XML Content Management with Infozone and Prowler

CONTENT HEAVY

FALKO BRÄUTIGAM, GERD MÜLLER AND ROBERT STRUNZ

The term content management has become somewhat overused these days, with a confusing range of providers and products trying very hard to distinguish themselves through specialised solutions. The following article takes a more detailed look at the approach of the open-source project Infozone.

Content management is closely linked to the development of large Web sites and their inherent problems. The sheer volume of data, the complexity of the structures and the related workflow make the use of software tools inevitable.

As the strategic importance of companies' Web sites to their business activity increases, the long-term prospects for providers of appropriate tools make this new market very attractive. Many large, and even more small, companies want a piece of the action. Consequently, they market their products under the term content management.

This has led to confusion in the market, but has at the same time made it very dynamic and interesting, especially as there is by no means a consensus on its future development and everyone is trying to make their mark.

One distinguishing feature is the ability of individual products to be integrated into existing IT structures. At this point the sophistication of different providers becomes apparent. While small providers concentrate on Web site administration, larger companies can and must also deliver integration into existing software systems such as ERP, document management or messaging solutions.

In this context, Web sites are no longer understood and used solely as a presentation medium, but are increasingly extended to become the company's central information platform. This leads to completely new content management

requirements. In addition to the editorial content, data from other sources now also has to be included and administered.

Infozone and Prowler

A very general approach to the issue of content management comes from the open-source community. The Infozone project (see the Infozone framework box) has as its aim the development and integration of components for building company portals (Enterprise Information Portal — EIP). The content management framework Prowler plays a central part in this.

Unlike other systems, Prowler is not a complete application but a Java framework, whose API follows the Java servlet model. This framework provides important content management functions, such as the administration of users, rights, versions and transactions, but in addition gives programmers the opportunity to integrate their own functions and extensions.

Transactional XML file system

At the heart of Prowler is a transactional XML file system, which makes it possible to insert the contents of the underlying data sources at any point of a hierarchical structure. Basically, any number of systems can be integrated as data sources, for instance application servers, ERP systems, mail or groupware servers, RDBMS, OODBMS and anything else that is important for business IT.

Unlike normal file systems, Prowler uses XML to display data and transactions are secure. Data from sources that cannot themselves supply XML are automatically converted into XML using special content adapters. This leads to a uniform XML view of all data, which substantially reduces the effort required for presentation via Cocoon and SchemoX (see the Infozone framework box), as well as for data queries and processing.

Apart from the consistent use of XML, another of Prowler's distinctive features is the integration of a JTA/XA compliant transaction manager. This

