

データ構造変換ツール

NetConscious, Inc.

2006 年 3 月 31 日

目次

1	テクニカルアーキテクチャ	1
1.1	コマンド・ランタイム (アプリケーションプラットフォーム)	2
1.2	コマンド・アドオン (アドオンプログラム)	2
2	プロセッサクラス	4
2.1	FileWriter プロセッサ	5
2.2	CsvFileReader プロセッサ	5
2.3	CsvFileWriter	5
2.4	CSVSchemaReader	5
2.5	BuiltinFilter	6
2.6	CsvConfigReader	7
2.7	FeatureTypeFilter	8
2.8	GetFeatureFilterReader	8
2.9	MispSoapClient	9
2.10	XpathFilter	9
2.11	XslTransformer	9

データ構造変換ツールの概要

データ構造変換ツールは、基本機能アクセスツールが内部で使用しているデータ処理部分のライブラリである。データ構造変換ツールは、Payload インターフェース、Message インターフェース、Processor インターフェースの 3 種類の Java インターフェースで実装されており、これらのインターフェースを実装したクラスを組み合わせることによって、様々な処理に対応できる。

1 テクニカルアーキテクチャ

MISP 基本機能アクセスツールは、拡張性を考慮し、各コマンドに対応したコマンド・アドオンと、それを実行するコマンドランタイムの二つで構成されている。また、コマンド・アドオンは、Message インターフェース、Payload インターフェース、Processor インターフェースの 3 種類の Java インタフェースを基本

としている。

1.1 コマンド・ランタイム（アプリケーションプラットフォーム）

MISP 基本機能アクセスツールは、NetConscious が LGPL として開発しているアプリケーションプラットフォーム^{*1}を基に開発されている。

このアプリケーションプラットフォームは、コンポーネントプラグインアーキテクチャと DI コンテナを基本技術として実装しており、コマンドラインツールからサーバアプリケーションまでを実装することができる。今回、コンポーネントは使用していないが、ServletContainer コンポーネントを追加することにより、容易にサーバ機能を追加実装することができる。

1.2 コマンド・アドオン（アドオンプログラム）

各コマンドは、コマンド・アドオンとして、コマンド・ランタイム上で動作するアドオンプログラムとして開発されている。共通のランタイム部分とコマンドの処理部分を分離することにより、設計の柔軟性と Java クラスの可搬性を高めている。

1.2.1 オブジェクトモデル

各コマンドの基本処理フローは、データを入力し、処理し、出力するという共通のフローを持っており、これらは3つの基本的なインターフェースによってオブジェクトモデルが形成されている。データ本体を扱う Payload インターフェース、処理を行う Processor インタフェース、Processor と Payload を関連付ける Message インターフェースである。

Payload インターフェース Payload インターフェースは、データ本体をストリーム形式で扱うためのインターフェースである。本ツールのコマンドは、このインターフェースを派生した FileBasedPayload の実装クラスである StandardPayload クラスを使用しており、すべてのデータはファイルを主体として処理される。以下に、StandardPayload クラスの使用例を示す。

```
// メッセージオブジェクトの生成
StandardMessage message = new StandardMessage();
StandardPayload payload = new StandardPayload();

// リクエストファイルのアタッチ
payload.attach(new File("request.xml"));
message.setPayload(payload);
```

Processor インターフェース Processor インターフェースは、Message オブジェクトが保持する Payload オブジェクトを処理するためのインターフェースである。本コマンドで開発している Processor インターフェースを実装したクラスは、Payload オブジェクトを処理するために、システムプロパティと Message オ

^{*1} NetConscious BMX (BlueMeme/Basis eXpress Edition)

