

7-1 Xi拡張の基礎知識

これまで見てきたように、Xiでは、一般的なプログラミングロジックの記述、ファイルの読み書きやデータベースへのアクセスなど、さまざまなことができます。では、このようなXiの機能を支えているものは何なのでしょう？ Xiは多機能である反面、その仕様は非常にシンプルです。仕様では下記の項目を規定しているだけです。

- ・ ツリーに関する仕様
- ・ 関数に関する仕様
- ・ 変数に関する仕様
- ・ 式に関する仕様

ただし、たったこれだけの仕様では、有用なプログラミング言語にはなりません。そこで、XiにはXi自身を拡張するための機構が用意されています。それが**タグライブラリ**と**忍者**です。変数の定義、条件分岐、それにループなど、Xiの基本的な部分も実はこの拡張機能を利用して構築されているのです。

例えば、`<xi:variable>`は「変数を定義する」関数で、標準タグライブラリに含まれます。また、タグライブラリと忍者以外にも「実行環境の拡張（オプション）」という機能が存在します。それは、Xi自体を拡張するのとはやや趣きが異なり、文字通りXiの実行環境自体を増やす拡張です。

7-1-1 タグライブラリとは？

タグライブラリとは文字どおり、XMLのタグを使用した複数のXi関数の集まりです。これは、XML形式の拡張で、主にタグで表現されるXMLのツリーを生成したり、プログラムの実行を制御する機能を追加するために使用されます。つまり、新しいXi関数を作成することで、Xiを拡張するのです。

7-1-2 忍者とは？

忍者は、Xiの式の中で使用されるオブジェクトです。Xiプログラム中の文字列や数値も忍者の1つです。そのほかにも、Java忍者のように、外部環境にアクセスするためのインタフェースとして働くものもあります。

忍者は、オブジェクト型の種類を増やすことによってXiを拡張します。「XiはXSLTに似た言語だが、CやPerlなどの一般的なプログラミング言語により近い」とよく言われますが、これは忍者の存在が大きいのではないのでしょうか。

7-1-3 実行環境の拡張とは？

現在のBxiはサブレットやコマンドプロンプトなどから利用することができます。このように、異なる環境からXiを利用できるようにすることを**実行環境の拡張（オプション）**と言います。上記のほかにも、Apache Cocoon フレームワーク^(注1)の中でXiを動作させることもできます。実行環境の拡張方法を知っていると、自分で作成したアプリケーションにXiを組み込むこともできます。

以上のことを踏まえると、Xiの拡張の方法は表7-1-1のようにまとめることができます。

表 7-1-1 Xi の拡張の方法

| 拡張対象 | 拡張箇所 | 拡張目的 | 命名規則 | 感覚 |
|---------|------------|---------------------------|------------------|---------------|
| タグライブラリ | Xi関数の拡張 | 結果ツリーフラグメントの生成、実行フローの制御など | 「Xi-」を頭に付加する | XMLらしい拡張 |
| 忍者 | オブジェクト型の拡張 | オブジェクトの作成、外部環境へのアクセスなど | 「-Ninja」を末尾に付加する | プログラミング言語的な拡張 |
| 実行環境の拡張 | 実行環境の整備 | 実行環境の整備 | 「Opt-」を頭に付加する | 処理系の拡張 |

7-1-4 API解説

ここでは、忍者やタグライブラリ、実行環境の拡張を作成するにあたって、最低限必要なクラスとインタフェースについて説明します。第2節以降を読むにあたっての参考にしてください。

❖ Ninja インタフェース

忍者を作成するために必要なメソッドを定義しているインタフェースです。作成する忍者が持つプロパティを設定 / 取得するためのメソッドや、忍者が持つメソッドを呼び出すためのメソッドが定義されています。代表的なメソッドとしては、表7-1-2のものが存在します。

