Sécurité des applications web

GUINKO Tonguim Ferdinand

7 décembre 2011

Table des matières

Chapitre 1

Flux syndiqués : rappels

Cours inspiré du livre «Java Web Programming with Eclipse», de David Turner et Jinseok Chae.

1.1 Création d'une application de base

1.1.1 Création du descripteur de l'application : web.xml

Ce descripteur web.xml :

1. définit une servlet nommée home et indique au container web que ce servlet doit être une instance de la classe package.HomeServlet ;
2. indique au container web que toute requête entrante à l'intention de home doit être traitée par l'instance de nom home.

```
<?xml version="1.0"?>
<web-app
     xmlns="http://java.sun.com/xml/ns/j2ee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
     version="2.4">
   <servlet>
      <servlet-name>home</servlet-name>
      <servlet-class>website.web.HomeServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>home</servlet-name>
      <url-pattern>/home</url-pattern>
   </servlet-mapping>
</web-app>
```

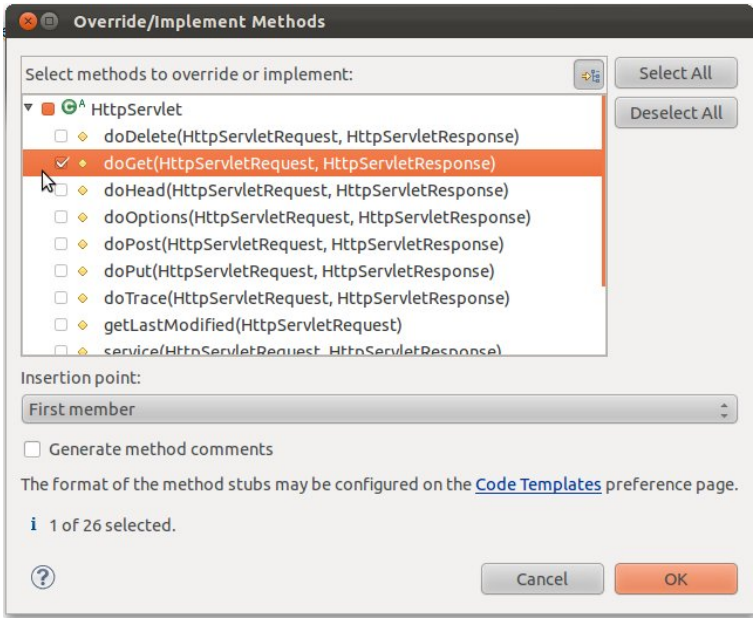
1.1.2 Création de la classe HomeServlet

Créez une servlet dans le package website.web. HomeServlet hérite de javax.servlet.http.HttpServlet.

Insérez la méthode DoGet() : pour cela, allez dans le menu source d'éclipse puis cliquez sur Override/Implements methods

Remplacez le corps de la méthode doGet par le code suivant :

```
PrintWriter writer = resp.getWriter();
writer.println("<h1>Bonjour les amis!</h1>");
```



Voici le code complet de la classe HomeServlet :

```
package website.web;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HomeServlet extends HttpServlet {

   protected void doGet(HttpServletRequest req, HttpServletResponse resp)
   throws ServletException, IOException {
      PrintWriter writer = resp.getWriter();
      writer.println("<h1>Bonjour les amis!</h1>");
   }
}
```

1.2 Création d'une vue

1.2.1 Création du fichier home.jsp

Créez la page home.jsp contenant le code suivant :

```
<jsp:useBean id="message" scope="request" class="java.lang.String" />
<html>
   <head>
      <title>Site web de Ferdinand</title>
   </head>
   <body>
      <h1>Page d'accueil</h1>
      <p>
 <%=message%>
      </p>
   </body>
</html>
```

La première ligne est un élément de type useBean. Le useBean génèrera un code qui tentera de localiser une instance d'une classe Java (un bean) portant le nom message dans un objet nommé request scope. En

d'autres termes, le useBean tentera de trouver la valeur associée à l'objet message dans l'objet HttpServletRequest crée par le container web pour la gestion de la requête courante. Si le useBean ne trouve pas la clé message elle créera alors une instance de java.lang.String en faisant appel à son constructeur sans argument (confère la définition d'un *bean*).

