

## 本日のお品書き

- JSpider の設定ファイルを理解する

- フォルダ構造
- jspider.properties
- plugins.properties
- sites.properties
- [pluginname].properties
- [sitename].properties
- まとめとして何か書け

- Rule

- 前回のおさらい
- Rule の interface
- Rule の実装例
- Rule のデータ構造

- Plugin

- 前回のおさらい
- Plugin の interface
- Plugin の実装例
- Plugin のデータ構造

- おまけ

- ~~¥conf¥~~jspider.properties
- ~~¥conf¥~~plugin.properties
- ~~¥conf¥~~sites.properties
- ~~¥conf¥~~[pluginname].properties
- ~~¥conf¥~~[sitename].properties

- ・ JSpider の設定ファイルを理解する

実装に入る前に、まず、JSpider の設定ファイルを見ていきます。

ユーザは設定ファイルを通じて JSpider にどのような動作をすべきかを伝えます。よって、設定ファイルの仕様を把握することは、JSpider の仕組みを理解することに繋がります。

もちろん、JSpider を hack するのではなく、単に私的なクローラとして使えるようにするためにも、設定ファイルの理解は必要です。

JSpider では、設定をフォルダに格納された複数のテキストファイルとして記述します。また、バッチファイルから JSpider を起動する際に、引数としてどの設定を使うか指定します。

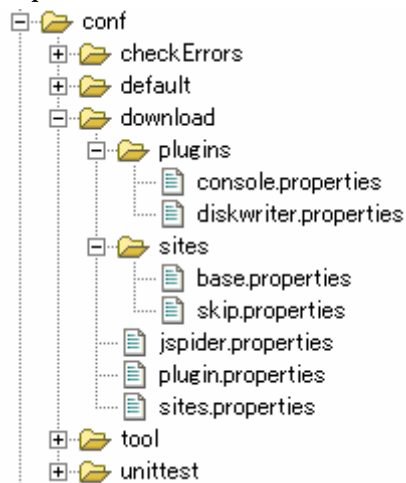
アーカイブを解凍した状態では、¥conf 以下に、"default", "checkErrors", "download", "tool", "unittest" の 5 つの設定が作成されています。たとえば、このなかから "download" を指定して JSpider を動かしたいときには、コマンドプロンプトから、

```
jspider http://www.seman.cs.uec.ac.jp download
```

のようにして実行します。

### フォルダ構造

JSpider の設定のフォルダ構造は次のようになっています。



¥conf 以下に、設定名のフォルダを作ります。その直下に、jspider.properties, plugin.properties, sites.properties の3つの設定ファイルを置きます。また、plugins フォルダ、sites フォルダをそれぞれ作成し、plugin ごと、サイトごとの設定ファイルを置きます。

また、それぞれのファイルは以下のような役割を担っています。

- ・ jspider.properties

JSpider 全般に関わる動作を指定します。また、すべてのサイトに共通する Rule を記述します。

- ・ plugin.properties

利用するプラグインを宣言します。また、Global Event Filter を設定します。個々の plugin についての設定は、plugins フォルダの中の各設定ファイルでなされます。

- ・ sites.properties

サイトごとの設定を宣言します。個々のサイトについての設定は、sites フォルダの中の各設定ファイルでなされます。

- ・ [pluginname].properties

plugin.properties で宣言された個々の plugin について、一つずつ作成されます。各 plugin の動作の詳細を設定します。

- ・ [sitename].properties

sites.properties で宣言された個々のサイトについて、一つずつ作成します。各サイトについて、動作の詳細を設定します。

以下、それぞれの設定ファイルについて、実例を挙げながら、その詳細を見ていきます。

実例としては、¥conf¥download 以下にあるものを使います。このドキュメントの最後を参照してください。

`jspider.properties`

#### ・ Proxy Configuration

プロキシに関する設定。

プロキシを使用する場合には、`jspider.proxy.use` に `true` を設定し、以下の詳細を埋めま  
す。プロキシを使わない場合は、`jspider.proxy.use` に `false` を設定します（それ以下を埋め  
る必要はありません）。

#### ・ Storage Configuration

ダウンロードしたデータを、どのように扱うかを指定します。「どのように扱うか」であ  
って、「どこに保存するか」ではないことに注意してください。ここではダウンロードした  
データを処理する過程の、テンポラリな領域を指定します。

標準的には"`net.javacoding.jspider.core.storage.memory.InMemoryStorageProvider`"が  
使われます。これを指定すると、データはメモリ上で管理されます。

"`net.javacoding.jspider.core.storage.jdbc.JdbcStorageImpl`"を指定すると、データベース  
を使うことができます。使用できるデータベースシステムとしては、マニュアル中では My  
SQL が挙げられていますが、JDBC を使うということから考えると、他のデータベースシ  
ステムでもいけるとおもいます。たぶん。

#### ・ Task Scheduler Configuration

なんかクラス名を指定しているのですが、これが何なのか、いまいち分かりません。ま  
た、困ったことに、マニュアルにはこの項目の説明がありません。

interface のコードを読んだ感じでは、`spider`、`thinker` のそれぞれの task を貯めておく、  
queue のような役割をしているのだと思われます。

