5- Web application deployment

Juan M. Gimeno, Josep M. Ribó

January, 2008

Contents

Introduction to web applications with Java technology

5- Web application deployment. Contents

Web application deployment. Concept

Deployment of a web application

Deploy a web application means to make it ready to be used by its clients

This is achieved by:

- structuring the files that constitute the web application in a certain standard way and by
- installing (usually, just moving) it in a certain location of the server

Application deployment. File Structure

The root directory of a web app. (`WebDir`) should contain the following files:

- Static files (.html/.png...): e.g., *header.html, updateForm.html, logo.png*

  They can also be located in a given directory (e.g. `html`)

- JSP pages: e.g., *showcatalog.jsp, dbUpdate.jsp*

  They can also be located in a given directory (e.g. `web`)

- `WEB-INF` (directory)

  This directory contains the deployment descriptor and the java packages needed by the web application

  This directory is explained in the following slide

- `META-INF` (directory)

  It contains some meta information about the `jar` or `war` files

  jar/war files are explained below

  The `context.xml` file (if the application has any) is also put here

  The `context.xml` *file is explained below*

The `WEB-INF` directory is constituted by:

- `web.xml` (*Web Application Deployment Descriptor*)

  File which contains information about the web application.

  This file describes the servlets (and possibly other components) that are used by the application. In particular, it contains:

  - Complete name of the classes that contain the servlets used by the application
  - Mapping of those classes to logical names used to refer to them in the application
  - Inicialization parameters for servlets
  - Container-managed security constraints

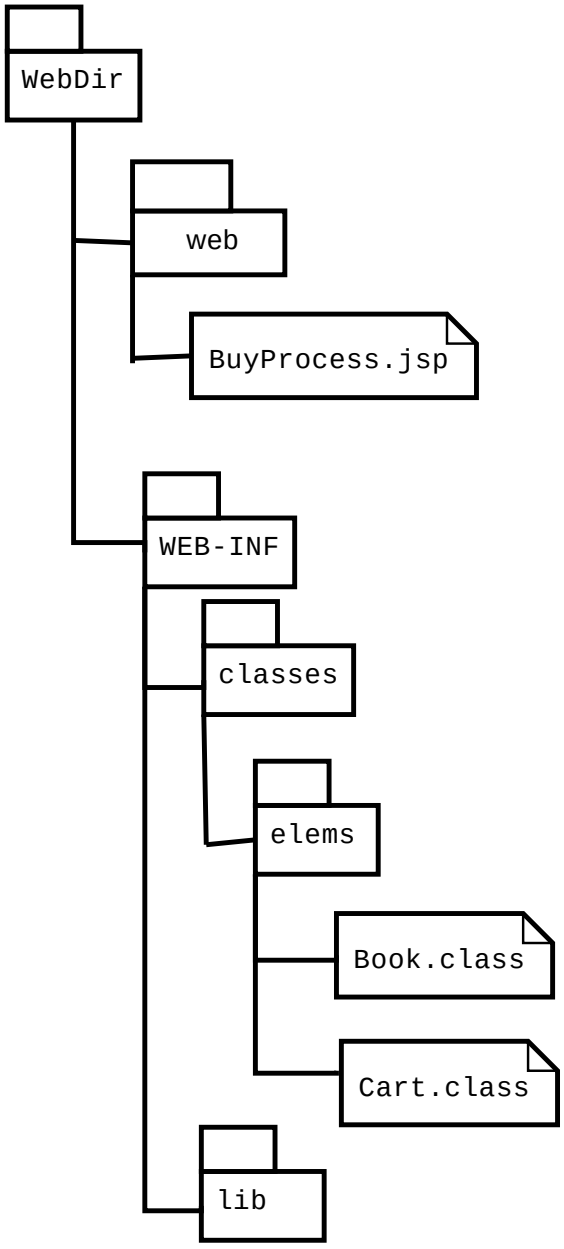
- `classes` (directory)

  Directory which contains classes which are used in the web application (e.g., beans and servlets). This classes are organized into directories according to their package structure.

  (e.g., the class `elems.Book` will be stored as `classes/elems/Book.class`

- `lib` (directory)

  Directory which contains classes which are used in the web application (e.g., beans and servlets). This classes are packaged and compressed in a `.jar` file. Example: `classes.jar`



Application deployment. Deployment descriptor

Deployment descriptor:
It is a file called `WEB-INF/web.xml` which contains information useful in order to deploy the web application

Some information that can be included in the `web.xml` file of a web application:

- Servlet aliases (see below)
- Time limit for sessions (see *Session module*)
- Tag libraries (see .....***)
- Global parameters for the application
- Some security configuration
- JNDI references for application resources (see ****)

The following example shows the general structure of the `web.xml` file associated to a web application. Most tags are optional. Other tags are also possible and have been omitted in this example