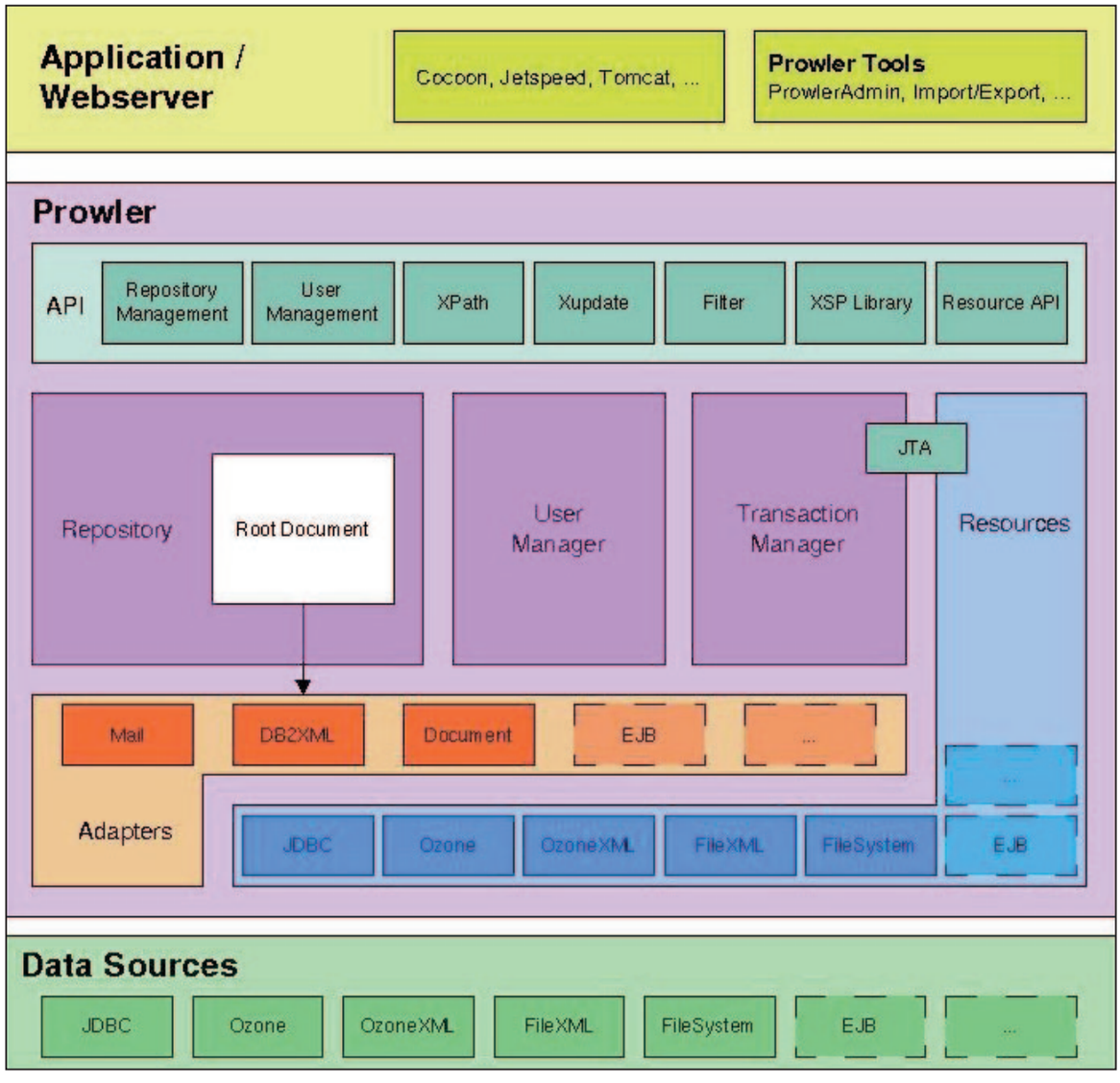

Figure 1: Prowler architecture

Info

http://www.infozone-group.org
http://java.sun.com/products/jta
http://tyrex.exolab.org
http://www.postgresql.org
http://www.ozone-db.org
http://xml.apache.org/cocoon
http://java.apache.org/jetspeed

Listing 1: Prowler configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<properties>
<prowler adapter="document"
loglevel="info warn error debug debug2 debug3 " name="intranet">
<usermanager  factory="org.infozone.prowler.um.lightweigt.LeightweightUserManager"/>
<transactionmanager factory="org.infozone.prowler.tm.lightweight.LightweightTransactionMana➥
ger"/>
</prowler>
<adapter alias="document"
factory="org.infozone.prowler.adapter.ProwlerDocumentAdapter"
xmlcache="ozonexml">
<prop name="versioning" value="false"/>
</adapter>
<adapter alias="db2xml"
factory="org.infozone.prowler.adapter.sql.DB2XMLAdapter"
xmlcache="ozonexml">
<prop name="propsFile" value="db2xml.properties"/>
<res name="postgres jdbc"/>
</adapter>