次回か次々回で扱う範疇だとも思うので、その際に改めて扱うことにしましょう。

#### ・ Threading Configuration

スレッド周りの設定。

`spiders`、`thinkers` のそれぞれについて、作成するスレッドの数と、モニタリングを行う  
か、行うならそのインターバルを指定します。

`spiders`、`thinkers` については、前回扱った"`micro-tasks`"のアレです。ちゃんと覚えてます  
か？

#### ・ User Agent Configuration

サーバに送信する環境変数"`User-Agent`"を設定します。

#### ・ Logging Configuration

ログをどうやって取るかを指定します。

標準的に使用される位置付けとなっているのが"net.javacoding.jspider.core.logging.impl.CommonsLoggingLogProvider"で、これを指定すると、Jakarta(apache.org)の log4j というライブラリを使ってロギングを行います。

`net.javacoding.jspider.core.logging.Log.java` がロギングを担当するクラスの interface です。これを実装することによって、好きなやり方でロギングを行うことができます。たとえばファイルではなくデータベースにログを吐き出す、とか。

#### ・ Rules Configuration

すべてのサイトに対して適用する Rule をここで指定します。

### plugin.properties

#### ・ General Event Filter Configuration

ここでは、Global Event Filters を設定します。

`jspider.filter.enabled` で Global Event Filters を有効にするかどうかを指定します。また、ここで true を設定した場合には、以下、engine, monitoring, spider の 3 つの Event の分類の各々に対して、適用する Filter を指定します。

#### ・ Plugin Configuration

`jspider.plugin.count` として plugin の数を、それ以下で、plugin の設定ファイルのファイル名を指定します。

ここで指定するのはファイル名にすぎないことに注意してください。plugin のクラス名を指定する必要はありません。また、当然ですが、名称の重複は NG です。

ここで指定されたファイル名で、plugins フォルダの中に設定ファイルを作成して、各 plugin の詳細な設定を行います。

### sites.properties

JSpider ではサイトごとに設定を記述できるわけですが、ここではどのサイトにどの設定ファイルを適用するかを指定します。

`Jspider.site.config.base` として base サイトの設定を、`jspider.site.config.default` にはそれ以外のサイトにデフォルトで適用される設定を指定します。

また、たとえば

```
www.seman.cs.uec.ac.jp=onai-lab
```

のようにして、直接サイト名を指定することも可能です。また、同じ設定ファイルを複数の行で指定することもできます。

```
www.seman.cs.uec.ac.jp=onai-lab  
ace.seman.cs.uec.ac.jp=onai-lab
```

ここで指定されたファイル名で、sites フォルダの中に設定ファイルを作成して、各サイトの詳細な設定を行います。

[pluginname].properties

plugin.properties で指定した各 plugin について、詳細な設定をおこないます。

plugin.class として plugin のクラス名を。

plugin.filter.enabled でその plugin に関連づけられた Event Filters を有効にするかどうかを指定します。true を設定した場合には、以下、engine, monitoring, spider の3つの Event の分類の各々に対して、適用する Filter を指定します。

plugin.config.\*には、使用する plugin ごとの設定をします。ここで設定する項目は、plugin によって異なります。

[sitename].properties

- site.handle

site.handle に、そのサイトを処理対象とするかどうかを指定します。

ここに false を設定すると、そのサイトは完全に無視されます。

- Proxy Configuration

jspider.properties で設定したプロキシの設定を有効にするかどうかを指定します。

わざわざこんな項目が設けてあるのは、LAN 内の (=ローカル IP アドレスが振られた)

Web サーバにあるサイトに対してはプロキシを切りたいからだとおもわれます。

#### ・ Throttling Configuration

そのサイトからどのようなスケジューリングでリソースをダウンロードするかを指定できます。

Throttle interface(`net.javacoding.jspider.core.throttle.Throttle.java`)を実装したクラスを指定します。

標準的には`net.javacoding.jspider.core.throttle.impl.DistributedLoadThrottleProvider`が使われます。ほとんどのケースでこれを指定しておけば良いとおもいます。マニュアルによると、あたかも人がアクセスしているかのように見せかける Throttle interface の実装も用意されてるそうです。

`net.javacoding.jspider.core.throttle.impl.DistributedLoadThrottleProvider`を指定した場合には、続いて `site.throttle.config.interval` で Web サーバにアクセスする最低の間隔を指定します。単位は ms です。

#### ・ Cookie Configuration

Cookie を有効にするかどうかを指定します。

#### ・ Robots.txt configuration

robots.txt を解釈するか、またそれに従うかどうかを設定できます。この設定によっては、robots.txt を完全に無視してクローリングさせることもできるっぽいです。

マニュアルには "Please don't use the robots.txt settings" とか何とか書かれているのみで、詳しい設定の仕方は説明されていません。

#### ・ User Agent Configuration

そのサイトに送信する "User-Agent" を指定できます。これを指定した場合、`jspider.properties` で指定したものは上書きされます。

#### ・ Rules Configuration

そのサイトに対する Rule を設定します。`jspider.properties` で設定したものと、両方とも有効な Rule として使われます。上書きはされません。御注意。

まとめとして

.....とまあ、ひたすら箇条書きとなってしまいましたが、JSpider の設定ファイルに関する説明は以上です。

JSpider が非常に柔軟性の高い構造になっていることが、設定ファイルの構成からも分かるとおもいます。可換性を持たせるべき箇所はすべて interface を継承して実装するという構造になっており、自分でそのようなクラスを実装することで、容易に処理の内容を切り替えることができます。また、Plugin と Event によるイベント・ドリブン・モデルを採用したことによって、特定の状況で呼び出されるコードを簡単に組み込めるようになっています。

