PROMCODE

次世代プロジェクト管理データ交換アーキテクチャ協議会

サービス開発ガイド

第 1 版

2013 年 10 月 22 日

南山大学
日本アイ・ビー・エム株式会社
富士通株式会社
日本電気株式会社
株式会社 NTT データ
株式会社日立製作所
株式会社野村総合研究所

目次

1 PROMCODE モデルの実行環境の概要

PROMCODE の実行環境は，PROMCODE リソースモデルを Web アプリケーションとして提供する PROMCODE プロバイダと，PROMCODE プロバイダを HTTP [1] プロトコルでアクセスする PROMCODE コンシューマから構成される．

PROMCODE プロバイダの例として，Excel®用に PROMCODE で開発されたアダプタ（以下，Excel 用アダプタと呼ぶ）がある．Excel 用アダプタは，OSLC（Open Services for Lifecycle Collaboration）[2] のリファレンス実装である Eclipse Lyo [3] のサンプルをベースとした Web アプリケーションである．また PROMCODE コンシューマの例として，ブラウザクライアント，および，Eclipse Lyo のサンプルをベースとした Excel 用に PROMCODE で開発されたクライアント(以下, Excel 用クライアント)がある．以下の図は典型的な環境の全体図である（図１）．

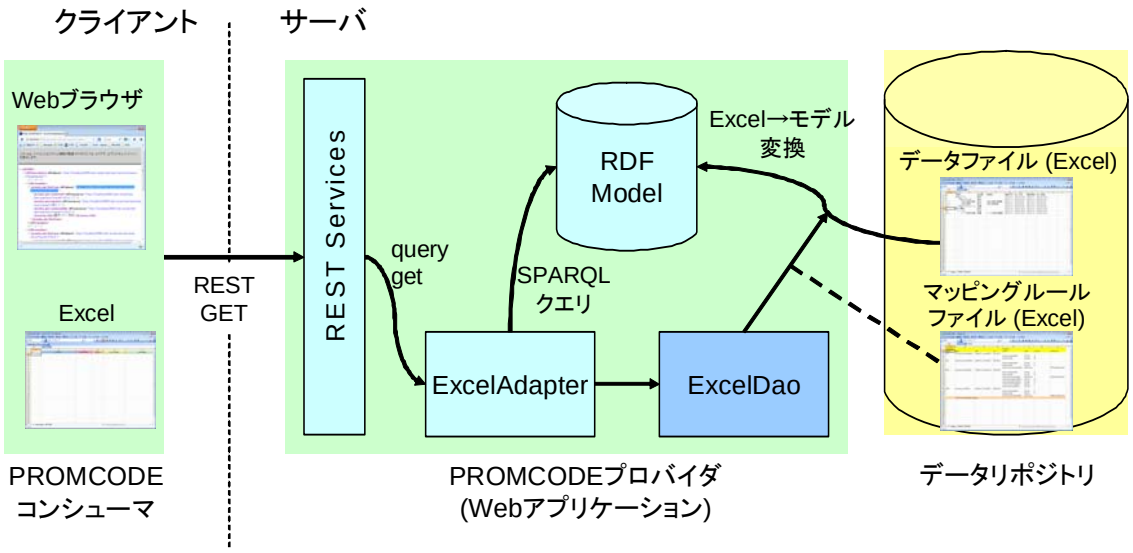

図 1，典型的な PROMCODE モデルの実行環境

Excel 用クライアントは，HTTP（REST [4]）アクセスの処理と，PROMCODE プロバイダから返される PROMCODE リソースモデルの処理を行う, VB スクリプトで記述されたクライアントアプリケーションである．

また，Excel 用アダプタは，Excel ファイルに記述されたデータをマッピングルールにしたがって読み込み，サーバ内部に RDF [5]データベースを作成し，クライアントの REST リクエストに対して RDF/XML 形式でリソースを返す Java で記述された Web アプリケーションである．Excel 用アダプタは図１にあるように，以下のようなコンポーネントで構成されている．

