usages usages = null;
if (item != null) {
   usages = item.getUsages();
} else if (alias != null) {
   usages = alias.getUsages();
}
```

制約の確認方法

REST-API

メソッド	GET
URI	%ベースURL%/api/repository/dictionary/usages

JavaEE開発モデル

```
Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

// 辞書項目、エイリアスの順に用途を取得する
Item item = dictionary.find(ItemId.of("%辞書項目のID%");
Alias alias = dictionary.find(AliasId.of("%エイリアスのID%");

Usages usages = null;
if (item != null) {
    usages = item.getUsages();
} else if (alias != null) {
    usages = alias.getUsages();
}

// 制約の一覧を取得する
RestrictionUsage restrictionUsage = usages.get("制約用途のID文字列");
List<Restriction> list = restrictionUsage.getRestrictions();
```

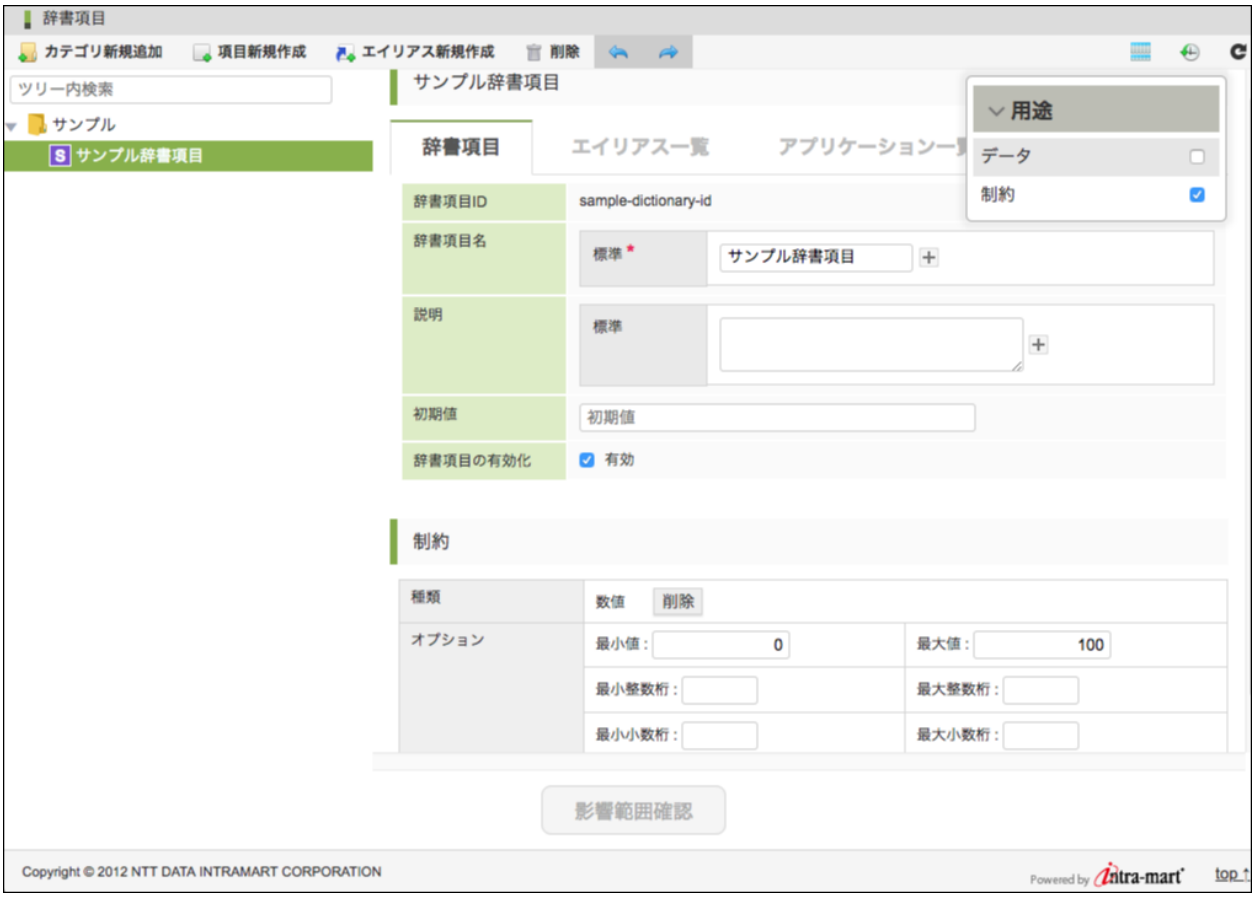
制約用途に指定した制約に適合するかどうかを確認する

辞書項目の設定

あらかじめ、次のような設定を行います。これら以外の項目は任意に指定してください。

辞書項目ID	sample-dictionary-item	
用途	制約にチェックを付ける	
数値制約	制約を追加で追加しておく	
	最小値	0
	最大値	100
	最小整数桁	空
	最大整数桁	空
	最小小数桁	空

最大小数桁	空



JavaEE開発モデル

次のような実装を行うことで、上記の制約に基づいたバリデーションを行うことができます。