設定項目はこれがすべてではありません。設定ファイルの詳細な仕様については、実際にマニュアルを参照してください。また、設定ファイルはこれがすべてではありません。"`¥common¥conf¥logging`"で、log4j の設定をするようになっています。こちらは必要になったらマニュアルを参照すれば良いとおもいます。基本的にこれをいじる必要が生じることはないでしょう。

- Rule

### 前回のおさらい

Rule はあるリソースをダウンロードすべきか、あるいは処理すべきかを決定するための仕組みです。

Rules は渡された URL を判断し、以下のいずれかの指示を返します。

- don't care

リソースを完全に無視する。

- accept

状態のチェックをし、リソースをダウンロードし、HTML ならパースしてリンクを抽出する。他によって上書きされる。

- ignore

状態のチェックのみをし、ダウンロードやパースは行わない。

- forbidden

状態のチェックのみをし、ダウンロードやパースは行わない。ignore より強い。

ところでソースを読んでみても、ignore と forbidden の違いがよく分かりませんでした。

Rule は複数個設定できます。また、その場合、URL はすべての Rule に対してチェックされます。Rule の返す指示には強弱の関係があり、選択された指示のなかでもっとも強いものを返します。

### Rule の interface

Rule の interface は `net.javacoding.spider.spi.Rule.java` です。

以下のようなものになっています。

### Rule.java

```
/**
 * Interface to be implemented upon each class that will act as a Decision-
 * taking class.
 * A group of Rule objects will together decide whether a certain URL is
 * eligible for spidering and/or parsing in a given context.
 *
 * $Id: Rule.java,v 1.1 2003/04/03 16:25:22 vanrogu Exp $
 *
 * @author G?nther Van Roey
```

```
*/
public interface Rule {

    /**
     * Returns the name of the rule.
     * @return the name of the rule
     */
    public String getName ( );

    /**
     * Applies the rule to the given URL.
     * @param context the context we're spidering in
     * @param currentSite the site we're currently spidering
     * @param url the URL to be evaluated
     * @return Decision object telling whether the URL is accepted for a
     * specific purpose.
     */
    public Decision apply(SpiderContext context, Site currentSite, URL url);
}
}
```

ここでキモとなるのは

```
Public Decision apply(SpiderContext context, Site currentSite, URL url);
```

です。Rule は SpiderContext, Site, URL のインスタンスをそれぞれ渡され、Decision のインスタンスとして、その判断結果を返します。

SpiderContext, Site, URL は、それぞれ JSpider のコンテキスト、処理対象のサイト、そして今まさに判断しようとしている URL に対応するオブジェクトです。

Rule は引数として渡されたこれらのオブジェクトから必要な情報を取得し、判断を行います。

SpiderContext は `net.javacoding.jspider.core.SpiderContext.java`、Site は `net.javacoding.jspider.api.model.Site.java` がそれぞれの interface となっています。

URL は `java.net.URL` です。

また、戻り値となる Decision (`net.javacoding.jspider.api.model.Decision.java`) ですが、ご多分に漏れずこれも interface です。

インスタンスを作る際には、Decision を継承した DecisionInternal (`net.javacoding.jspider.core.model.DecisionInternal.java`) が使われます。

実質的には、判断の結果を表現する列挙型として扱われます。定数を宣言している箇所を抜き出します。

### Decision.java

```
public static final int RULE_DONTCARE = 0;
public static final int RULE_ACCEPT = 1;
public static final int RULE_IGNORE = 2;
public static final int RULE_FORBIDDEN = 3;
```