Modifiez le fichier la classe HomeServlet afin qu'elle génère une chaîne de caractères qui sera portée par l'objet requête HttpServletRequest et passée au fichier home.jsp

1.2.2 Modification de la classe HomeServlet

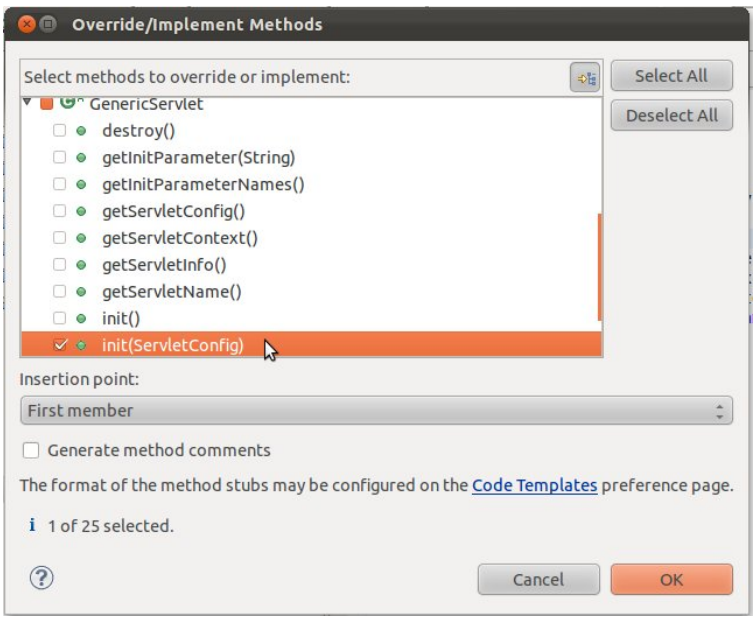
HomeServlet doit donc être modifiée afin de transmettre le traitement des requêtes à home.jsp. Pour cela, une seule instance de RequestDispatcher est nécessaire et sera créée dans la méthode Init() de la servlet HomeServlet. En rappel, la méthode Init() d'une servlet est celle qui est invoquée lorsque le container charge la servlet pour la première fois.

```
private RequestDispatcher homeJsp;
```

La valeur de homeJSP sera fixée dans la méthode init() de la servlet.

Insérez la méthode init(ServletConfig) : pour cela :

1. allez dans le menu source d'éclipse puis cliquez sur Override/Implements methods
2. déroulez l'onglet GenericServlet



La méthode Init() sera utilisée pour créer une instance de RequestDispatcher qui se chargera de l'exécution de home.jsp.

Pour cela, remplacez le corps de la méthode Init() par le code suivant :

```
ServletContext context = config.getServletContext();
homeJsp = context.getRequestDispatcher("/WEB-INF/home.jsp");
```

Ensuite, remplacez le contenu de la méthode doGet() par les lignes suivantes :

```
req.setAttribute("message", "Aurevoir!");
homeJsp.forward(req, resp);
```

Le contenu de la classe HomeServlet doit ressembler à ceci :

```
package website.web;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javax.servlet.RequestDispatcher;


@SuppressWarnings("serial")
public class HomeServlet extends HttpServlet {

    private RequestDispatcher homeJsp;

@Override
public void init(ServletConfig config) throws ServletException {
ServletContext context = config.getServletContext();
homeJsp = context.getRequestDispatcher("/WEB-INF/home.jsp");
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
req.setAttribute("message", "Aurevoir!");
homeJsp.forward(req, resp);
}
}
```