```
/**
 * 辞書項目IDと値の組み合わせで以下のような結果が返ります。
 */
public void someMethod() throws DictionaryServiceException {
    this.validate("sample-dictionary-item", 0);   // true
    this.validate("sample-dictionary-item", 100); // true
    this.validate("sample-dictionary-item", -1);  // false
    this.validate("sample-dictionary-item", 101); // false
}

/**
 * 指定した辞書項目の制約を用いて、対象の値が適合するかどうかを確認します。
 *
 * @param itemId 辞書項目ID
 * @param target 対象の値。ここでは数値が制約に合致するかどうかを確認することとします。
 * @return 適合する場合は true を、適合しない場合は false を返します。
 * @throws DictionaryServiceException 辞書サービスでエラーが発生した場合にスローされます。
 */
public boolean validate(final String itemId, final Double target) throws
DictionaryServiceException {
    final Dictionary dictionary =
RepositoryServiceProvider.getInstance().getDictionaryService().getCurrent();

    final Item item = dictionary.find(ItemId.of(itemId));
    if (item != null) {
        final Usages usages = item.getUsages();
        final RestrictionUsage restrictionUsage = usages.get(UsageId.of(RestrictionUsage.ID));
        for (final Restriction restriction : restrictionUsage.getRestrictions()) {
            // validate の第一引数はバリデーションエラーのメッセージに含める名称です。
            // 現在提供している制約のエラーメッセージでは使用されていません。
            final ValidateResult validationResult = restriction.validate("名前", target);
            if (validationResult.getInvalidParams().size() > 0) {
                return false;
            }
        }
    }
    return true;
}
```

バリデーション

項目
- バリデーションを利用する

バリデーションを利用する

制約用途に指定した制約に適合するかどうかを確認する の設定を行っている場合、次のような実装を行うことでリクエストの パラメータに対してバリデーションを行うことができます。

JavaEE開発モデル

JSR 303 Bean Validationに準拠したカスタムバリデーションとして、辞書項目の制約に基づいたものを提供します。
@Dictionary(id=”xxx”) アノテーションを付与することにより、辞書項目に定義されている制約を適用することが可能です。
この例では SampleForm の foo に対して 0 から 100 までの範囲内であるという制約が適用されます。

```
package sample;

import
jp.co.intra_mart.foundation.repository.metadata.dictionary.validator.Dictionary;

public class SampleForm {
   // 辞書項目IDをアノテーションに指定する
   // id = 辞書項目IDまたはエイリアスID、name = 項目名
   @Dictionary(id = "sample-dictionary-item", name = "foo")
   private Long foo;
}
```

スクリプト開発モデル

Jssp Validator のカスタムバリデーションとして、辞書項目の制約に基づいたものを提供します。
バリデーションルールのキーに dictionary を、値に辞書項目IDを指定することにより、辞書項目に定義されている制約を適用することが可能です。
この例ではリクエストパラメータの foo に対して 0 から 100 までの範囲内であるという制約が適用されます。

validation.js