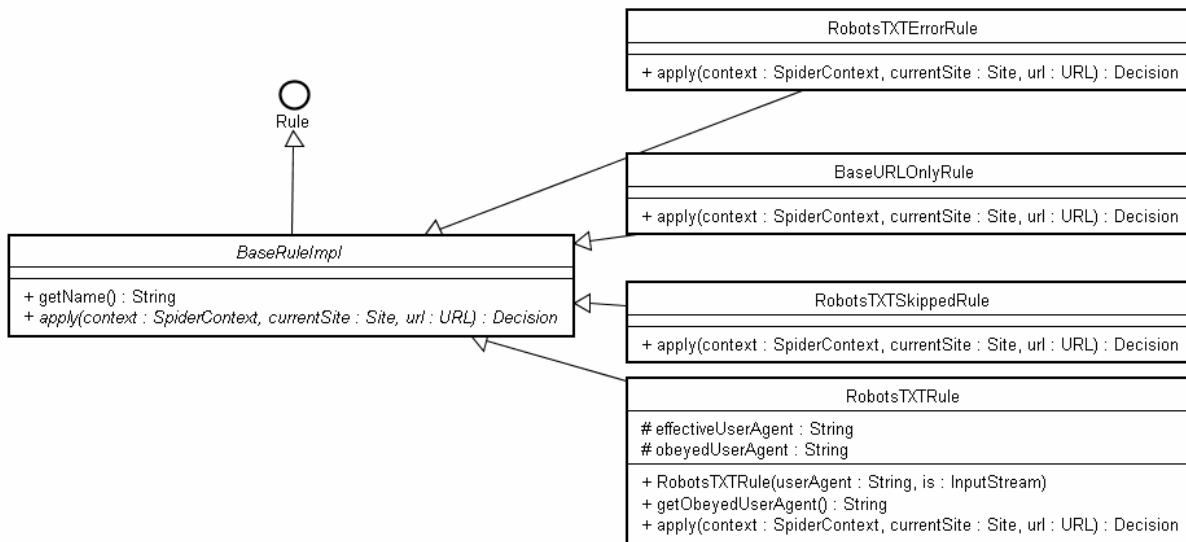
数の大きい方が強いという関係になっています。

この3つのオブジェクトからどのような情報が取れるかは、いちいち扱いません。各自コードを読んでください。

設定した内容はおそらくすべて取れますし、状況に依存した情報も割と取れるっぽいですが、ただ、インターフェースとしてあるからといって、そこにちゃんとした情報が格納されているとは限りませんが。

### Rule の実装例

Rule のUML を下に示します。Rule interface を継承した BaseRuleImpl という抽象クラスがあり、さらにこれを継承して個々の Rule を実装するというお作法になっています。Rule によっては独自の属性や関数を持ちます。UML には JSpider に実装されている Rule の中から、4つの実装を適当に抜き出して示しました。



実際に、すでに実装されている Rule からいくつかの apply メソッドを抜き出してみます。Rule の実装は以下のフォルダにあります。説明の便宜上、UML に載せたのと別のものを持ってきています。

```
¥net¥javacoding¥jspider¥mod¥rule¥
```

```
¥net¥javacoding¥jspider¥core¥rule¥impl¥
```

### AcceptAllRule.java

```
public Decision apply(SpiderContext context, Site currentSite, URL url) {
    return new DecisionInternal(Decision.RULE_ACCEPT, "accept all rule - so resource is
accepted");
}
```

### BaseSiteOnlyRule.java

```
public Decision apply(SpiderContext context, Site currentSite, URL url) {
    if (context.getBaseURL().getHost().equalsIgnoreCase(url.getHost())) {
        return new DecisionInternal(Decision.RULE_ACCEPT, "url accepted");
    } else {
        return new DecisionInternal(Decision.RULE_IGNORE, "url ignored because it points
to an external site");
    }
}
```

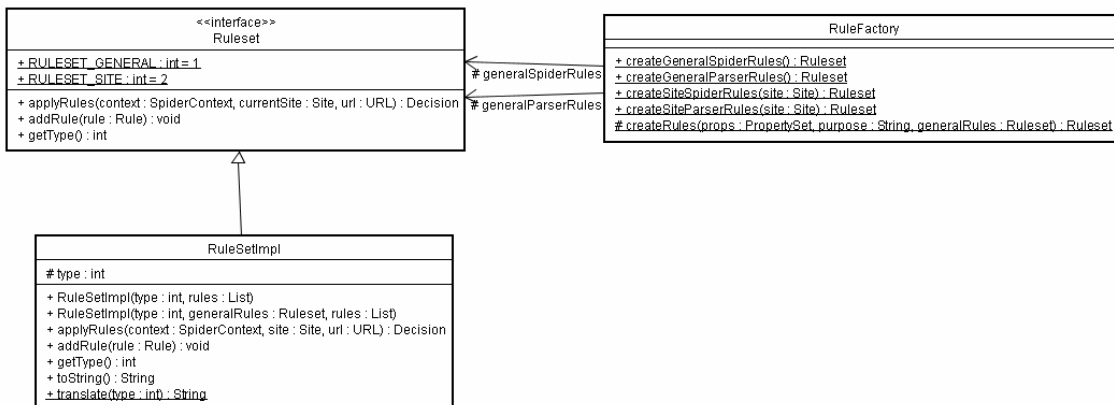
### TextHtmlMimeTypeOnlyRule.java

```
public Decision apply(SpiderContext context, Site currentSite, URL url) {
    FetchedResource resource =
(FetchedResource)context.getStorage().getResourceDAO().getResource(url);
    String mime = resource.getMime();
    Decision decision = new DecisionInternal(Decision.RULE_IGNORE, "mimetype is '" + mime
+ "' - resource ignored");

    if ( mime == null ) {
        decision = new DecisionInternal(Decision.RULE_ACCEPT, "mimetype is null - defaulted
to text/html - accepted" );
    } else if (mime.toLowerCase().indexOf("text/html") > -1) { // can also contain charset
info
        decision = new DecisionInternal(Decision.RULE_ACCEPT, "mimetype is '" + mime + "'
- resource accepted");
    }

    return decision;
}
```

## Rule のデータ構造



RuleはRuleSet interface(¥net¥javacoding¥jspider¥core¥rule¥RuleSet.java)を実装したクラスで管理されます。

