Maya – JavaServer Templates –

chapter.1 ServiceProvider

ServiceProvider は、カスタマイズの集中する機能ブロックです。ErrorHandler、InjectionResolver、ExpressionResolver をカスタマイズします。また、EngineSetting によってエンジンからアクセスされる設定パラメータ群を管理します。

1.1. web.xml

Maya は Servlet としてエンジンを提供しています。サーブレットコンテナにインストールするには、WEB アプリケーションディスクリプタ(¥WEB-INF¥web.xml)に MayaServlet を登録します。

List. 1-1 / WEB-INF/web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <servlet>
        <servlet-name>Maya</servlet-name>
        <servlet-class>org.seasar.maya.impl.MayaServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Maya</servlet-name>
        <url-pattern>*.html</url-pattern>
    </servlet-mapping>
</web-app>
```

List. 1-1に例示したweb.xmlでは、Mayaサーブレット(org.seasar.maya.MayaServlet)をURLパターン*.htmlにディスパッチしています。MayaServletは、たとえば「/maya-pages/*」のような特定フォルダに対してディスパッチすることも可能です。

1.2. エンジン設定

Maya のエンジン設定は、maya.conf という名前の設定 XML ファイルに行います。maya.conf は /WEB-INF/フォルダ直下に配置されたものを読み出します。ファイルが配置されていない場合は、各設定項目について、デフォルトの値が適用されます。

1.2.1 maya.conf

maya.confのDTDはList. 1-2の通りです。

List. 1-2 maya-conf_1_0.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!NOTATION maya.conf_1_0 PUBLIC
        "-//The Seasar Foundation//DTD Maya Config 1.0//EN">

<!ELEMENT service (description?, source?, engine?,
            expression?, specificationBuilder?, templateBuilder?) >

<!ELEMENT description (#PCDATA)>
```

```
<!ELEMENT source (description?, entry*, parameter*)>
```

```
<!ELEMENT entry (description?, parameter*)>
<!ATTLIST entry class CDATA #REQUIRED>

<!ELEMENT engine (description?, pageContext?,
           specification?, errorHandler?, parameter*)>
<!ATTLIST engine checkTimestamp (true | false) "true">
<!ATTLIST engine outputWhitespace (true | false) "true">
<!ATTLIST engine suffixSeparator CDATA #IMPLIED>
<!ATTLIST engine reportUnresolvedID (true | false) "true">

<!ELEMENT pageContext (description?)>
<!ATTLIST pageContext errorPageURL CDATA #IMPLIED>
<!ATTLIST pageContext bufferSize CDATA #IMPLIED>
<!ATTLIST pageContext needSession (true | false) "true">
<!ATTLIST pageContext autoFlush (true | false) "true">

<!ELEMENT specification (description?)>
<!ATTLIST specification path CDATA #IMPLIED>

<!ELEMENT errorHandler (description?, parameter*)>
<!ATTLIST errorHandler class CDATA #REQUIRED>

<!ELEMENT expression (description?, resolver*)>

<!ELEMENT resolver (description?, parameter*)>
<!ATTLIST resolver class CDATA #REQUIRED>

<!ELEMENT specificationBuilder (description?, parameter*)>

<!ELEMENT templateBuilder (description?, resolver*, library?, parameter*)>

<!ELEMENT library (description?, scanner*, parameter*)>

<!ELEMENT scanner (description?, parameter*)>
<!ATTLIST scanner class CDATA #REQUIRED>

<!ELEMENT parameter (description?)>
<!ATTLIST parameter name CDATA #REQUIRED>
<!ATTLIST parameter value CDATA #REQUIRED>
```

サンプルはList. 1-3の通りです。

List. 1-3　maya.conf

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE service
    PUBLIC "-//The Seasar Foundation//DTD Maya Config 1.0//EN"
    "http://maya.seasar.org/dtd/maya-conf_1_0.dtd">
<service>
  <engine>
    <specification path="/WEB-INF/default.maya"/>
    <errorHandler
        class="org.seasar.maya.standard.engine.error.TemplateErrorHandler">
      <parameter name="errorTemplateRoot" value="/WEB-INF/error"/>
    </errorHandler>
  </engine>
  <templateBuilder>
```