- REST Services
  JAX-RS [6] によるいくつかのサービスクラス．REST リクエストにしたがって関連付けられたクラスのメソッドが呼び出され，RDF/XML として結果をクライアントに返す．
- ExcelAdapter
  Excel 用アダプタのメインとなるクラスでデータリポジトリと，PROMCODE リソースを保持する RDF モデルを管理する．REST Services からのクエリーにしたがって，必要なら RDF モデルをロードし，クエリー結果を取得する．
- ExcelDao
  データファイルを読み込み，マッピングルールに基づいて PROMCODE リソースを抽出し，RDF

モデルを生成するクラス．なお，マッピングルールの Excel ファイルと，具象クラスの ExcelDaoImpl は，独自のデータファイルに対応する際に主に修正，拡張の対象となる．

- RDF Model
  抽出された PROMCODE リソースを保持する RDF データベース．

これらのコンポーネントを構成するクラスは以下の関係図に示される（図２）．

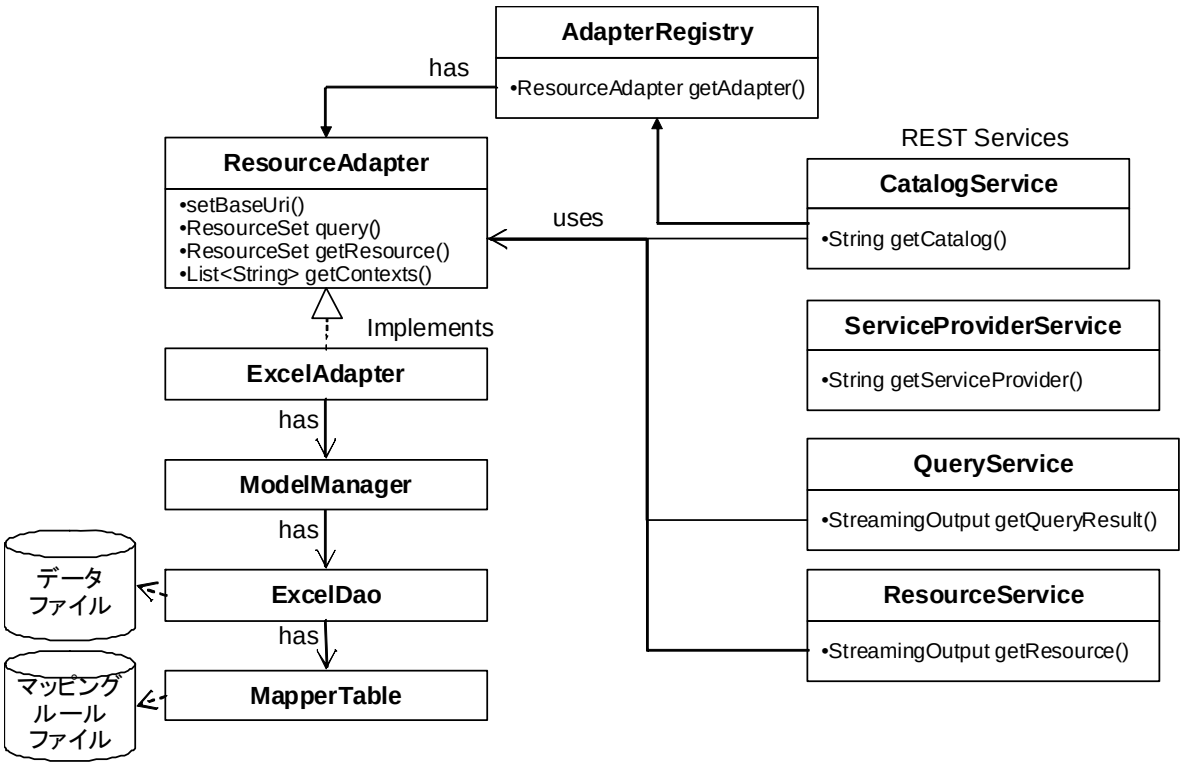

図 2，Excel 用アダプタのクラス図

2 PROMCODE モデルの開発環境のセットアップ

2.1 Excel 用アダプタのセットアップ

PROMCODE モデルの Excel 用アダプタは，Eclipse Lyo のサンプルをベースとして拡張されている．Eclipse Lyo のサンプルは，OSLC ChangeManagement のリソース ChangeRequest を，Excel で記述されたデータファイルから，マッピングルールにしたがって抽出し，OSLC コンシューマに提供するアプリケーションである．
ここでまず Eclipse Lyo の Excel 用アダプタの開発環境のセットアップ手順について説明する．なお，このセットアップ手順は，reference implementations (RIOs)のセットアップ手順とほぼ同様であるので，このセットアップ手順とともに以下のリンクに記載の RIOs のセットアップ手順をあわせて参照されたい．