### RuleSet.java

```

/**
 * Interface of a RuleSet, a set of Rules that will be applied as a group
 * on URLs.
 *
 * $Id: RuleSet.java,v 1.8 2003/04/03 16:24:53 vanrogu Exp $
 *
 * @author G·ther Van Roey
 */
public interface RuleSet {

    /** A general ruleset applied to all urls in the system. */
    public static final int RULESET_GENERAL = 1;

    /** A site-specific ruleset, only applied to urls of that site. */
    public static final int RULESET_SITE = 2;

    /**
     * Applies the ruleset to the given URL.
     * @param context context we're spidering under
     * @param url the url to decide on
     * @param currentSite the site currently being spidered
     * @return Decision object expressing the ruleset's decision about the
     * given url in the given context.
     */
    public Decision applyRules(SpiderContext context, Site currentSite, URL url);

    /**
     * Adds a rule to the ruleset.
     * @param rule rule to be added
     */
    public void addRule(Rule rule);

    /**
     * Returns the type of the ruleset - GENERAL or SITE.
     * @return type of the ruleset
     */
    public int getType ( );
}
    
```

複数の Rule interface のインスタンスを管理するコレクションの役割を負います。  
また、

```
public Decision applyRules(SpiderContext context, Site currentSite, URL url);
```

で、管理する Rule のインスタンスに対して判断をさせ、その結果を統合し、Ruleset としての判断結果を返します。

これを実装したのが RuleSetImpl(¥net¥javacoding¥jspider¥core¥rule¥impl¥RuleSetImpl.java)です。  
applyRules メソッドのみを抜き出して見ましょう。

#### RuleSetImpl.java

```
public Decision applyRules(SpiderContext context, Site site, URL url) {
    Decision decision = null;

    if (generalRules != null) {
        decision = generalRules.applyRules(context, site, url);
    } else {
        decision = new DecisionInternal();
    }

    if (decision.isVetoable()) {

        Rule[] rules = (Rule[]) localRules.toArray(new Rule[localRules.size()]);
        for (int i = 0; i < rules.length; i++) {
            Rule rule = rules[i];
            Decision lastDecision = rule.apply(context, site, url);
            decision.addStep(rule.getName(), type, lastDecision.getDecision(),
lastDecision.getComment() );
            decision.merge(lastDecision);

            if (!lastDecision.isVetoable()) {
                break;
            }
        }

        decision.addStep("Ruleset",type, decision.getDecision(), "ruleset final decision");

        return decision;
    }
}
```

自分の管理する Rule に対して、次々と apply を呼び出し、その戻り値をマージ(2つの Decision を比べて、より強いものを残す処理)しています。

RuleSetImpl は RuleFactory(¥net¥javacoding¥jspider¥core¥rule¥RuleFactory.java)

の中で使用されます。

### RuleFactory.java

```
/**
 *
 * $Id: RuleFactory.java,v 1.12 2003/04/29 17:53:48 vanrogu Exp $
 *
 * @author G·ther Van Roey
 */
public class RuleFactory {

    protected static Ruleset generalSpiderRules;
    protected static Ruleset generalParserRules;

    public static synchronized Ruleset createGeneralSpiderRules() {
        if (generalSpiderRules == null) {
            PropertySet props =
ConfigurationFactory.getConfiguration().getJSpiderConfiguration();
            PropertySet jspiderProps = new MappedPropertySet ( ConfigConstants.JSPIDER,
props );
            generalSpiderRules = createRules(jspiderProps, "spider", null);
        }
        return generalSpiderRules;
    }

    public static synchronized Ruleset createGeneralParserRules() {
        if (generalParserRules == null) {
            PropertySet props =
ConfigurationFactory.getConfiguration().getJSpiderConfiguration();
            PropertySet jspiderProps = new MappedPropertySet ( ConfigConstants.JSPIDER,
props );
            generalParserRules = createRules(jspiderProps, "parser", null);
        }
        return generalParserRules;
    }

    public static Ruleset createSiteSpiderRules(Site site) {
        PropertySet props =
ConfigurationFactory.getConfiguration().getSiteConfiguration(site);
        PropertySet siteProps = new MappedPropertySet ( ConfigConstants.SITE, props );
        return createRules(siteProps, "spider", generalSpiderRules);
    }

    public static Ruleset createSiteParserRules(Site site) {
        PropertySet props =
ConfigurationFactory.getConfiguration().getSiteConfiguration(site);
        PropertySet siteProps = new MappedPropertySet ( ConfigConstants.SITE, props );
        return createRules(siteProps, "parser", generalParserRules);
    }

    protected static Ruleset createRules(PropertySet props, String purpose, Ruleset
generalRules) {
        PropertySet rulesProps = new MappedPropertySet(ConfigConstants.RULES, props);

        int rulesCount = rulesProps.getInteger(purpose + "." + ConfigConstants.RULES_COUNT, 0);

        Log log = LogFactory.getLog(RuleFactory.class);

        List rules = new ArrayList();

        for (int i = 0; i < rulesCount; i++) {
            String prefix = purpose + "." + (i + 1);
            PropertySet ruleProps = new MappedPropertySet(prefix, rulesProps);
```

```
Class ruleClass = ruleProps.getClass(ConfigConstants.RULE_CLASS, null);
Rule rule = null;

if (ruleClass != null) {

    try {
        Class[] paramTypes = new Class[1];
        paramTypes[0] = PropertySet.class;

        Object params[] = new Object[1];
        params[0] = new MappedPropertySet ( ConfigConstants.RULE_CONFIG,
ruleProps);

        Constructor constructor = ruleClass.getDeclaredConstructor(paramTypes);
        rule = (Rule) constructor.newInstance(params);

    } catch (NoSuchMethodException e) {
        log.debug("rule " + ruleClass.getName() + " hasn't got a config-param
constructor");
    } catch (SecurityException e) {
        log.info("SecurityException finding constructor");
    } catch (InstantiationException e) {
        log.info("InstantiationException finding constructor");
    } catch (IllegalAccessException e) {
        log.info("IllegalAccessException finding constructor");
    } catch (InvocationTargetException e) {
        log.info("InvocationTargetException finding constructor");
    }
}

if (rule == null) {
    try {
        rule = (Rule) ruleClass.newInstance();
    } catch (InstantiationException e) {
        log.error("InstantiationException on Rule", e);
    } catch (IllegalAccessException e) {
        log.error("IllegalAccessException on Rule instantiation", e);
    }
}

if (rule == null) {
    log.error("rule couldn't be instantiated");
} else {
    log.debug ( "added rule " + rule.getClass().getName() + " to " + purpose
+ " ruleset");
    rules.add(rule);
}
} else {
    log.error("cannot find rule class '" +
ruleProps.getString(ConfigConstants.RULE_CLASS, "<unknown>") + "' for " + purpose + " rules");
}
}

if (generalRules == null) {
    return new RuleSetImpl(Ruleset.RULESET_GENERAL, rules);
} else {
    return new RuleSetImpl(Ruleset.RULESET_SITE, generalRules, rules);
}
}
}
```