1.3 Création d'une application de syndication de flux

Téléchargez les librairies suivantes :

1. ROME 1.0 jar file `http://java.net/projects/rome/sources/svn/content/tags/rome-1_0_0/www/dist/rome-1.0.jar?rev=840`
2. JDOM `http://www.jdom.org/` : une librairie permetant la manipulation de documents XML ;
3. `purl-org-content-0.3.jar` : une librairie permettant de décomposer une URL pour par exemple en extraire un élément ;

1.3.1 Modification de la servlet HomeServlet

Modifiez la servlet HomeServlet de telle sorte qu'elle puisse trouver une source de flux syndiqués et rendre son contenu accessible à la page JSP.

Replacez le corps de la méthode doGet() par le code suivant :

```
URL url = new URL("http://rss.news.yahoo.com/rss/tech");
SyndFeedInput syndFeedInput = new SyndFeedInput();
SyndFeed syndFeed = null;
XmlReader xmlReader = new XmlReader(url);
syndFeed = syndFeedInput.build(xmlReader);
req.setAttribute("syndFeed", syndFeed);
homeJsp.forward(req, resp);
```

Importez les librairies suivantes dans la classe HomeServlet :

```
import java.net.URL;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.io.SyndFeedInput;
import com.sun.syndication.io.XmlReader;
```

Pour corriger la dernière erreur de compilation causée par l'instruction suivante :

```
syndFeed = syndFeedInput.build(xmlReader);
```

procédez de la manière suivante :

1. sélectionnez l'instruction entière ;
2. allez dans le menu source d'éclipse puis cliquez sur Surround with puis sélectionnez Try/catch block.

Deux gestionnaires d'erreurs seront générés :

1. l'un pour gérer une erreur de type IllegalArgumentException
2. l'autre pour gérer une erreur de type FeedException.

Le code de la classe HomeServlet doit ressembler à ceci :

```
package website.web;

import java.io.IOException;

import java.io.PrintWriter;
import java.net.URL;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import javax.servlet.RequestDispatcher;

import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.io.FeedException;
import com.sun.syndication.io.SyndFeedInput;
import com.sun.syndication.io.XmlReader;


@SuppressWarnings("serial")
public class HomeServlet extends HttpServlet
{

    private RequestDispatcher homeJsp;

@Override
public void init(ServletConfig config) throws ServletException
{
ServletContext context = config.getServletContext();
homeJsp = context.getRequestDispatcher("/WEB-INF/home.jsp");
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException
{
URL url = new URL("http://rss.radio-canada.ca/fils/nouvelles/nouvelles.xml");
SyndFeedInput syndFeedInput = new SyndFeedInput();
SyndFeed syndFeed = null;
XmlReader xmlReader = new XmlReader(url);
try
{
syndFeed = syndFeedInput.build(xmlReader);
} catch (IllegalArgumentException e)
{
// TODO Auto-generated catch block
```

```
e.printStackTrace();
} catch (FeedException e)
{
// TODO Auto-generated catch block
e.printStackTrace();
}
req.setAttribute("syndFeed", syndFeed);
homeJsp.forward(req, resp);
}
}
```

Testez l'application en saisissant le lien suivant dans la barre d'adresse de votre navigateur : `http://localhost:8080/SecuriteAppWeb/home`

1.3.2 Modification de la page home.jsp

Il s'agit de construire la page qui envoie le contenu au navigateur.

```
<%@ page import="com.sun.syndication.feed.synd.SyndFeed" %>
<%@ page import="com.sun.syndication.feed.synd.SyndEntry" %>
<%@ page import="java.util.Iterator" %>

<jsp:useBean id="syndFeed" scope="request" type="SyndFeed" />

<html>
   <head>
      <title>website</title>
   </head>
   <body>
      <h1>Flux syndiqués</h1>
      <h2><%=syndFeed.getTitle()%></h2>
      <ul>
 <%
   Iterator it = syndFeed.getEntries().iterator();
   while (it.hasNext())
   {
      SyndEntry entry = (SyndEntry) it.next();
 %>
    <li>
       <a href="<%=entry.getLink()%>"><%=entry.getTitle()%></a>
    </li>
 <% } %>
      </ul>
   </body>
</html>
```

1.3.3 Création du projet publisher

Créez un projet d'application web. Nommez le publisher.