- Building and Running Lyo reference implementations (RIOs) in Eclipse
  http://wiki.eclipse.org/Lyo/BuildRIO

2.1.1 開発環境の準備

開発マシンを用意し，その上に，以下の Eclipse 環境を準備する．

- JDK (1.6 以上)
- Eclipse (3.6 以上)
- 最新の EGit および m2e プラグイン

2.1.2 プロジェクトの取得

EGit の Git Repositories ビューを開き，以下の git リポジトリの master ブランチをクローンし，ブランチのプロジェクトをインポートする．

- リポジトリ：git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.rio.git
  プロジェクト：org.eclipse.lyo.rio.core
- リポジトリ：git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.core.git
  プロジェクト：org.eclipse.lyo.oslc4j.core
  org.eclipse.lyo.core.query
- リポジトリ：git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.server.git
  プロジェクト：org.eclipse.lyo.samples.excel

2.1.3 ビルド

取得したすべてのプロジェクトを選択し，コンテキストメニューから Maven > Update Project Configuration を実行する．次に，org.eclipse.lyo.oslc4j.core，org.eclipse.lyo.core.query，org.eclipse.lyo.rio.core，org.eclise.lyo.samples.excel の順に，それぞれのプロジェクトにある pom.xml を選択し，コンテキストメニューから Run As > Maven Install を実行する．

2.1.4 実行とテスト

メニューから Run > Run Configuration を選び，Maven Build をダブルクリックする．以下のフィールドを設定し，Run ボタンを押して Excel 用アダプタを起動する．

- Base Directory → ${workspace_loc:/org.eclipse.lyo.samples.excel}
- Goals → jetty:run-exploded

ブラウザを使用して，以下の URL にアクセスしてページが表示されることを確認する．

- http://localhost:8080/oslc-excel

2.2 Excel 用クライアントの実行

Eclise Lyo の Excel 用クライアントは OSLC のクライアントアプリケーションをマクロとして実装した Excel ファイルであり，Excel 用アダプタのプロジェクト org.eclipse.lyo.samples.excel に，フォルダ client の下の Client of RIO for Excel.xls として含まれている．
Client of RIO for Excel.xls をダブルクリックし Excel 用クライアントを起動する．なお，クライアントの起動時にマクロを有効にするか否かを求められるので，マクロを有効にする，を選択する．
次に Excel の Settings シートを開き，サーバ接続情報とプロパティ取得，表示情報の設定を確認する．設定が必要な情報は以下の通りである．

- Requirement Server URL:
  クライアントがアクセスするサーバの URL を指定する．デフォルトで以下の Excel 用アダプタのカタログサービスの URL が指定されている．
  http://localhost:8080/oslc-excel/rest/services/catalog/projectA
- Select

Excel 用クライアントで表示するプロパティを列挙する．デフォルトで ChangeRequest リソースのプロパティがいくつか列挙されている．

- Columns
  Excel 用クライアントに表示するプロパティのカラム位置を指定する．

設定を確認したら WorkItems シートを開き，ツールバーに表示された Download (OSLC)ボタンを押して，ChangeRequest リソースがいくつか取得，表示されることを確認する．

2.3 PROMCODE リソースモデルへの対応

PROMCODE リソースモデルへ対応するために，Excel 用アダプタ，および Excel 用クライアントを修正，変更する必要がある．

2.3.1 Excel 用アダプタの修正

Eclipse Lyo の Excel 用アダプタは OSLC ChangeManagement 用に記述されているので，これを PROMCODE 用に書き換える必要がある．

- config.xml の変更
  org.eclipse.lyo.samples.excel の下にあるコンフィギュレーションファイル config.xml の記述を以下のように変更する．

「namespace1」を以下の記述で置き換える．

```
<entry key="namespace1">promcode_pm,http://promcode.org/ns/pm#</entry>
```