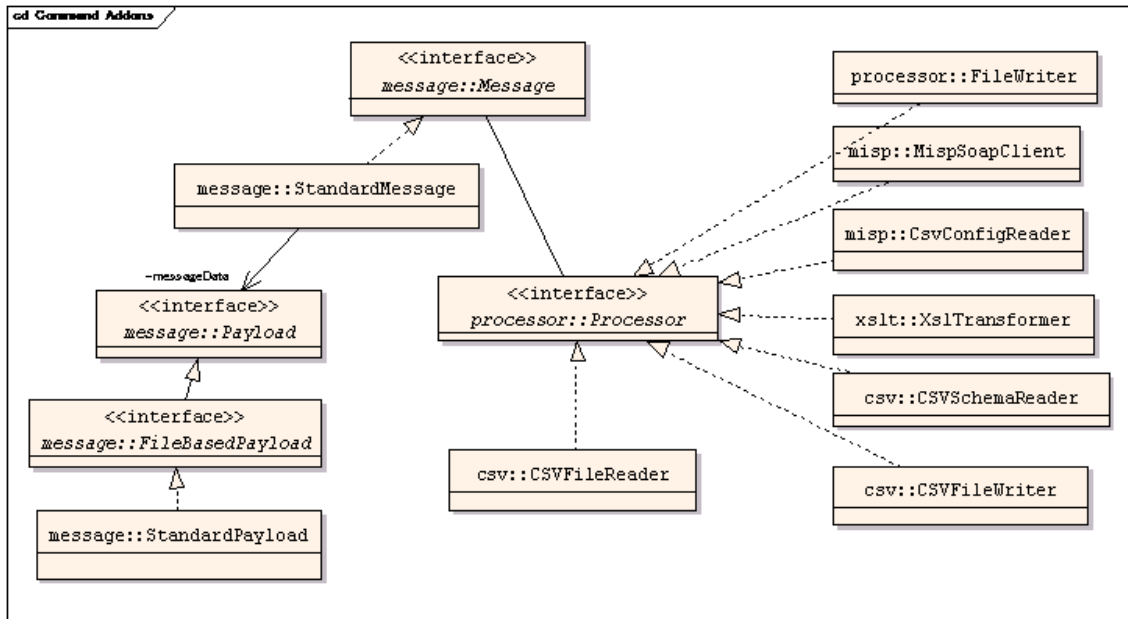


図1 クラスダイアグラム

プロジェクトが保持しているプロパティの双方を参照し、処理を実行している。以下に Processor の実装クラスである FileWriter の使用例を示す。

```

// メッセージオブジェクトの生成
StandardMessage message = new StandardMessage();
StandardPayload payload = new StandardPayload();

// リクエストファイルのアタッチ
payload.attach(new File("request.xml"));
message.setPayload(payload);

// プロパティの設定
message.setProperties(new Properties());
message.getProperties().setProperty(FileWriter.PROPS_OUTPUT_PATH, "result.txt");

// 処理の実行
FileWriter fileWriter = new FileWriter();
fileWriter.process(message);
  
```

Message インターフェース Message インターフェースは、Payload オブジェクトと複数の Processor オブジェクトを扱うためのインターフェースである。Message インターフェースを使用することにより、様々な変換処理を行う複数の Processor オブジェクトを組み合わせ、一つの処理を実行することができる。ま

た、Processor の処理順序とプロパティを指定できるため、複雑な多段階変換処理を実現することができる。本ツールのコマンドは、このインターフェースを実装した StandardMessage クラスを使用している。以下に StandardMessage クラスの使用例を示す。

```
// メッセージオブジェクトの生成
StandardMessage message = new StandardMessage();
StandardPayload payload = new StandardPayload();

// リクエストファイルのアタッチ
payload.attach(new File("request.xml"));
message.setPayload(payload);

// プロパティの設定
message.setProperties(new Properties());
message.getProperties().setProperty(FileWriter.PROPS_OUTPUT_PATH, "result.txt");
message.getProperties().setProperty(XslTransformer.PROPS_XSLT_FILE, "sample.xslt");

// 処理の実行
MispSoapClient soapClient = new MispSoapClient();
XslTransformer transformer = new XslTransformer();
FileWriter fileWriter = new FileWriter();

// プロセッサの登録 （追加した順序で実行される）
message.setProcessorList(new ArrayList());
message.getProcessorList().add(soapClient);
message.getProcessorList().add(transformer);
message.getProcessorList().add(fileWriter);

// 処理の実行
message.send();
```

2 プロセッサクラス

プロセッサクラスは、Processor インターフェースを実装したクラスである。本ツールは、複数のプロセッサクラスを組み合わせることによって、コマンド処理を実現している。