表 7-1-2 org.baykit.xi.ninja.Ninja インタフェースのメソッド

| メソッド名 | 説明 |
|-----------------|--|
| invoke | 忍者のメソッドを起動する際に呼び出される |
| getProperty | 忍者が持つプロパティを取得する際に呼び出される |
| setProperty | 忍者が持つプロパティを設定する際に呼び出される |
| getIterator | xi:for-each 関数のselect引数などで、コレクションの各項目にアクセスするイテレータを取得する際に呼び出される |
| getStringValue | 忍者を文字列に変換する際に呼び出される |
| getBooleanValue | 忍者を論理値に変換する際に呼び出される |
| getNumberValue | 忍者を数値に変換する際に呼び出される |
| getNodeValue | xi:copy-of 関数などで忍者を結果ツリーフラグメントに変換する際に呼び出される |
| getClassName | この忍者のクラス名を取得する際に呼び出される |

注1 : <http://xml.apache.org/cocoon/>

❖ NinjaBase クラス

NinjaBaseクラスは、Ninjaインタフェースのメソッドのデフォルト実装を持つ抽象クラスです。このクラスには、Ninjaインタフェースのメソッドに対応するdoXXXメソッドが用意されており、必要なdoXXXメソッドをオーバーライドするだけで忍者が作成できるようになっています。このクラスは、開発者がNinjaインタフェースのメソッドをすべて実装しなければならない労力を軽減するために存在します。

❖ ManagableNinja インタフェース

ManagableNinjaインタフェースは、忍者が初期化 / クリーンアップされる時にコールバックされるメソッドを定義するインタフェースです。Engineインタフェースを実装したクラスのインスタンスを取得したい場合や、Xiプロセッサが終了する前にリソースの解放などを行いたい場合に、このインタフェースを実装します。このインタフェースのメソッドには表7-1-3のものがあります。

表 7-1-3 org.baykit.xi.ninja.ManagableNinja インタフェースのメソッド

| メソッド名 | 説明 |
|-----------------|---|
| initializeNinja | Xiプロセッサ起動時に忍者の初期化を行う。Engineインタフェースを引数にする |
| terminateNinja | Xiプロセッサ終了時に忍者のクリーンアップを行う。このメソッドは忍者がグローバル変数に割り当てられている場合にのみ呼び出される |

❖ TreeFragment クラス

TreeFragmentクラスは、Xiの結果のツリーフラグメント型に相当するクラスです。ノードの挿入や、追加したノードのリストを取得することができます。Xi関数の返り値は結果ツリーフラグメント型なので、タグライブラリを実装する場合はこのクラスを必ず使用します。代表的なメソッドとして、表7-1-4に挙げたものがあります。

表 7-1-4 org.baykit.xi.ninja.TreeFragment クラスのメソッド

| メソッド名 | 説明 |
|--------------|---------------------------------|
| addNode | ノードを追加する |
| insertNode | ノードをこのTreeFragmentの先頭に挿入する |
| getNodeList | このTreeFragmentが保持するノードのリストを取得する |
| getNodeCount | このTreeFragmentが保持するノードの個数を返す |

❖ NodeSet クラス

NodeSetクラスは、Xiのノード集合型に相当するクラスです。TreeFragmentクラスと同じメソッドを持ち、さらにTreeFragment型に変換するtoTreeFragmentメソッドを持っています。

❖ Function インタフェース

Function インタフェースは、各 Xi 関数が実装すべきインタフェースです。代表的なメソッドとして、表 7-1-5 に挙げたものがあります。

表 7-1-5 org.baykit.xi.core.Function インタフェースのメソッド

| メソッド名 | 説明 |
|-------------|---------------------|
| execute | この関数を実行する際に呼び出される |
| validate | この関数の構造をチェックする |
| getArgument | 関数の実行時に引数（属性値）を取得する |
| getParent | この関数の親関数を取得する |
| getChildren | この関数のボディを取得する |

❖ FunctionBase クラス

FunctionBase クラスは、Function インタフェースのメソッドのデフォルト実装を持つ抽象クラスです。Function インタフェースを実装してタグライブラリを作成する場合は多くのメソッドを実装する必要がありますが、このクラスを継承すれば、doExecute メソッドだけをオーバーライドするだけで済みます。