「backlink1」を以下の記述で置き換える．

```
<entry key="backlink1">http://promcode.org/ns/pm#composedBy,http://promcode.org/ns/pm#consistsOf</entry>
<entry key="backlink2">http://promcode.org/ns/pm#consistsOf,http://promcode.org/ns/pm#composedBy</entry>
<entry key="backlink3">http://promcode.org/ns/pm#producedBy,http://promcode.org/ns/pm#produces</entry>
<entry key="backlink4">http://promcode.org/ns/pm#produces,http://promcode.org/ns/pm#producedBy</entry>
<entry key="backlink5">http://promcode.org/ns/pm#requiredBy,http://promcode.org/ns/pm#requires</entry>
<entry key="backlink6">http://promcode.org/ns/pm#requires,http://promcode.org/ns/pm#requiredBy</entry>
```

「serviceDomain」を以下の記述で置き換える．

```
<entry key="serviceDomain">http://promcode.org/ns/pm#</entry>
```

「queryCapability1」を以下の記述で置き換える．

```
<entry key="queryCapability1">scopeitem,http://promcode.org/ns/pm#ScopeItem</entry>
<entry key="queryCapability2">workitem,http://promcode.org/ns/pm#WorkItem</entry>
<entry key="queryCapability3">artifact,http://promcode.org/ns/pm#Artifact</entry>
```

- サンプルデータの作成
  後述のサンプルの Excel データ，および，サンプルのマッピングルールからそれぞれ Excel ファイルを作成する．データファイルの名前は data.xls とし，マッピングルールファイルの名前は mapper.xls とする．これらのファイルで Excel 用アダプタプロジェクトのデータフォルダ repository¥projectA の中身を置き換える．

以上の修正，変更を加えた Excel 用アダプタを，2.1 章で説明した方法で起動する．Excel 用アダプタが起動したらブラウザで以下のクエリー用の URL にアクセスして，PROMCODE リソースが取得，表示されることを確認する．

- http://localhost:8080/oslc-excel/rest/services/projectA/query/scopeitem
- http://localhost:8080/oslc-excel/rest/services/projectA/query/workitem
- http://localhost:8080/oslc-excel/rest/services/projectA/query/artifact

2.3.2 サンプルデータ

サンプルの Excel データは以下の通り．

表 2，サンプル Excel データ

ID	作業名	担当者	成果物	開始予定	終了予定	開始実績	終了実績
1	予約システム	田中		2012/1/1	2013/12/31	2012/1/1	2013/12/31
1.1	予約システム UI	田中		2012/1/1	2013/12/31	2012/1/1	2013/12/31
1.1.1	予約 UI 要件定義	鈴木	予約 UI 要件定義書	2012/1/1	2013/3/31	2012/1/1	2013/3/31
1.1.2	予約 UI 設計	鈴木	予約 UI 設計書	2012/4/1	2013/6/30	2012/4/1	2013/6/30
1.2	予約システムエンジン	佐藤		2012/1/1	2013/6/30	2012/1/1	2013/6/30
2	テスト環境	田中		2012/7/1	2013/10/31	2012/7/1	2013/10/31
2.1	インフラ	田中		2012/7/1	2013/10/31	2012/7/1	2013/10/31
2.1.1	インフラ要件定義	佐藤	インフラ要件定義書	2012/7/1	2013/10/31	2012/7/1	2013/10/31

サンプルのマッピングルールは以下の通り．

表 2，サンプルマッピングルール

Resource				Property			
名	型	行	URI	名	型	カラム	参照
SI1	promcode_pm:ScopeItem	Sheet1,1,*,exist(B)	SI1_%s,A				
				dcterms:identifier	String	SI1_%s,A	
				dcterms:title	String	B	
SI2	promcode_pm:ScopeItem	Sheet1,2,*,exist(C)	SI2_%s,A				
				dcterms:identifier	String	SI2_%s,A	
				dcterms:title	String	C	
				promcode_pm:composedBy	Resource		SI1[mostRecent]
WI3	promcode_pm:WorkItem	Sheet1,2,*,exist(D)	WI3_%s,A				
				dcterms:identifier	String	WI3_%s,A	
				dcterms:title	String	D	
				promcode_pm:requiredBy	Resource		SI2[mostRecent]
				promcode_pm:plannedStart	DateTime	G	
				promcode_pm:plannedEnd	DateTime	H	
				promcode_pm:actualStart	DateTim	I	

				promcode_pm:actualEnd	e DateTime	J	
AR3	promcode_pm:Artifact	Sheet1,2,*,exist(F)	AR3_%s, A				
				dcterms:identifier	String	AR3_%s, A	
				dcterms:title	String	F	
				promcode_pm:producedBy	Resource		WI3[mostRecent]
	<END OF MAPPING RULE>						