2.1 FileWriter プロセッサ

FileWriter は、Payload オブジェクトの内容を指定したファイルまたは標準出力に出力するプロセッサである。以下に、パラメータを示す。

パラメータ

filewriter.input.encoding 入力ファイルのエンコーディング
filewriter.output.encoding 出力ファイルのエンコーディング
filewriter.output.path 出力ファイル名 (filewriter.value.output.path.stdout を指定すると標準出力)

2.2 CsvFileReader プロセッサ

CsvFileReader プロセッサは、CSV を、CSVX 形式 (CSV の内部フォーマット) に変換するプロセッサである。以下に、パラメータを示す。

パラメータ

csvfilereader.csvfile.encoding 入力ファイルのエンコーディング
csvfilereader.csvx.encoding 出力ファイルのエンコーディング
csvfilereader.csvfile.comment CSV ファイルのコメントの文字
csvfilereader.csvfile.separator CSV ファイルの値のデリミタ文字
csvfilereader.csvfile.parenthesis CSV ファイルの値の括弧文字
csvfilereader.xslt.copy.template コピー用 XSLT ファイル

2.3 CsvFileWriter

CsvFileWriter は、CSVX 形式を、CSV に変換するプロセッサである。以下に、パラメータを示す。

パラメータ

csvfilewriter.output.encoding 入力ファイルのエンコーディング
csvfilewriter.output.path 出力ファイルのエンコーディング
csvfilewriter.xslt.csvx2csv.template CSVX 形式から CSV に変換するための XSLT ファイル

2.4 CSVSchemaReader

CSVSchemaReader は、CSV 形式を、スキーマ定義に変換するプロセッサである。以下に、パラメータを示す。

パラメータ

csvschemareader.csvfile.encoding 入力ファイルのエンコーディング

csvschemareader.csvschema.encoding 出力ファイルのエンコーディング
csvschemareader.csvfile.comment CSV ファイルのコメントの文字
csvschemareader.csvfile.separator CSV ファイルの値のデリミタ文字
csvschemareader.csvfile.parenthesis CSV ファイルの値の括弧文字

2.5 BuiltinFilter

BuiltinFilter は、GeometoryPropertyType(LineString/Point/Polygon) 部分を、MISP 形式または CSV 形式に変換するプロセッサである。MISP 形式と CSV 形式の変換方向は、BuiltinFilter のコンストラクタで決定される。

```
// CSV 形式から MISP 形式へ変換を行う場合
BuiltinFilter builtinFilter = new BuiltinFilter(BuiltinFilter.MODE_CSV2XML);
// MISP 形式から CSV 形式に変換を行う場合
BuiltinFilter builtinFilter = new BuiltinFilter(BuiltinFilter.MODE_XML2CSV);
```

以下に、パラメータを示す。

パラメータ

なし

2.5.1 GeometoryPropertyType の表現例

基本的にデータ値と CSV のカラムは、一対一でマッピングされるが、データ型が GeometoryPropertyType (LineString/Point/Polygon) の場合は、BuiltinFilter プロセッサクラスによって、特定のルールに従って変換することが出来る。以下に変換例を示す。

MISP による LineString の表現

```
<gml:Point>
<gml:coordinates>1.2345,6.7890</gml:coordinates>
</gml:Point>
```

CSV による LineString の表現

```
LINESTRING(1.2345 6.7890,1.1111 2.2222,3.3333 4.4444)
```

MISP による Point の表現

```
<gml:Point>
  <gml:coordinates>1.2345,6.7890</gml:coordinates>
</gml:Point>
```

CSV による Point の表現

```
POINT(1.2345 6.7890)
```

MISP による Point の表現

```
<gml:Polygon>
  <gml:outerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>1.2,6.7 1.1,2.2 3.3,4.4</gml:coordinates>
    </gml:LinearRing>
  </gml:outerBoundaryIs>
  <gml:innerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>1.2,6.7 5.6,7.7 8.8,9.9</gml:coordinates>
    </gml:LinearRing>
  </gml:innerBoundaryIs>
  <gml:innerBoundaryIs>
    <gml:LinearRing>
      <gml:coordinates>1.2,6.7 1.1,2.2 3.3,4.4</gml:coordinates>
    </gml:LinearRing>
  </gml:innerBoundaryIs>
</gml:Polygon>
```