```
var init = {
  'foo': { // リクエストパラメータ名
      caption: "CAP.Z.EXAMPLE.NAME", // エラーメッセージのキャプション
      dictionary: 'sample-dictionary-item'
   }
}
```

sample.js

```
let foo;
/**
 * @validate validation#init
 * @onerror handleError
 */
function init(request) {
  foo = request.foo;
}

function handleError(request, validationErrors) {
  var param = {
    errors: validationErrors.getMessages()
  };
  forward("error", param);
}
```

コラム

現在クライアント側のバリデーションを行うことはできません。

列挙型API

列挙型APIとは。

label-value形式の列挙データを統合的に管理する機能を提供します。
また、辞書データと連動させ、辞書項目の値としてこの列挙データからのみ選べるといった連携を行います。
本項ではこの列挙型の情報を取得するAPIの利用方法を説明します。

列挙型

列挙型を取得する

REST-API

メソッド	GET
URI	%ベースURL%/api/repository/metadata/enumeration

JavaEE開発モデル

```
// 列挙型を取得する
EnumerationService enumerationService =
RepositoryServiceProvider.getInstance().getEnumerationService();
Enumerations enumerations = enumerationService.getEnumerations();
```

列挙型検索

列挙型を検索する

スクリプト開発モデル

列挙型検索画面を呼び出す方法は2通りあります。

・列挙を全て呼び出したい場合（※1）

sample.js

```
function openSearchEnumerationDialog() {
 // 列挙型検索画面を呼び出します
 window.open('%ベースURL%/repository/search/metadata/enumeration');
}
```

・列挙が有効化のデータのみを呼び出したい場合（※2）

sample.js

```
function openSearchEnumerationDialog() {
 // 列挙型検索画面を呼び出します
 window.open('%ベースURL%/repository/search/metadata/enumeration?enabled=true');
}
```

コラム

下記のようにコールバック関数を利用することもできます。

（※1）の場合(‘%ベースURL%/repository/search/metadata/enumeration#%コールバック関数%’)

（※2）の場合(‘%ベースURL%/repository/search/metadata/enumeration?enabled=true#%コールバック関数%’)

コールバック関数を指定しない場合、デフォルトのコールバック関数は’onEnumerationSelected’です。

結果を返却する

スクリプト開発モデル

sample.js

```
window.onEnumerationSelected = function (selected) {
 if (selected) {
  // 列挙IDを取得します
  var enumerationId = selected.identify;
  // 列挙名を取得します
  var enumerationName = selected.name.default;
 }
}
```

コラム

・コールバック関数を指定する場合

onEnumerationSelectedを指定したコールバック関数に変更してください。

引数のオブジェクトの説明をします。

・selected

取得できる情報は、以下です。

プロパティ	説明	型
identify	型を含まない列挙ID	string
parentId	親のID	string
name（※1）	列挙名	object
description（※1）	説明	object
enumerationItems（※2）	列挙項目	array
enabled	列挙の有効化を示すフラグ	boolean
label	ツリーの表示名	string

（※1）name、descriptionには多言語情報が格納されており、構成は以下です。

プロパティ	説明	型
default	標準	string
ja	日本語	string
en	英語	string
zh_CN	中国語（簡体字）	string

（※2）enumerationItemsには以下の要素が格納されています。

プロパティ	説明	型
label（※3）	列挙項目表示名	object
value	値	string

（※3）labelも多言語情報が格納されています。構成はname、descriptionと同様です。

コラム

それぞれのプロパティを取得する場合のサンプルは以下の通りです。

```
//列挙名（日本語）を取得する場合
var enumerationName = selected.name.ja;

// 説明（英語）を取得する場合
var description = selected.description.en;

//列挙項目の1件目の列挙項目表示名（標準）を取得する場合
var enumerationName = selected.enumerationItems[0].label.default;
```