2.3.3 Excel 用クライアントの修正

Eclipse Lyo の Excel 用クライアントは OSLC ChangeManagement を対象としているため，PROMCODE のリソースモデルに対応するためには VB マクロを修正する必要がある．たとえば，PROMCODE モデルの WorkItem を対象にしたい場合には以下のように修正する．

1. Excel 用クライアントを起動し，Visual Basic エディターを起動
2. 標準モジュール JazzIF を開く
3. xmlns:oslc_cm='http://open-services.net/ns/cm# を xmlns:promcode_pm='http://promcode.org/ns/pm# に変更
4. //oslc_cm:ChangeRequest を //promcode_pm:WorkItem に変更
5. "http://open-services.net/ns/cm#ChangeRequest"を "http://promcode.org/ns/pm#WorkItem"に変更

上記ステップ４およびステップ５で WorkItem を ScopeItem などとすることにより，他のクラスに対応することができる．

2.4 データフォルダの指定

PROMCODE リソースモデルのデータである Excel ファイルと，マッピングルールである Excel ファイルはデータフォルダに格納されている必要がある．前章において作成したこれらのファイルは，org.eclipse.lyo.samples.excel プロジェクトの下のフォルダ，repository¥projectA に置かれているが，データフォルダを別の場所に置く事もできる．
データフォルダは，org.eclipse.lyo.samples.excel プロジェクト直下にあるコンフィギュレーションファイル config.xml で以下のように記述されている．

```
<entry key="repositoryLocation">repository</entry>
```

この例では，データフォルダは相対パスで指定されており，org.eclipse.lyo.samples.excel プロジェクト直下のフォルダ repository を指している．Excel 用アダプタで別のデータフォルダを使用したい場合は，この設定を変更すればよい．
なお，データファイル，マッピングルールファイルは，必ずデータフォルダの下にサブフォルダを作成してから，そこに格納する．サブフォルダの名前は，Excel 用アダプタにアクセスする際の URL の一部として使われる．たとえばサブフォルダが projectA という名前であるので，リソースをクエリーするための URL は以下のようになる．

- http://localhost:8080/oslc-excel/rest/services/**projectA**/query/scopeitem

このように，サブフォルダの名前を変更した場合は，URL を変更しなければならないことに注意する．

3 Excel 用アダプタの処理フロー

PROMCODE コンシューマが，Excel 用アダプタに対してリソースのクエリーのリクエストをした場合の Excel 用アダプタ内の大まかな処理の流れは以下のようになる．