CSV による Point の表現

```
POLYGON((1.2 6.7,1.1 2.2,3.3 4.4),(1.2 6.7,5.6 7.7,8.8 9.9),(1.2 6.7,1.1 2.2,3.3 4.4))
```

2.6 CsvConfigReader

CsvConfigReader は、CSV 変換ルール定義ファイルから CsvGetFeature 用 XSLT または CsvInsert 用 XSLT を生成するプロセッサである。

パラメータ

csvxsltgenerator.xslt.type 生成する XSLT の種類を指定する。CsvGetFeature であれば csvxsltgenerator.xslt.type.getfeature、CsvInsert であれば csvxsltgenerator.xslt.type.insert を指定する。
csvxsltgenerator.xslt.getfeature.csv.template CsvGetFeature 用 XSLT 生成用 XSLT
csvxsltgenerator.xslt.insert.csv.template CsvInsert 用 XSLT 生成用 XSLT
csvxsltgenerator.output.encoding 出力ファイルのエンコーディング

2.6.1 CSV 変換ルール定義

CSV 変換は、CSV 変換ルール定義ファイルを使用して、CsvFileReader プロセッサクラスと、CsvConfigReader プロセッサクラスを使用して変換を行う。CsvFileReader は、CSV 形式から内部形式である CSVX 形式に変換し、CsvConfigReader は、CSV 変換ルール定義ファイルより、CSVX と MISP 間の変換用の

XSLT を動的に生成する。CSV 変換ルール定義ファイルには、CSV のカラムと MISP の XML の XPath 形式を一对で記述する。以下に記述例を示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<cx:Config xmlns:cx="http://staff.aist.go.jp/i.noda/Standard/2005/CsvXml">
<cx:xml>
<xsd:schema id="BuildingInfo.xsd">
... スキーマ定義 (省略)
</xsd:schema>
</cx:xml>
<!-- CSV の変換ルールを定義 -->
<cx:csv element="BuildingFireInfo" cs="," rs="\n" quoteChar="'" commentPrefix="#">
<cx:columnList>
<!-- XPath に MISP の構造、name に CSV の構造を定義する -->
<cx:column xpath="id" name="id"/>
<cx:column xpath="damage/level" name="damagelevel"/>
<cx:column xpath="time/begin" name="beginntime"/>
<cx:column xpath="time/end" name="endtime"/>
</cx:columnList>
</cx:csv>
</cx:Config>
```

2.7 FeatureTypeFilter

FeatureTypeFilter は、スキーマ定義を、XPath 形式に変換するプロセッサである。

パラメータ

featuretypefilter.output.encoding 出力ファイルのエンコーディング
featuretypefilter.separator XPath とデータ型のセパレータ文字

2.8 GetFeatureFilterReader

GetFeatureFilterReader は、様々なフィルタを、GetFeature リクエストのフィルタに変換するプロセッサである。^{*2}

パラメータ

mispsoapclient.filter.type フィルタの種類を指定する。XML 形式であれば mispsoapclient.filter.xml、S 式であれば mispsoapclient.filter.s.expression を指定する。

^{*2} 現在、MISP の XML 形式のみ対応

2.9 MispSoapClient

MispSoapClient は、Payload オブジェクトの内容をサーバへ送信し、レスポンスを Payload オブジェクトに格納するプロセッサである。

パラメータ

mispsoapclient.server.host	サーバのホスト名またはIPアドレス
mispsoapclient.server.port	サーバのポート
mispsoapclient.xslt.copy.template	コピー用 XSLT ファイル

2.10 XpathFilter

XpathFilter は、XML 形式を XPath 形式に変換するプロセッサである。

パラメータ

xpathfilter.output.encoding	出力ファイルのエンコーディング
xpathfilter.separator	XPath を値のセパレータ文字列

2.11 XsltTransformer

XsltTransformer は、XML ファイルを指定された XSLT ファイルを用いて別の形式に変換するプロセッサである。

パラメータ

xslttransformer.xml.encoding	出力ファイルのエンコーディング
xslttransformer.xslt.file	XSLT ファイル
xslttransformer.omit.xml.declaration	XML 宣言を出力するかどうかのフラグ (yes/no)
xslttransformer.indent	インデントを出力するかどうかのフラグ (yes/no)