<resource alias="postgres jdbc" factory="org.infozone.prowler.resource.jdbc.JDBCResourceImpl">
<prop name="xadatasource" value="org.infozone.prowler.resource.jdbc.PostgresXADataSourceFa➥
ctory"/>
<prop name="pass" value=""/>
<prop name="hostname" value="localhost"/>
<prop name="port" value="5432"/>
<prop name="database" value="test"/>
<prop name="user" value="test"/>
<prop name="protocol" value="postgresql"/>
<prop name="driver" value="org.postgresql.Driver"/>
</resource>
<resource alias="ozonexml" factory="org.infozone.prowler.resource.ozone.OzoneXMLResource">
<prop name="user" value="intranet"/>
<prop name="url" value="ozonedb:remote://localhost:3333"/>
<prop name="passwd" value=""/>
</resource>
</properties>
```

enables access to different transactional data sources, for instance databases or application servers, within one global transaction or one HTML page. The transaction manager ensures data consistency through a two phase commit.

In addition, it is responsible for the flexible allocation of transactions to server threads, which makes it an important scalability factor, particularly for Web applications with thousands of simultaneous users.

Prowler, therefore, is not an extended editing system, but data integration technology that enables transactional access to the contents of a large variety of data sources through an XML interface.

Listing 2: Exemplary Java code

```
import org.infozone.prowler.*;
import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilder;

public class ProwlerTest {
public static void main( String[] args ) throws Exception {
// 1. Initialise Prowler
Prowler prowler = new Prowler( "prowler.xproperties" );
// 2. Open Session
session = prowler.newSession( "root", "" );
session.open( null );
// 3. Parse root document and set
DocumentBuilder db = ProwlerServices.newDocumentBuilder();
Document doc = db.parse( "rootdoc.xml" );
session.setRootDocument( doc );

// 4. Parse and link 'hamlet.xml'
doc = db.parse( "hamlet.xml" );
session.addDocument( doc, Prowler.newAdapterID( "document" ),
new ProwlerRepositoryPath( "/root/hamlet", session.rootDocument() ) );
// 5. Close session and shut down Prowler
session.close();
prowler.shutdown();
}
}
```

An example

Before things get too abstract, we will demonstrate Prowler's functionality by way of a simple example. Our little application is intended to perform two tasks: query and display the content of an SQL database and file an XML document in a database.

The following data sources are provided to Prowler as resources: an SQL database in the form of PostgreSQL and a database that can store XML directly in the form of Ozone/XML.

Which data sources are used with which parameters in Prowler applications is described in an XML file, independent of the actual logic itself. Listing 1 shows the configuration for the example.

We are assuming that Prowler has already been installed and that all files are in the current directory.

We are defining two resources: *postgres_jdbc* establishes the connection to PostgreSQL via JDBC and *ozonexml* is responsible for the link to Ozone. To make the content of both resources available as XML we define two adapters: *db2xml* converts the relational data from PostgreSQL to XML using the RDBMS/XML mapping tool DB2XML. *document* on the other hand does not convert data, but caches the XML documents in *ozonexml*, from where it can also retrieve them.

Now for the application itself. In principle there are two ways of working with Prowler, either directly through a Java API or through a Cocoon XSP library for the realisation of Web pages. We will demonstrate both methods briefly.

The Java API

In the first part of the application we will build a repository using the Java API and file an XML document in Prowler. For the XML document we are using that universally known and loved Shakespeare drama, *hamlet.xml*. The SQL database on the other hand contains a simple table *addresses* with the columns *name*, *firstname* and *town*, which already contains a few entries. The code, minus the exception handling, can be seen in Listing 2.

Prowler is initialised in steps 1 and 2 and then a session is opened. In step 3 the application-specific

Figure 2: Infozone architecture

Listing 3: An XML page for Cocoon

```
<?xml version="1.0"?>
<! File: jones.xml >
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="mystyle.xsl" type="text/xsl"?>
<! Here the Prowler namespace is defined and the corresponding XSP library linked>
<xsp:page
xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:response="http://www.apache.org/1999/XSP/Response"
xmlns:prowler="http://www.infozone-group.org/prowler">
<page>

<! Login>
<prowler:login id="login" user="root" passwd="" properties="prowler.xproperties"/>
<prowler:onSuccess idref="login">