```
    <resolver
   class="org.seasar.maya.standard.builder.specification.DirectInjectionResolver"/>
    <library>
      <scanner
       class="org.seasar.maya.standard.builder.library.scanner.TLDLibraryScanner"/>
    </library>
  </templateBuilder>
</service>
```

1.2.2 source

source ノードは、Maya 内部で用いるリソースローダーのカスタマイズに用いられます。子の entry ノードには、必須の class 属性に org.seasar.maya.source.factory.SourceEntry を実装したクラスの完全修飾名を記述します。

1.2.3 engine

engine ノードは、多くの属性を持ちます。属性は Maya エンジンの挙動を設定します。特に specification 属性は重要で、デフォルト設定 XML のパスを記述します。spacification のほかにも以下の設定項目があります。

- checkTimestamp テンプレートファイルの更新時に再ビルドするチェックを行う。
- outputWhitespace テンプレート上の Ignorable な空白を出力する。
- suffixSeparator テンプレートファイルのページ名と接尾辞の区切り文字。デフォルト「$」。
- reportUnresolvedID テンプレート上の id 属性が Maya ノードで該当無い場合、例外。

子ノードの pageContext は、Maya エンジン中でリクエスト毎に取得する JSP の PageContext オブジェクトの初期化パラメータ群です。内部で JspFactory#getPageContext()メソッドの引数として渡されます。

子ノードの specification は、デフォルトの設定 XML を指定します。必須の path 属性には、.maya ファイルへのパス文字列を設定します。

子ノードの errorHandler は、例外発生時のエラーハンドラをエンジンに追加します。記述順に追加され、例外発生時には先に追加されたハンドラから順に処理されていきます。errorHandler ノードには必須の class 属性があります。この属性値には org.seasar.maya.engine.error.ErrorHandler インターフェイスを実装したクラスの Java 完全修飾名のみが記述できます。

1.2.4 expression

expression ノードは、式言語処理エンジンのトークン毎の処理を行うレゾルバを追加するものです。子ノードの resolver は、必須の class 属性に org.seasar.maya.el.resolver.ExpressionResolver インターフェイスの実装クラス完全修飾名を記述します。記述順に式言語処理エンジンに追加され、実行時には先に追加したレゾルバから順に利用されます。

1.2.5 templateBuilder

templateBuilder ノードは、テンプレートビルド時のノード処理を行うレゾルバを追加するものです。子ノードの resolver は、必須の class 属性に org.seasar.maya.builder.specification.InjectionResolver インターフェイスを実装したクラスの Java 完全修飾名を記述します。expression と同じく、ビルダーは記述順により利用します。

このlibraryノードは、さらにその子ノードにscannerノードを保持し、MLDマネージャの設定を行います。scannerノードの必須のclass属性には、org.seasar.maya.builder.library.scanner.LibraryScannerを実装したクラスを指定します。List. 1-3では、JSPのTLDをパースする、TLDLibraryScannerを追加指定しています。

1.2.6 parameter

engine、errorHandler、resolver の各ノードは、parameter ノードを子に持つことができます。parameter ノードはユーザー定義のカスタムパラメータを引き渡すためのもので、必須の name 属性および必須の value 属性に記述された値を、引き渡します。name 属性が Java プロパティのプロパティ名で、value 属性がその設定値となります。

1.3. ServiceProvider

ServiceProvider は、Maya のエンジンカスタマイズが想定される箇所のファクトリ的機能を集中させた機能ブロックです。org.seasar.maya.provider.ServiceProvider インターフェイスを実装し、そのインスタンスは、ServiceProviderFactory を経由して取得されます。

ServiceProvider は、WEB アプリケーションのスコープにおいて保持されます。

1.3.1 ServiceProviderFactory の取得

ServiceProvider は、org.seasar.maya.provider.ServiceProviderFactory を通じて取得されます。

```
ServiceProviderFactory factory = ServiceProviderFactory.getDefaultFactory();
```

static メソッドである getDefaultFactory()で取得される ServiceProvider は、あらかじめ登録されているインスタンスです。Maya の基本パッケージにて、ServiceProviderFactoryImpl が提供されています。

```
static {
    ServiceProviderFactory.setDefaultFactory(new ServiceProviderFactoryImpl());
    ServiceProviderFactory.setServletContext(getServletContext());
}
```

ServiceProviderFactory をカスタム実装した場合には、setDefaultFactory()メソッドによってインスタンスをセットします。複数回セットされた場合には、最後にセットされたインスタンスが有効となります。また、ServiceProvider を正常に動作させるためには、static メソッドの setServletContext()を用いて、WEB コンテナのコンテキストをセットする必要があります。

1.3.2 ServiceProvider の取得

```
ServiceProvider provider = ServiceProviderFactory.getServiceProvider();
```

ServiceProvider は、ServiceProviderFactory の getServiceProvider()メソッドにて取得します。

1.3.3 S2 コンテナによる ServiceProvider 実装

maya.conf を読み込んで設定する Maya 標準の ServiceProviderImpl とは別に、S2 コンテナを活用した ServiceProvider の実装も提供予定です。maya.conf の代わりに、S2 の dicon 設定 XML ファイル（maya.dicon）を利用して ServiceProvider を初期化します。

1.4. ModelProvider

ModelProvider は、Maya のサービス時にモデルオブジェクトを必要とするときに、ユーザー定義より適切なオブジェクトを提供するファクトリです。org.seasar.maya.provider.ModelProvider インターフェイスを実装し、そのインスタンスは、ServiceProviderFactory を経由して取得されます。

ServiceProvider は、WEB アプリケーションのスコープにおいて保持され、その取得は ServletContext をキーとして行われます。

1.4.1 ModelProvider の取得

```
ModelProvider provider = ServiceProviderFactory.getModelProvider();
```

ModelProvider も、ServiceProviderFactory より取得します。

1.4.2 S2 コンテナによる ModelProvider 実装

S2 を利用した ModelProvider では、ModelProvider#getModel()メソッドによるオブジェクト取得について、標準の ModelProviderImpl では、クラスローダーを利用してモデルのクラスを読み込み、Class#newInstance()メソッドを用いてインスタンス生成しますが、これを、S2 コンテナ上のコンポーネントを取得するように機能を差し替えます。

ModelProvider#getModel()メソッドは、各エンジン・ページ・テンプレートのモデルオブジェクトの取得にも利用されますので、これらのモデルオブジェクトに、S2Dao 等の S2 コンテナ派生プロダクトを直接利用するなどの柔軟性が得られます。