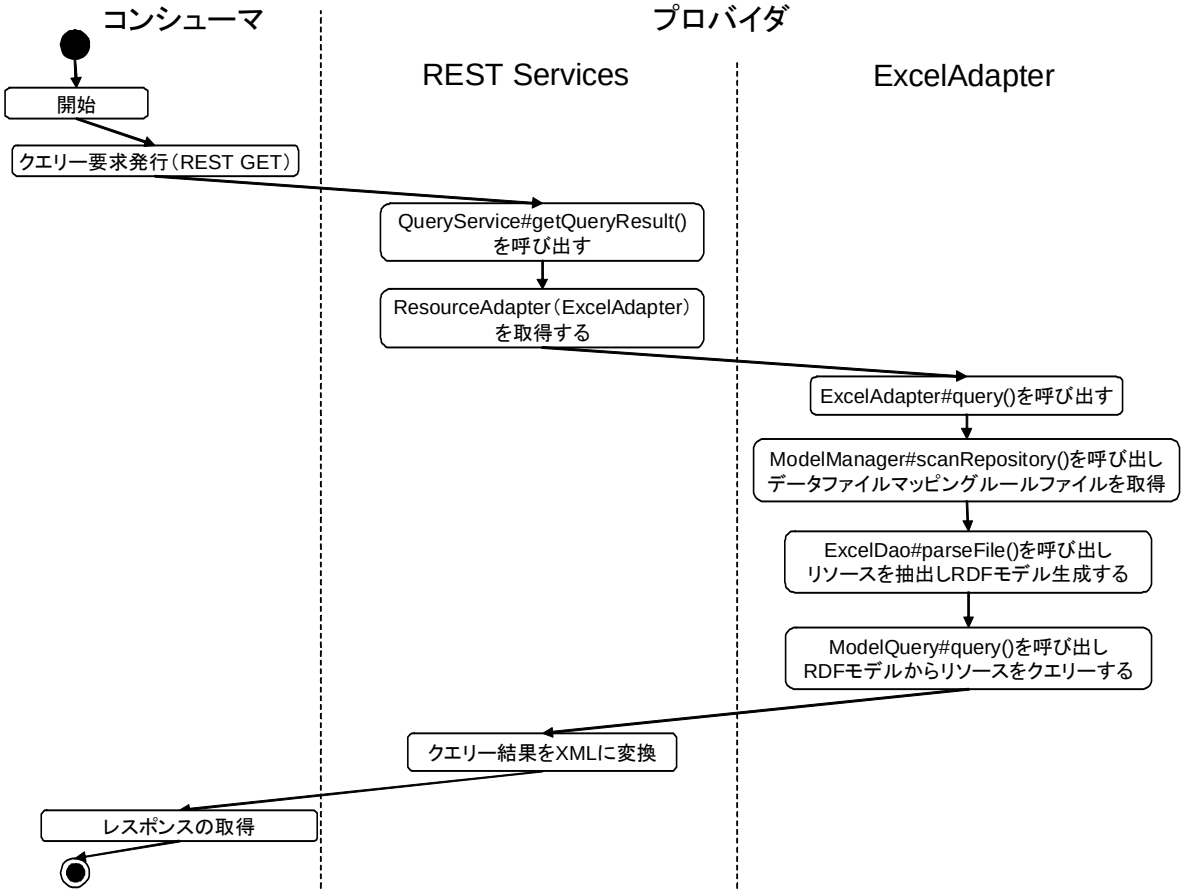

図 3，クエリー要求に対する処理の概要

コンシューマの要求に対して，JAX-RS のフレームワークによりまず QuerySercice#getQueryResult()が呼び出される．QueryService#getQueryResult()では URL に対応した ResourceAdapter のインスタンス取得し，query メソッドを呼び出す．なお，ここで取得される ResourceAdapter は ExcelAdapter のインスタンスである．ExcelAdapter#query()メソッドでは，リソースを抽出するために ModelManager#scanRepository()メソッドを呼び出し，データフォルダに置かれているデータファイルとマッピングルールファイルを取得する．さらに ExcelDaoImpl#parseFile()メソッドを呼び出し，データファイルのコンテンツを PROMCODE のリソースモデルへと変換し，RDF モデルを生成する．ExcelAdapter#query()はさらに ModelQuery#query()を呼び出し，SPARQL クエリーを用いて必要なリソースを RDF モデルから取得する．取得されたリソースは ResourceSet#outputAsXML()メソッドによって RDF/XML 形式へシリアライズされ，クライアントにレスポンスとして返送される．
前述の通り，これらのステップの内，主に拡張，修正の対象となるのは上記処理フローで示される，データファイルからリソースの抽出を行うメソッドである ExcelDaoImpl#parseFile()である．

4 マッピングルールの定義

マッピングルールは mapper.xls という名前の Excel ファイルに記述し，データフォルダに置くことに

よって Excel 用アダプタに読み込まれる．
定義は Excel ファイルの 3 行目以降に記述するリソース定義の集まりであり，一つのリソース定義はたとえば以下のようになる（表 3）．

表 3，マッピングルールの例

SI2	promcode_pm:ScopeItem	Sheet1,2,*,exist(C)	SI2_%s,A			
				dcterms:identifier	String	SI2_%s,A
				dcterms:title	String	C
				promcode_pm:composedBy	Resource	SI1[mostRecent]