<! Search for addresses>
<prowler:xpathQuery path-order="path1 path2" id="addresses">
<prowler:path id="path1">/root/addresses</prowler:path>
<prowler:path id="path2">/database/table0/record0[name/text()='Jones']</prowler:path>
</prowler:xpathQuery>

<prowler:onError idref="addresses">
Error retrieving addresses.
</prowler:onError>
</prowler:onSuccess>
<prowler:onError idref="login">
Login failed.
</prowler:onError>

</page>

</xsp:page>
```

Prowler structure is created (i.e. the root document). This structure is also XML and, in our example, looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
<hamlet/>
<addresses prowler:href="db2xml:
addresses order by name/" prowler:type=
"adapter"/>
</root>
```

It is interesting to note that the link to the address table is already inserted here. This linkage could of course also be created via the API. *prowler:href* points to the adapter *db2xml* with the table *addresses*, which means that the table is displayed as an XML document at this point if required.

Next we use *addDocument()* to store Shakespeare's *hamlet.xml* and put a link to it into the *hamlet* tag. That is the first application finished, and after its execution we have created a small repository.

Cocoon as front end

However, the Java API is only used for programming the specific presentation logic of applications. Anything else can be done much more easily, for instance with the Prowler XSP library. With this it is possible to dynamically integrate Prowler content in XML documents via Cocoon and to then convert it into HTML, WML, PDF or another output format. In order to be able to use the Prowler XSP library in Cocoon, the Cocoon configuration file must first be extended as follows:

```
processor.xsp.logicsheet.prowler.java =
resource://org/infozone/ui/xsp/prowler.xsl
```

The aim of our example page is to show the addresses of people called Jones. This is shown in Listing 3.

First is the usual login. Then follows an XPath query. For this we have to specify an explicit path through the repository. The *pathorder* attribute determines in which sequence the specified paths are to be processed.

We start with the root document at */root/addresses* and then move from link to link. As there is only one link to negotiate, the next part of XPath already contains the actual query: */database/table0/ record0 [name/text() ='Jones']*. The result is returned as a DOM tree and is integrated into the XML document via XSP. A style sheet that is not shown here converts the output into HTML and sends it to the browser. This little Web application makes do without a single line of Java code, but it achieves quite a lot.

Using Prowler's other functions via a Java API or an XSP library is just as easy as the data access itself. These are, for instance, user, rights and version administration, workflow support, transaction administration and the export/import of the entire content into a file system or other formats.

All of that cannot be described in detail in the context of this article. Nevertheless, we hope to have given some idea of content management in general and Prowler in particular. Questions relating to the article or to Prowler are welcome, either to the authors or straight to the Prowler mailing list. ■

The authors

The authors Gerd Müller, Falko Bräutigam and Robert Strunz work as computer scientists. Apart from their managerial tasks, Falko and Gerd primarily deal with technological questions concerning Infozone. Robert is responsible for SMB's business communication.

The Infozone framework – rationalisation through integration

Infozone was started in June 2000. The project, distributed under an Apache licence, is a Java/XML component framework for the professional creation of complex Enterprise Information Portals (EIP), providing access to a large variety of electronic data sources through integration. EIPs are designed to trigger rationalisation effects in larger companies and organisations by channelling information flows.
In this context, Infozone provides a company's employees with personalised browser-based access to all business-related information. But other user groups, such as customers, suppliers and partners can also access any of the company's data sources that they have been authorised to use.
Within the framework itself, some very central components are developed directly as part of the Infozone group. Others originate from the Apache project and have been selected and adapted for use in Infozone. Figure 2 shows the Infozone architecture. At the moment the following components are used:

- *Web Server/Servlet Engine – Apache/Tomcat*
- *XML Publishing Framework – Cocoon*
- *XML Content Management Engine – Prowler*
- *Object Oriented XML Database System – ozone/XML*
- *Forms Generator – SchemoX*

For the future, the Infozone group is planning the integration and development of a number of further modules. Top of the list is Jetspeed from Apache XML, which is intended to simplify the creation of personalised Web applications.