❖ Engine インタフェース

Engine インタフェースは、Xi を処理するプロセッサの中核をなすインタフェースです。式を評価したり、DOM の Document オブジェクトを取得することができます。デバッグ用のメソッドも持っているので、動作を確認したい場合には有用です。主なメソッドとして、表 7-1-6 に挙げたものがあります。

表 7-1-6 org.baykit.xi.core.Engine インタフェースのメソッド

| メソッド名 | 説明 |
|--------------------|---|
| executeBody | Xi 関数のボディを実行する |
| evaluate | 式を評価する |
| getVariableManager | 変数を管理しているオブジェクトを取得する。変数の宣言、参照、更新が可能 |
| debug | デバッグメッセージを出力する |
| warn | 警告メッセージを出力する |
| trace | トレース情報を出力する |
| getNodeFactory | DOM の Document オブジェクトを取得する。各種ノードを生成するのに使用 |
| getResultTree | 現在の Xi 関数の実行結果であるツリーを取得する |

❖ Processor インタフェース

Processor インタフェースは、Xi スクリプトを処理するプロセッサを表すインタフェースです。Xi スクリプトを実行させたり、Xi プロセッサに対して忍者やタグライブラリを登録することができます。主に実行環境の拡張を行う場合に使用し、代表的なメソッドとしては、表 7-1-7 に挙げたものがあります。

表7-1-7 org.baykit.xi.app.Processor インタフェースのメソッド

| メソッド名 | 説明 |
|------------|--|
| initialize | Xi プロセッサを初期化する。引数にProperties オブジェクトを与えることで、忍者やタグライブラリを追加することができる |
| execute | Xi スクリプトを実行する |
| addGlobal | 忍者をグローバル変数として追加する |
| addTaglib | タグライブラリを追加する |

❖ ResultTree クラス

ResultTree クラスは、Xi スクリプトを処理した結果ツリーを表すクラスです。この結果ツリーが持つ DocumentFragment オブジェクトを取得したり、Xi スクリプトの実行が完了したかどうかを判断したり、`<pr:output>` で指定した出力形式を取得したりすることができます。代表的なメソッドとしては、表7-1-8 に挙げたものがあります。

表7-1-8 org.baykit.xi.app.ResultTree クラスのメソッド

| メソッド名 | 説明 |
|--------------|--|
| getNodes | この結果ツリーが保持する DocumentFragment オブジェクトを取得する |
| isTerminated | Xi スクリプトの実行が強制終了されたかどうかを判断する |
| getMethod | <code><pr:output method></code> で指定した出力メソッドを取得する |
| getEncoding | <code><pr:output encoding></code> で指定した出力エンコーディングを取得する |
| getMediaType | <code><pr:output media-type></code> で指定したメディアタイプを取得する |

❖ XiException クラス

XiException クラスは、Xi スクリプトを実行した際に発生した例外（エラー）をスローするためのクラスです。この例外をスローする際は、ほかの例外をラップすることができます。代表的なメソッドとして、表7-1-9 に挙げたものがあります。

表7-1-9 org.baykit.xi.app.XiException クラスのメソッド

| メソッド名 | 説明 |
|--------------|--|
| XiException | コンストラクタ。引数として別の例外を与えることにより、その例外をラップできる |
| getRootCause | ラップした例外を取得する |

7-1-5 前準備

○ 動作環境

本節の動作確認には、JDK と BayServer が必要です。第2章「2-1 BayServer のインストール」を参考に、これらをインストールしましょう。また、第2章「2-2 Xi/Xi-Extension のインストール」にも目を通しておいってください。筆者は、表7-1-10 の環境でサンプルプログラムの動作確認を行っています。