リソース定義の各カラムに記述する情報の概要は以下の通り．
1. リソース定義の名前
2. PROMCODE リソースモデルの型
3. データファイルの検索範囲と条件
4. URL の構築式
5. プロパティの型
6. プロパティ値のデータ形式
7. データファイル内でプロパティ値が記述されている場所
8. リソースの参照式

以下に，各カラムの内容についてさらに説明する．

4.1リソース定義の名前

リソース定義を表す名前である．ユニークであれば任意の名前でよい．リソース間のリンクを作成する場合，別のリソース定義から参照される．

4.2PROMCODE リソースモデルの型

リソース定義から生成される PROMCODE リソースの型を表す．表 3 の例ではリソース ScopeItem が生成される．

4.3データファイルの検索範囲と条件

カンマで区切られた 4 つの値からなる情報で，Excel データファイル内の検索範囲と条件を表す．検索範囲内で条件が満たされた場合，このマッピング定義が有効となり，マッピング定義に従ってデータファイルから情報が読み出され PROMCODE リソースが生成される．
各値は以下の通り．
1. Excel のシート番号もしくはシートの名前．
2. 検索範囲である最初の行番号．
3. 検索範囲である最後の行番号．最終行までが検索範囲である場合特殊文字「*」を指定できる．
4. 検索条件．現在用意されている検索条件は以下の通り．
   (ア) exist(cell)： cell で指定されるセルに値がある場合マッチする．
   (イ) notexist(cell)： 指定したセルに値が無い場合マッチする．

表 3 の例にある「Sheet1,2,*,exist(C)」は名前が「Sheet1」であるシートの，2 行目から最終行までを検索範囲とし，その行の C カラムに何らかの値がある場合に，リソース生成が行うことを示す．

4.4URL 構築式

PROMCODE リソースの URL を構築する情報であり，ここに指定された文字列がベースの URL に連

結されてしようされる．文字列リテラルを指定してもよいし，「%s」を文字列に含めて，特定のカラムの値を「%s」部分に挿入することもできる．
表 3 の例にある「SI2_%s,A」は，%s を A カラムの値で置き換え，A カラムの値がたとえば「1.1」の場合，「SI2_1.1」という文字列となり，ベース URL に連結される．

4.5プロパティの型

リソース定義から生成される PROMCODE リソースにセットされるプロパティの型を表す．プロパティの定義はリソース定義の次の行以降，複数行にわたって指定することができる．

4.6プロパティ値のデータ形式

プロパティの値の形式を指定する．たとえばプロパティが文字列型である場合，プロパティ値の形式として「String」を指定し，プロパティがリソース型である場合は「Resource」を指定する．

4.7データファイル内でプロパティ値が記述されている場所

プロパティが Resource 型以外である場合，値をどこから読み出すかを指定する．以下のようにいくつかの指定方法がある．

- カラム指定．たとえば「K」と指定するとリソースを読み出したデータファイルの行の K カラムから値を取得する．
- セル指定．たとえば「K10」と指定すると 10 行目，K カラムから値を取得する．
- 領域名指定．データファイルに指定されている領域の左上のセルから値を取得する．

なお，表 3 の例にある「SI2_%s,A」のように，構築式を書くことができる．この場合，%s を A カラムの値で置き換え，A カラムの値が例えば「1.1」の場合，「SI2_1.1」という文字列となり，プロパティの値として設定される．

4.8リソースの参照式

プロパティが Resource 型である場合に，参照先となるリソースをリソース定義の名前で指定する．参照先のリソース定義はこのリソース定義より前にマッピング定義ファイルに記述されている必要がある．複数の存在しうるリソースから一つを選び出すために，検索用の関数を指定する．例にある「SI1[mostRecent]」で指定される参照先は，「SI1」というリソース定義によって生成された PROMCODE リソースであり，かつ，このリソースを読み出したデータファイルの同じ行か最近の行で生成された PROMCODE リソースである．
検索用の関数は現在以下のものが用意されている．

- sameLine ･･･ 現在の行
- mostRecent ･･･ 現在の行か最近の行

5 Excel 用アダプタの拡張のヒント

様々なフォーマットの Excel データファイルに対応するには，前章で説明したマッピングルールの Excel ファイルに，検索範囲や条件式を記述して対応すればよい．しかし，想定されていないフォーマットのデータファイルを処理するために，Excel 用アダプタのプログラムの拡張が必要な場合がある．ここでは ExcelAdapter に必要な拡張について，いくつかの例を示す．

5.1マッピングルールの拡張

マッピング定義用のクラスは以下の通りである．

- org.eclipse.lyo.samples.excel.adapter.MapperTable

マッピングルールファイルを読み込み MappingEntry のインスタンスを保持するクラス. メソッド initialize()は引数で指定されたマッピング定義ファイルを読み込み，情報をパースし，必要な MappingEntry のインスタンスを作成する.