Voici le contenu du fichier web.xml de ce projet :

```
<?xml version="1.0"?>
<web-app
     xmlns="http://java.sun.com/xml/ns/j2ee"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
</web-app>
```

Créez un fichier nommé nouvelles.rss dans le repertoire WebContent. Voici le contenu du fichier nouvelles.rss :

```
<rss version="2.0">
<channel>
  <title>Mes informations</title>
  <description>Ceci est mon site web personnel</description>
  <language>fr-ca</language>
  <item>
```

```
    <title>Site web de GUINKO T. Ferdinand</title>
    <link>http://tonguim.free.fr/</link>
    <description>Vous êtes ici sur le site web de GUINKO T. Ferdinand.</description>
  </item>
  <item>
    <title>Autre site web utile</title>
    <link>http://cse.csusb.edu/</link>
    <description>Ce site web est aussi très intéressant.</description>
   </item>
</channel>
</rss>
```

Déployez l'application publisher et visualisez la à l'URL http ://localhost :8080/publisher/nouvelles.rss.

Ensuite mettez à jour le contenu de la classe HomeServlet du projet SecuriteAppWeb en remplaçant l'URL qui s'y trouve par celle-ci http ://localhost :8080/publisher/nouvelles.rss puis exécutez, puis visualisez l'application SecuriteAppWeb.

1.4 Creation d'une application de publication de nouvelles

1.4.1 Création d'une base de données de nouvelles

```
create table news_item
(
   id integer primary key,
   titre text not null,
   url text not null
);

create table sequence
(
   next_value integer
);

insert into sequence value (1000);
```

La dernière instruction insert une ligne dans la table sequence. Le nombre inséré dans cette table est la valeur initiale de départ de l'incrémentation de la clé primaire de la table news_item.

MySQL est muni d'une fonctionalité permettant de gérer automatiquement l'incrémentation d'une clé primaire ; mais cette fonctionalité ne fait pas partie de la syntaxe normalisée du langage SQL, et de plus tous les SGBD tel que PostGreSQL ne sont pas munis d'une telle fonctionalité, d'où l'intérêt ici de proposer une structure générique supportée par tous les SGBD.

Créez, à la racine du projet publisher un repertoire nommé database

Créez, à l'intérieur du repertoire database, un script insertdb.sql avec le contenu suivant :

```
insert into news_item (id, title, url) values (1, 'CNN', 'http://www.cnn.com/');
insert into news_item (id, title, url) values (2, 'FOX News', 'http://www.foxnews.com/');
```

Créez, à l'intérieur du repertoire database, un script cleandb.sql avec le contenu suivant :

```
drop table if exists news_item;
drop table if exists sequence;
```

1.4.2 Création du fichier Ant build

Créez, à l'intérieur du repertoire database, un fichier nommé build.xml Le fichier build.xml doit avoir le contenu suivant :

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

<project name="publisher" default="all" basedir=".">
  <property name="mysql.params" value="-u publisher -ppublisher -D publisher" />
  <target name="all" depends="cleandb, createdb, insertdb"></target>

  <target name="cleandb">
     <exec executable="mysql" input="cleandb.sql">
<arg line="${mysql.params}" />
     </exec>
  </target>

  <target name="createdb">
     <exec executable="mysql" input="createdb.sql">
<arg line="${mysql.params}" />
     </exec>
  </target>

