

スキーマ生成ツール

NetConscious, Inc.

2006 年 3 月 31 日

目次

1	コマンドの使用方法	1
1.1	コマンド種類と用途	1
1.2	Csv2Schema コマンド	1
1.3	-debug オプションの使用について	4
2	テクニカルアーキテクチャ	5
2.1	コマンドツール群のディレクトリ構成	5
2.2	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)	5
2.3	コマンド・アドオン (アドオンプログラム)	6
2.4	コマンドの実行	6
2.5	システムプロパティによるパラメータ設定	9

スキーマ生成ツールの概要

スキーマ生成ツールは、CSV 形式から MISP のスキーマ定義に変換するツールである。

1 コマンドの使用方法

1.1 コマンド種類と用途

スキーマ生成ツールが提供するコマンドは、Csv2Schema である。

1.2 Csv2Schema コマンド

Csv2Schema は、CSV で定義されたスキーマ定義を、MISP のスキーマ定義 (XML Schema) に変換するためのコマンドツールである。Csv2Schema を用いることにより、複雑な XML Schema を記述することなく、スキーマ定義を生成することができる。

```
usage: Csv2Schema [-debug] [-out OUTPUT_FILE] [-in INPUT_FILE]
           -host HOSTNAME -csvcfg CONFIG_FILE
-debug          中間ファイルを削除しません。
-in <INPUT_FILE> 入力ファイル名 (CSV ファイル)
-out <OUTPUT_FILE> 出力ファイル名 (XML ファイル)
```

schema.csv をスキーマ定義に変換して結果をコンソール出力する場合の例

```
Csv2Schema -in schema.csv
Csv2Schema < schema.csv
```

schema.csv をスキーマ定義に変換して結果を result.xml ファイルに出力する場合の例

```
Csv2Schema -in schema.csv -out result.xml
Csv2Schema -out result.xml < schema.csv
```

Csv2Schema コマンドのパラメータ

csv2schema.create.register.request.xslt.template RegisterFeatureType リクエストを生成する XSLT
ファイル

1.2.1 スキーマの CSV 形式

Csv2Schema コマンドが使用する CSV 形式と変換結果を以下に示す。

ID	http://staff.aist.go.jp/i.noda/Rescue/RandomCity/2.0/Building.xsd		
Namespace	gml	http://www.opengis.net/gml	
Namespace	rcbase	http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base	
TargetNamespace	http://staff.aist.go.jp/i.noda/Rescue/RandomCity/2.0		
Import	http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base		
	location	http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base/common.xsd	
FeatureType	Building	BuildingType	
ElementType	BuildingType		
	extension	rcbase:GeometryFeatureType	
	attribute	bid	integer
	element	representativePoint	geometryPropertyType
	element	structureType	integer
	element	nFloors	integer
	element	damage	damageType
	element	meshId	rcbase:IDType

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  id="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/2.0/Building.xsd"
  targetNamespace="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/2.0"
  xmlns="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/2.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:rcbase="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <xsd:import
    namespace="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base"
    schemaLocation="http://staff.aist.go.jp/i.noda/Rescue/RandomCity/base/common.xsd"/>

  <xsd:element name="Building" type="BuildingType"/>

  <xsd:complexType name="BuildingType">
    <xsd:complexContent>
      <xsd:extension base="rcbase:GeometryFeatureType">
        <xsd:attribute name="bid" type="integer"/>
        <xsd:sequence>
          <xsd:element name="representativePoint" type="geometryPropertyType"/>
          <xsd:element name="structureType" type="integer"/>
          <xsd:element name="nFloors" type="integer"/>
          <xsd:element name="damage" type="damageType"/>
          <xsd:element name="meshId" type="rcbase:IDType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="damageType">
    <xsd:complexContent>
      <xsd:sequence>
        <xsd:element name="grade" type="integer"/>
        <xsd:element name="prob" type="float"/>
        <xsd:element name="direction" type="integer"/>
      </xsd:sequence>
    </xsd:complexContent>
  </xsd:complexType>