- org.eclipse.lyo.samples.excel.adapter.MapperEntry
  マッピングルールの各カラム情報を保持するクラス.

前述の，データファイルの検索条件やリソース参照式を拡張したい場合は，マッピングルールクラスは特に拡張する必要はなく，後述のデータファイルアクセスクラスを拡張するだけでよい．前章で説明したマッピングルールで表現できない場合は，MapperEntry にカラムに対応するエントリを追加して，MapperTable クラスのメソッド initialize()でマッピング定義の Excel ファイルから追加されたエントリを読み出すように拡張し，同時に，新しく追加されたエントリでデータファイルをアクセスするように，データファイルアクセスクラスを拡張する.

5.2データファイルアクセスクラスの拡張

マッピング定義に従って Excel データファイルを読み出すクラスは以下の通りである.

- org.eclipse.lyo.samples.excel.adapter.dao.internal.ExcelDaoImpl
  メソッド parseFile()は引数として指定された Excel データファイルを読み込み，情報を抽出して，RDF モデルを生成する.

以下に，想定される拡張として，データファイルの検索条件の拡張と，リソースの参照式の拡張について説明する.

5.2.1 データファイルの検索条件の拡張

前章で説明したように，データファイルの検索条件として，exist，notexist，not の 3 つが定義されている．ここで，新たな検索条件を追加する場合は以下の手順で拡張する.

- 検索条件の名前と引数を定義し，マッピング定義ファイルに記述する.
- メソッド ExcelDaoImpl.parseFile()内の，Excel データファイルの行を処理するループの最初に，exist，notexist，not の処理があるので，ここに新たな検索条件名に対応した処理を追加する．条件にマッチした場合は処理を続行し，マッチしない場合は以降のループ内の処理をスキップすればよい.

5.2.2 リソース参照式の拡張

前章で説明したように，リソース参照式として，sameLine と mostRecent が定義されている．新しいリソース参照式を追加する場合は，以下の手順で拡張する.

- リソース参照式の名前を定義し，マッピング定義ファイルに記述する
- リソース参照式は，プロパティがリソース型のときにリソース間のリンクを作成する場合に使われる．メソッド ExcelDaoImpl.parseFile()から呼び出されるメソッド processReference()で処理される．ここに，新しいリソース参照式の処理を追加する．なお，リンク作成の処理はメソッド addReferenceProperty()で行われる．リソースのリンクにバックリンクが必要な場合は config.xml にバックリンク情報を登録する.

拡張の例として，dcterms:contributor プロパティについて，FOAF の Person リソースを参照する場合を考える．この場合，外部で定義されている Person リソースへの URL を取得しなければならないが，たとえば以下のような新たな関数を定義して解決する.

```
E[searchPerson]
```

この関数を処理するためのコードは ExcelDaoImpl#processReference()等に追加する．仮想コードは以

下のようになる.

```
// searchPerson
suffixIndex = referenceDef.toLowerCase().indexOf("[searchPerson]");
if (suffixIndex > 0)  {
      セル（例えばEカラム）の値を取得
      セルの値からURLを作成（URL構築式はマッピングルール，config.xmlに指定してもよい）
      プロパティを作成し，リソースに追加
}
```

参考文献

[1] Hypertext Transfer Protocol -- HTTP/1.1
http://tools.ietf.org/html/rfc2616

[2] Open Services for Lifecycle Collaboration
http://open-services.net/

[3] Eclipse Lyo - Enabling tool integration with OSLC
http://eclipse.org/lyo/

[4] Representational State Transfer (REST)
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[5] Resource Description Framework (RDF)
http://www.w3.org/RDF/

[6] JAX-RS: The Java API for RESTful Web Services
http://jcp.org/aboutJava/communityprocess/final/jsr311/index.html