  <target name="insertdb">
     <exec executable="mysql" input="insertdb.sql">
<arg line="${mysql.params}" />
     </exec>
  </target>
</project>
```

1.4.3 Exécution des scripts

1. Faites un clique droit sur le fichier build.xml, puis sélectionnez Run As puis Ant Build ... Choisissez bien Ant Build ... avec les 3 points de suspension ;
2. Décochez la case à cocherall [default] (voir figure 1.4.3)
3. Cochez les case à cocher createdb et insertdb dans cet ordre
4. Vérifiez, dans la zone de saisie intitulée Target Execution Order, que les 2 scripts sont cités dans l'ordre suivant : createdb et insertdb
5. Cliquez sur Run

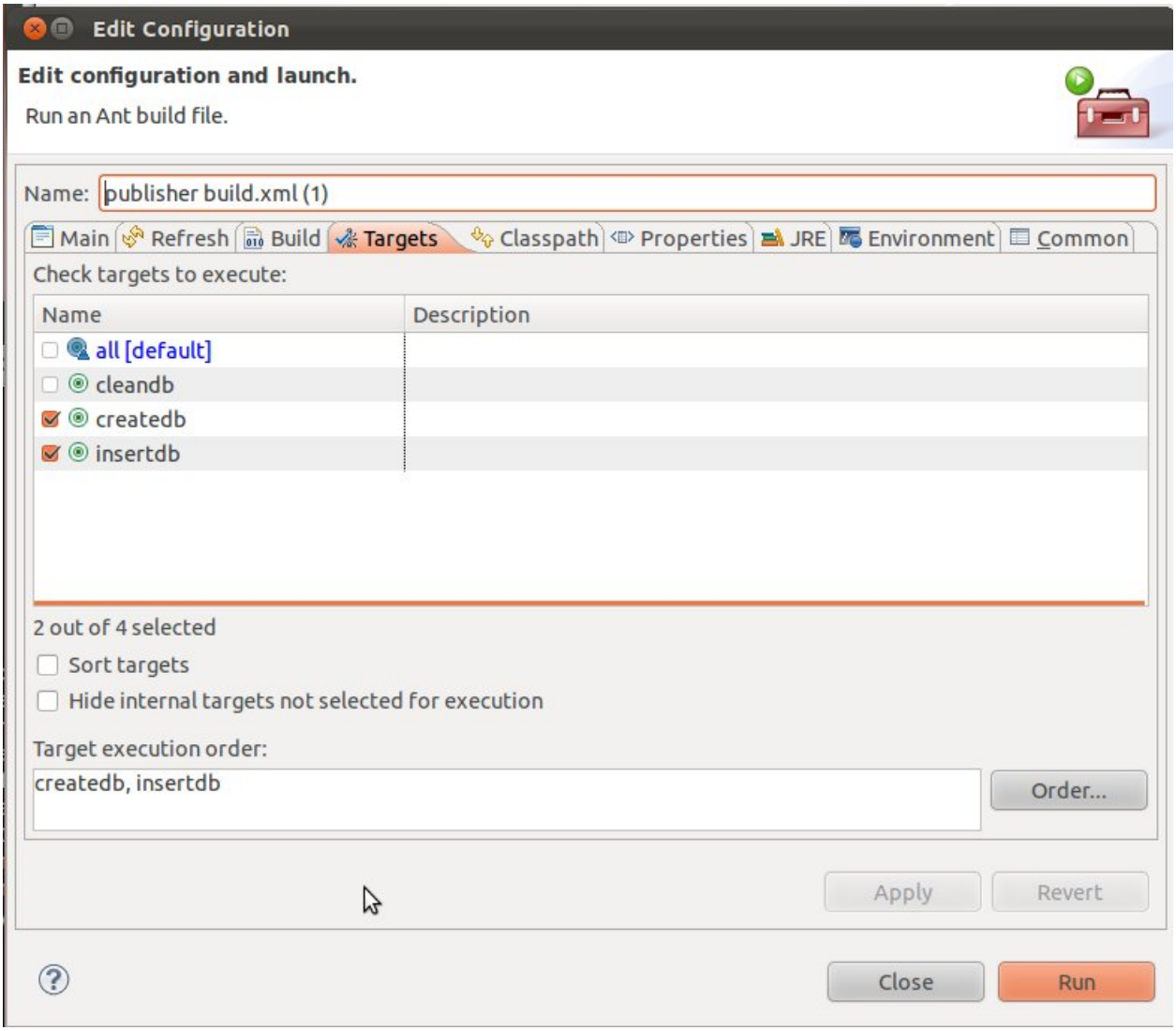


FIGURE 1.1 – Construction d'une application à l'aide de Ant

1.5 Génération dynamique de flux

1.5.1 Création d'une servlet

Créez, dans le package publisher.web, la servlet NewsFeedServlet qui hérite de la super classe HttpServlet,

Voici à quoi doit ressembler le contenu de la classe NewsFeedServlet