</xsd:schema>

```

1.3 -debug オプションの使用について

デバックオプションを使用することにより、各コマンドツールが一時的に使用するファイルをコマンド終了後も残すことができる。一時的に使用するファイルには、DaRuMa サーバに送信する SOAP のメッセージや、レスポンスが含まれるため、障害時の問題発見にデバックオプションを使用することができる。

2 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。

2.1 コマンドツール群のディレクトリ構成

本ツールのディレクトリ構成は、大きく分けて、コンポーネントプラグインのための components、各種設定ファイルを格納する etc、コマンド・ランタイムのライブラリを格納する lib、コマンド・アドオン本体及びライブラリを格納する work で構成されている。以下にディレクトリ構成を示す。

<HOME>

```
components <-機能拡張を行う場合のコンポーネントを格納するディレクトリ
etc <-各コマンドの設定ファイルを格納するディレクトリ
lib <-コマンド実行環境の基本ライブラリファイルを格納するディレクトリ
work <-コマンド実行環境の作業用ディレクトリ
    beans <-コマンド実行環境をカスタマイズする場合に使用するディレクトリ
    classes <-拡張クラスを格納するためのディレクトリ
    components <-各コンポーネントが使用する作業用ディレクトリ
    lib <-コマンド実行環境の拡張ライブラリファイルを格納するディレクトリ
    log <-ログフォルダ
    samples <-サンプルデータディレクトリ
    sysprops <-各コマンドのパラメータをカスタマイズする場合に使用するディレクトリ
    xsl <-各コマンドが使用する XSLT ファイルを格納するディレクトリ
```

2.2 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} NetConscious BMX (BlueMeme/Basis eXpress Edition)

2.3 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。

2.4 コマンドの実行

本ツールのコマンドは、すべてスクリプトを使用して実行することが可能であるが、Java コマンドを使用して起動することができる。各コマンドは、全てコマンド・ランタイム上で動作するコマンド・アドオンであるため、コマンド・ランタイムを起動後、コマンド・アドオンをロードして実行する必要がある。

本ツールは、起動された JavaVM のシステムプロパティから JAVA_HOME の設定値を取得するため、JAVA_HOME の設定は必須ではない。

2.4.1 コマンド・ランタイムの起動

コマンド・ランタイムを起動するには、ランタイムのホームディレクトリをシステムプロパティで指定し、startup.jar を -jar オプションを指定して Java コマンドを実行する必要がある。ランタイムのホームディレクトリを省略した場合は、Java VM を起動したカレントディレクトリをホームとして設定する。以下に、コマンド・ランタイムの起動方法と示す。

```
set DARUMA_HOME=c:\darumatool
java -Dapplication.home="%DARUMA_HOME:\=/" -jar startup.jar
```

コマンドを実行するカレントディレクトリが本ツールのホームである場合は、-Dapplication.home は省略可能である。

2.4.2 コマンド・アドオンのロード

コマンド・ランタイム単体を起動した場合、etc ディレクトリ以下の default.system.properties.xml を読み込み標準の StandardApplication クラス (標準のコマンド・アドオン) が起動する。特定のコマンド・アドオンをロードするには、コマンド・ランタイムの起動時に、システムプロパティ system.properties.xml に対して、ロードしたいコマンド・アドオンが記述されたプロパティファイルを設定する必要がある。以下に、コマンド・アドオンのロード方法を示す。