RuleFactory は SpiderContext interface を実装した SpiderContextImpl クラス(¥net¥javacoding¥jspider¥core¥impl¥SpiderContextImpl.java)のなかで使われます。

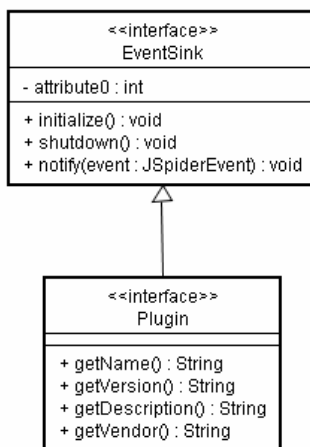
- Plugin

## 前回のおさらい

Plugin ってのは、各種のイベントに対応する処理を実行するための仕組みです。  
次のような Plugin があります。

- レポートを作成する
- コンソールにメッセージを出力する
- リソースをストレージに書き出す
- メールを送信する
- etc...

## Plugin の interface



Plugin の interface は `net.javacoding.jspider.spi.Plugin.java` です。

## Plugin.java

```
/**
 * Interface that must be implemented by a JSpider module. This interface
 * allows interaction between the JSpider core and the module.
 *
 * $Id: Plugin.java,v 1.1 2003/04/03 16:25:22 vanrogu Exp $
 *
 * @author G·ther Van Roey
 */
```

```
public interface Plugin extends EventSink {

    /**
     * Returns the name of the module.
     * @return name of the module
     */
    public String getName();

    /**
     * Returns the version string of the module.
     * @return version string of the module
     */
    public String getVersion();

    /**
     * Returns a description of the module.
     * @return description of the module
     */
    public String getDescription();

    /**
     * Returns the vednor of the module.
     * @return the vendor of the module
     */
    public String getVendor();
}
```

ですが、これ自体は名前やバージョンといった情報のインターフェースを宣言しているだけです。

Plugin としての実質的な処理は、継承元の EventSink interface のなかで宣言されています。

¥net¥javacoding¥jspider¥api¥event¥EventSink.java を見てみましょう。

### EventSink.java

```
/**
 * Interface that will be implemented upon each class that will be used as an
 * event sink for JSpider API events.
 *
 * $Id: EventSink.java,v 1.3 2003/02/27 16:47:31 vanrogu Exp $
 *
 * @author G·ther Van Roey
 */
public interface EventSink {

    public void initialize ( );

    public void shutdown ( );

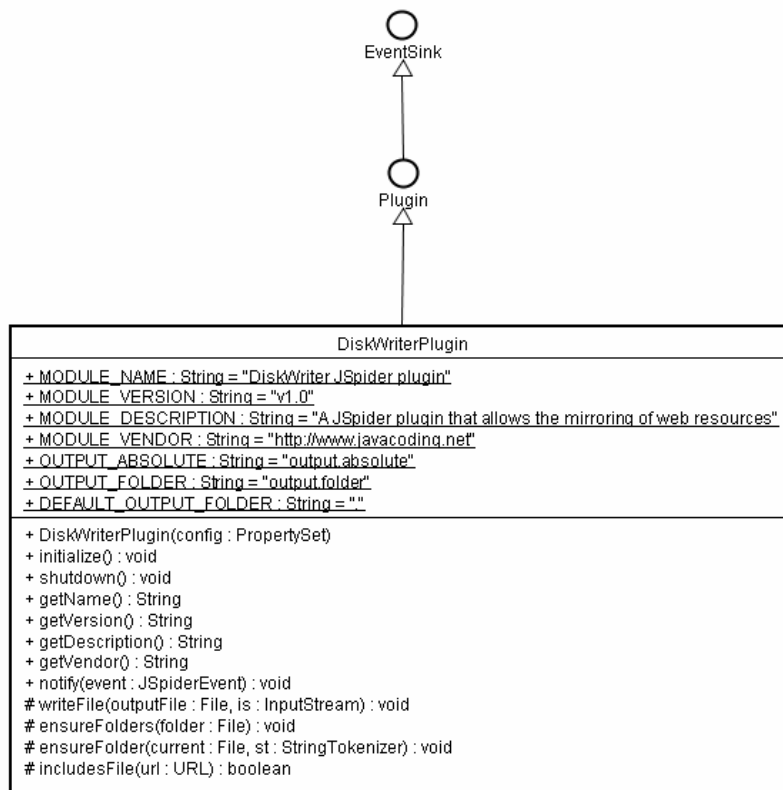
    /**
     * Notifies the event sink of a new event.
     * @param event the event that occurred
     */
    public void notify(JSpiderEvent event);
}
```