表 7-1-10 動作環境

| 対象 | ソフトウェア | バージョン | ダウンロード先 |
|-------------|--------------|----------|---|
| OS | Windows 2000 | SP3 | |
| Java | Sun J2SDK | 1.4.1_01 | http://java.sun.com/j2se/1.4/ja/ |
| アプリケーションサーバ | BaySever | 1.1.5 | http://www.baykit.org/bserv/index.html |

この章ではJDKがC:\j2sdk-1.4.1_01に、BayServerがC:\%BSENVにインストールされているものとして扱います。

○ ディレクトリ構成

C:\%BSENV\webappsの下に「xi-ext」というディレクトリを作成します。ここがメインのディレクトリとなります。なお、順を追って説明しますが、最終的なディレクトリ構成は図7-1-1のようになります。

○ プログラム動作に必要な作業

また、この節で作成したサンプルプログラムを動作させるためには、以下の作業を行う必要があります。

WEB-INF、WEB-INF\classes、WEB-INF\libディレクトリの作成

C:\%BSENV\bxi\WEB-INF\lib下にあるjarファイルをxi-ext\WEB-INF\libディレクトリにコピー
system.xmlにWebアプリケーションの設定を追加

WEB-INFディレクトリの下にweb.xmlを作成

は、system.xmlの161行目付近にリスト7-1-1の内容を追加します。

リスト7-1-1 system.xmlに追加するリスト

```

1: <application name="Xi-Extension"
2:           context-path="/xi-ext"
3:           document-base="webapps/xi-ext"
4:           log="false"
5:           reload="never" />

```

のweb.xmlはリスト7-1-2のようになります。これは、C:\%BSENV\bxi\WEB-INF\web.xmlをそのままWEB-INFディレクトリにコピーしてもよいでしょう。

リスト7-1-2 Hello 忍者の使用例 (xi-ext\WEB-INF\web.xml)

```

1: <?xml version="1.0" encoding="ISO-8859-1"?>
2:

```

```
3: <!DOCTYPE web-app
4:     PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
5:     "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
6:
7: <web-app>
8:
9:     <servlet>
10:         <servlet-name>
11:             Bxi
12:         </servlet-name>
13:         <servlet-class>
14:             org.baykit.xi.ext.opt.servlet.XiServlet
15:         </servlet-class>
16:         <init-param>
17:             <param-name>properties</param-name>
18:             <param-value>WEB-INF/xi.properties</param-value>
19:         </init-param>
20:     </servlet>
21:
22:     <servlet-mapping>
23:         <servlet-name>
24:             Bxi
25:         </servlet-name>
26:         <url-pattern>
27:             *.xi
28:         </url-pattern>
29:     </servlet-mapping>
30:
31: </web-app>
```

以上で最低限必要な準備が整いました。次節から、Xiを拡張するための方法について説明していきます。

図7-1-1 ディレクトリ構成図

```

-xi-ext/
├── WEB-INF/
│   ├── classes/ .....[ classファイルを置くディレクトリ ]
│   └── lib/ .....[ 作成する忍者、タグライブラリのjarファイルを置くディレクトリ ]
├── hello-ninja/ .....[ Hello忍者のディレクトリ ]
│   ├── hello.xi
│   └── org/
│       ├── baykit/
│       │   └── xi/
│       │       ├── ext/
│       │       │   ├── ninja/
│       │       │   │   └── hello/
│       │       │   │       └── Hello.java
│       │       └── timer/
│       │           └── Timer.java
│       └── xi-last-modified/ .....[ 最終更新日取得タグライブラリのディレクトリ ]
│           ├── last-modified.xi
│           └── org/
│               ├── baykit/
│               │   └── xi/
│               │       ├── ext/
│               │       │   └── taglib/
│               │       │       └── last_modified/
│               │       │           └── LastModifiedFunction.java
│               └── opt-ant/ .....[ Antオプションのディレクトリ ]
│                   ├── org/
│                   │   ├── baykit/
│                   │   │   └── xi/
│                   │   │       ├── ext/
│                   │   │       │   └── opt/
│                   │   │       │       └── ant/
│                   │   │       │           └── XiTask.java

```