```
set DARUMA_HOME=c:\darumatool
java -Dapplication.home="%DARUMA_HOME:\=/"
    -Dsystem.properties.xml="./etc/properties.xml" -jar startup.jar
```

コマンドを実行するカレントディレクトリが本ツールのホームである場合は、-Dapplication.home は省略可能である。

2.4.3 コマンド・アドオンの設定ファイル

コマンドアドオンの設定ファイルは、etc ディレクトリの各コマンド名のディレクトリ以下に格納されている。設定ファイルには、system.properties.xml と system.context.xml の2種類の XML ファイルがある。

system.properties.xml ファイル system.properties.xml は、コマンド・ランタイム及びコマンド・アドオンの動作を設定するものであり、以下の system.context.xml のパスや、起動するメインクラスを記述する。サーバーモードの切り替えもこのファイルを行うことができる。メインクラスを変更することによって、様々なコマンド・アドオンを同一のコマンド・ランタイム上で起動することができる。以下に system.properties.xml の設定例を示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<!--モード切替 -->
<entry key="system.server.mode">false</entry>
<entry key="system.silent.mode">true</entry>
<!-- システムパラメータ -->
<entry key="system.context.cfg.xml">
${application.home}/etc/darumaclient/system.context.xml
</entry>
<!-- ランタイムパラメータ -->
<entry key="application.name">DarumaClient for MISP</entry>
<entry key="application.description">
2006 AIST DaRuMa Project - Daruma Client for MISP
</entry>
<entry key="application.main.class">daruma.main.DarumaClient</entry>
<!-- アドオンパラメータ -->
<entry key="mispssoapclient.xslt.copy.template">
${application.home}/work/xsl/standard_copy.xslt
</entry>
<entry key="darumaclient.retrieve.soap.body.xslt.template">
${application.home}/work/xsl/retrieve_soap_body.xslt
</entry>
</properties>
```

system.context.xml ファイル system.context.xml は、DI コンテナで処理される Bean 定義ファイルである。本ツールでは、コマンドラインのオプション及びパラメータの定義に使用している。以下に system.context.xml の設定例を示す。

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
</bean>
<bean id="commandLineHelper" class="application.runtime.CommandLineHandler">
<property name="commandName"
value="ShowFeatureTypes [-debug] [-out OUTPUT_FILE] -host HOSTNAME"/>
<property name="parser">
<bean class="org.apache.commons.cli.BasicParser"/>
</property>
<property name="optionList">
<list>
<bean class="application.runtime.CommandLineOption">
<property name="optionName" value="debug"/>
<property name="description" value="中間ファイルを削除しません。"/>
<property name="hasValue" value="false"/>
<property name="required" value="false"/>
<property name="argName" value=""/>
<property name="argCount" value="0"/>
<property name="optionalArg" value="false"/>
</bean>
<!-- Server Parameters -->
<bean class="application.runtime.CommandLineOption">
<property name="optionName" value="host"/>
<property name="description" value="サーバのホスト名又は IP アドレス"/>
<property name="hasValue" value="true"/>
<property name="required" value="true"/>
<property name="argName" value="HOSTNAME"/>
<property name="argCount" value="1"/>
<property name="optionalArg" value="false"/>
</bean>
<bean class="application.runtime.CommandLineOption">
<property name="optionName" value="out"/>
<property name="description" value="出力ファイル名 (XML ファイル)"/>
<property name="hasValue" value="true"/>
<property name="required" value="false"/>
<property name="argName" value="OUTPUT_FILE"/>
<property name="argCount" value="1"/>
<property name="optionalArg" value="false"/>
</bean>
</list>
</property>
</bean>
</beans>

```


2.5 システムプロパティによるパラメータ設定

各 Processor オブジェクトやコマンドは、システムプロパティによって、動作が決定される。システムプロパティを設定する方法は、コマンド・ランタイム及びコマンド・アドオンの双方で用意されており、適用される優先度によって使い分けることができる。

1. Java VM の-D オプションによる設定値
2. etc ディレクトリ以下の default.system.properties.xml ファイルの値
3. etc ディレクトリ以下の各コマンド用ディレクトリの system.properties.xml ファイルの値
4. work/sysprops ディレクトリ以下の拡張子が xml であるファイルの値
5. プログラム内で System オブジェクトに対して設定した値