これが Plugin としての動作となります。

initialize で初期化を、shutdown で終了処理を。また、Plugin に Event が到達するたびに notify が呼び出されます。

戻り値が void となっていることから分かるとおり、Core( & Dispatcher)は Event を一方的に通知するのみで、そのフィードバックを受け取ることはしていません。

### Plugin の実装例



すでに実装されている Plugin からいくつかの apply メソッドを抜き出してみます。  
Plugin の実装は `net¥javacoding¥jspider¥mod¥plugin¥` 以下にあります。

DiskWriterPlugin.java

```
public void initialize() {
}

public void shutdown() {
}

public void notify(JSpiderEvent event) {
    if (event instanceof ResourceFetchedEvent) {
        ResourceFetchedEvent rfe = (ResourceFetchedEvent) event;
        URL url = rfe.getURL();
        String path = url.getHost() + url.getPath();
        if (includesFile(url)) {
            path = url.getHost() + url.getFile();
        } else {
            path = url.getHost() + url.getPath() + "/index.html";
        }
        File outputFile = new File(outputFolder, path);
        ensureFolders(outputFile);
        log.debug("Writing " + outputFile);
        writeFile(outputFile, rfe.getResource().getInputStream());
        log.info("Wrote " + outputFile);
    }
}
```

## Plugin のデータ構造

設定された Plugin は、PluginFactory クラス(¥net¥javacoding¥jspider¥core¥impl¥PluginFactory.java)でインスタンスングされます。

## PluginFactory.java

```
/**
 *
 * $Id: PluginFactory.java,v 1.9 2003/04/22 16:43:34 vanrogu Exp $
 *
 * @author G?nther Van Roey
 */
public class PluginFactory {

    PluginInstantiator pluginInstantiator;

    public PluginFactory() {
        pluginInstantiator = new PluginInstantiator();
    }

    public Plugin[] createPlugins() {

        Log log = LogFactory.getLog(PluginFactory.class);
        ArrayList loadedPlugins = new ArrayList();

        PropertySet props = ConfigurationFactory.getConfiguration().getPluginsConfiguration();
        PropertySet pluginsProps = new MappedPropertySet(ConfigConstants.PLUGIN_S,props);
```

```
int pluginCount = pluginsProps.getInteger(ConfigConstants.PLUGINS_COUNT, 0);
log.info("Loading " + pluginCount + " plugins.");

for (int i = 0; i < pluginCount; i++) {
    String pluginInstance = pluginsProps.getString( "" + (i+1) + "." +
ConfigConstants.PLUGINS_CONFIG, null);
    if (pluginInstance != null) {
        log.info("Loading plugin configuration '" + pluginInstance + "
'...");
        PropertySet config = ConfigurationFactory.getConfiguration().getPluginConfiguration(pluginInstance);
        PropertySet pluginConfig = new MappedPropertySet(ConfigConstants.PLUGIN, config);
        Class pluginClass = pluginConfig.getClass(ConfigConstants.PLUGIN_CLASS, null);
        if (pluginClass == null) {
            log.info("Plugin class '" + pluginConfig.getString(ConfigConstants.PLUGIN_CLASS, "") + "' not found");
        } else {
            PropertySet pluginParams = new MappedPropertySet(ConfigConstants.PLUGIN_CONFIG, pluginConfig);
            Plugin plugin = pluginInstantiator.instantiate(pluginClass, pluginInstance, pluginParams);

            PropertySet filterConfig = new MappedPropertySet(ConfigConstants.PLUGIN_FILTER, pluginConfig);
            if (filterConfig.getBoolean(ConfigConstants.PLUGIN_FILTER_ENABLED, false)) {

                log.info("Plugin uses local event filtering");
                loadedPlugins.add(new PluginSocket(plugin, filterConfig));
            } else {
                log.info("Plugin not configured for local event filtering");
                loadedPlugins.add(plugin);
            }

            log.info("Plugin Name      : " + plugin.getName());
            log.info("Plugin Version : " + plugin.getVersion());
            log.info("Plugin Vendor  : " + plugin.getVendor());
        }
    } else {
        log.info("Plugin configuration '" + pluginInstance + "' couldn't be loaded");
    }
}

log.info("Loaded " + loadedPlugins.size() + " plugins.");
return (Plugin[]) loadedPlugins.toArray(new Plugin[loadedPlugins.size()]);
}
```

Plugins メソッドを呼び出すと PluginFactory は設定を参照し、指定された Plugin のインスタンスを作成してその配列を返します。

Plugin のインスタンスングは PluginInstantiator クラス(¥net¥javacoding¥jspider¥core¥impl¥PluginInstantiator.java)が行います。