```
public void init(ServletConfig config) throws ServletException
{
try {
       Class.forName("com.mysql.jdbc.Driver");
}
catch (ClassNotFoundException e){
       throw new ServletException(e);
}
}
```

Voici le code complet de la classe NewsFeedServlet

```
package publisher.web;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.sun.syndication.feed.synd.SyndEntry;
import com.sun.syndication.feed.synd.SyndEntryImpl;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.feed.synd.SyndFeedImpl;
import com.sun.syndication.io.FeedException;
import com.sun.syndication.io.SyndFeedOutput;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.List;

/**
 * Servlet implementation class NewsFeedServlet
 */
@WebServlet("/NewsFeedServlet")
public class NewsFeedServlet extends HttpServlet {
@Override
public void init(ServletConfig config) throws ServletException
{
// TODO Auto-generated method stub
try
{
       Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException e)
{
       throw new ServletException(e);
}
}

protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

SyndFeed feed = new SyndFeedImpl();
feed.setFeedType("rss_2.0");
feed.setTitle("Mes nouvelles");
feed.setLink("http://localhost:8080/publisher/");
feed.setDescription("Ce flux d'information a été crêé en utilisant ROME.");
List<SyndEntry> entries = new ArrayList<SyndEntry>();

try {
    Connection connection = DriverManager.getConnection(
    "jdbc:mysql://localhost/publisher", "publisher", "publisher");
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement
    .executeQuery("select * from news_item;");
    while (resultSet.next()) {
String title = resultSet.getString("title");
String url = resultSet.getString("url");
SyndEntry entry = new SyndEntryImpl();
entry.setTitle(title);
entry.setLink(url);
entries.add(entry);
    }
    connection.close();
} catch (SQLException e) {
    throw new ServletException(e);
}

resp.setContentType("text/xml");

feed.setEntries(entries);
Writer writer = resp.getWriter();
SyndFeedOutput output = new SyndFeedOutput();
try {
    output.output(feed, writer);
} catch (FeedException e) {
throw new ServletException(e);
}
    }
}
```

L'instruction ResultSet contient le résultat de la commande executeQuery. Pour chaque ligne de l'instruction ResultSet, une ligne de flux syndiqué est formée et ajoutée à l'objet 'instruction SyndFeed à travers le code suivant :

```
<rss>
   <channel>
      <title></title>
      <item>
 <title></title>
 <link></link>
      </item>
      <item>
    :
    :
   </channel>
</rss>
```

Enfin, modifiez le fichier web.xml de l'application publisher ainsi qu'il suit :

```
<?xml version="1.0"?>
<web-app
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">
   <servlet>
      <servlet-name>news-feed</servlet-name>
      <servlet-class>publisher.web.NewsFeedServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>news-feed</servlet-name>
      <url-pattern>/news.rss</url-pattern>
   </servlet-mapping>
</web-app>
```

Testez successivement les 2 URL suivantes :

1. http ://localhost :8080/publisher/news.rss
2. http ://localhost :8080/SecuriteAppWeb/home