設定されたクラス名から getDeclaredConstructor してます。JAVA って便利ですね。

PluginFactory は SpiderContextFactory クラス(`net.javacoding.jspider.core.SpiderContextFactory.java`)の中で使われます。

- ・ おまけ

### jspider.properties

```
# -----  
# JSpider Main Configuration File for config 'download'  
# -----  
#  
# $Id: jspider.properties,v 1.9 2003/04/03 16:10:31 vanrogu Exp $  
#  
# -----  
  
# -----  
# Proxy Configuration  
# -----  
  
jspider.proxy.use=false  
jspider.proxy.host=  
jspider.proxy.port=  
jspider.proxy.authenticate=false  
jspider.proxy.user=  
jspider.proxy.password=  
  
# -----  
# Storage Configuration  
# -----  
  
jspider.storage.provider=net.javacoding.jspider.core.storage.memory.InMemoryStorageProvider  
  
# -----  
# Task Scheduler Configuration  
# -----  
  
jspider.taskscheduler.provider=net.javacoding.jspider.core.task.impl.DefaultSchedulerProvider  
jspider.taskscheduler.monitoring.enabled=true  
jspider.taskscheduler.monitoring.interval=1000  
  
# -----  
# Threading Configuration  
# -----  
  
jspider.threads.spiders.count=5  
jspider.threads.spiders.monitoring.enabled=true  
jspider.threads.spiders.monitoring.interval=1000  
jspider.threads.thinkers.count=1  
jspider.threads.thinkers.monitoring.enabled=true  
jspider.threads.thinkers.monitoring.interval=1000  
  
# -----  
# User Agent Configuration  
# -----  
  
#jspider.userAgent=JSpider (http://j-spider.sourceforge.net)  
  
# -----  
# Logging Configuration  
# -----  
  
jspider.log.provider=net.javacoding.jspider.core.logging.impl.CommonsLoggingLogProvider  
  
# -----
```

```
# Rules Configuration
# -----
jspider.rules.spider.count=1
jspider.rules.spider.1.class=net.javacoding.jspider.mod.rule.OnlyHttpProtocolRule

jspider.rules.parser.count=1
jspider.rules.parser.1.class=net.javacoding.jspider.mod.rule.TextHtmlMimeTypeOnly
Rule
```

### Plugin.properties

```
# -----  
# Plugin configuration for config 'download'  
# -----  
#  
# $Id: plugin.properties,v 1.3 2003/04/22 16:43:33 vanrogu Exp $  
# -----  
# -----  
# General Event Filter Configuration  
# -----  
  
jspider.filter.enabled=true  
jspider.filter.engine=net.javacoding.jspider.mod.eventfilter.AllowAllEventFilter  
jspider.filter.monitoring=net.javacoding.jspider.mod.eventfilter.AllowAllEventFilter  
jspider.filter.spider=net.javacoding.jspider.mod.eventfilter.AllowAllEventFilter  
  
# -----  
# Plugin Configuration  
# -----  
  
jspider.plugin.count=2  
jspider.plugin.1.config=console  
jspider.plugin.2.config=diskwriter
```

### Sites.properties

```
# -----  
---  
# Websites configuration file  
# -----  
---  
#  
# $Id: sites.properties,v 1.4 2003/04/25 21:28:55 vanrogu Exp $  
#  
# -----  
---  
  
jspider.site.config.base=base  
jspider.site.config.default=skip
```

**Plugins¥console.properties**

```
# -----  
-  
# Console Output Plugin Configuration File  
# -----  
-  
#  
# $Id: console.properties,v 1.3 2003/04/02 20:54:56 vanrogu Exp $  
#  
# -----  
-  
  
plugin.class=net.javacoding.jspider.mod.plugin.console.ConsolePlugin  
  
plugin.filter.enabled=true  
  
plugin.filter.engine=net.javacoding.jspider.mod.eventfilter.AllowAllEventFilter  
plugin.filter.monitoring=net.javacoding.jspider.mod.eventfilter.AllowAllEventFilter  
plugin.filter.spider=net.javacoding.jspider.mod.eventfilter.ErrorsOnlyEventFilter  
  
plugin.config.prefix=[Plugin]  
plugin.config.addspace=true
```

## Sites¥base.properties

```
# -----  
-  
# Base Site Configuration File  
# -----  
-  
#  
# $Id: base.properties,v 1.1 2003/04/25 21:28:57 vanrogu Exp $  
#  
# -----  
-  
site.handle=true  
  
# -----  
-  
# Proxy Configuration  
# -----  
-  
site.proxy.use=true  
  
# -----  
-  
# Throttling Configuration  
# -----  
-  
site.throttle.provider=net.javacoding.jspider.core.throttle.impl.DistributedLo  
adThrottleProvider  
site.throttle.config.interval=1000  
  
# -----  
-  
# Cookie Configuration  
# -----  
-  
site.cookies.use=true  
  
# -----  
-  
# Robots.txt configuration  
# -----  
-  
site.robotstxt.fetch=false  
site.robotstxt.obey=false  
  
# -----  
-  
# User Agent Configuration  
# -----  
-  
site.userAgent=JSpider (http://j-spider.sourceforge.net)  
  
# -----  
-  
# Rules Configuration  
# -----  
-  
site.rules.spider.count=0
```

```
site.rules.parser.count=0
```