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

<web-app>

  <context-param> ...</context-param>
  <servlet>...</servlet>
  <servlet-mapping>...</servlet-mapping>
  <welcome-file-list> ... </welcom-file-list>
  <taglib>...</taglib>
  <ejb-ref>...</ejb-ref>

</web-app>
```

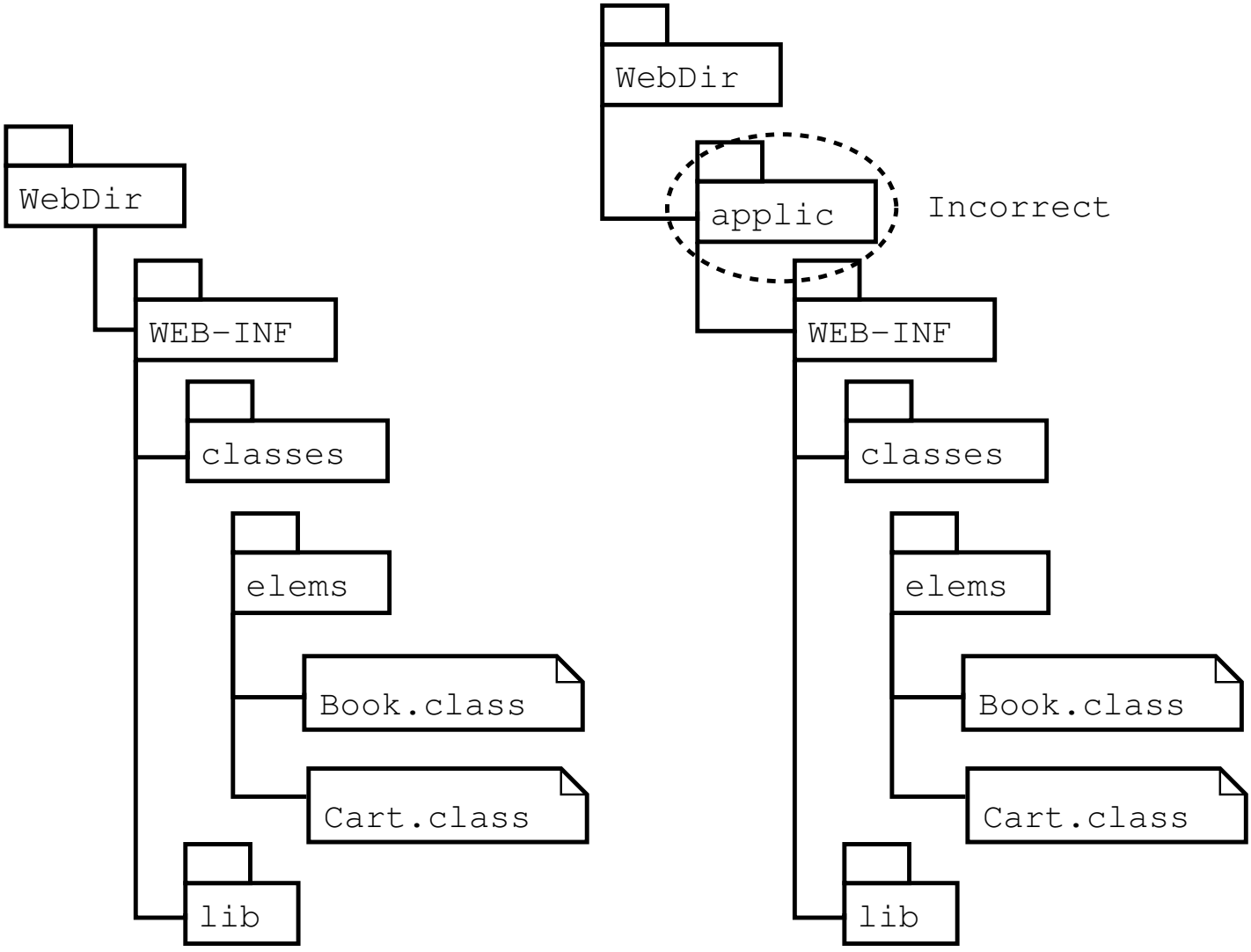
A complete description of the `web.xml` deployment descriptor can be found at:

- http://java.sun.com/webservices/docs/1.0/tutorial/doc/JavaWSTutorialTOC.html
- In addition, some of them will be presented in this course

A very important remark:

The directory `WEB-INF` must be located exactly in the directory labeled as `DocBase` in the `context` declaration:

```
<Context path="/mywebap" docBase="/home/josepma/WebDir"
         reloadable="true" debug="0"/>
```



Application deployment. Location

A web application should be deployed within a specific directory called **root directory of the web application**

Let `WebDir` be the root directory of a specific web application

If Tomcat is used as a container, `WebDir` can be located:

1. At `CATALINA_HOME/webapps`
2. In a context defined at `CATALINA_HOME/conf/server.xml`
3. In a `context.xml` file

1. **At** `CATALINA_HOME/webapps`

   (i.e., `CATALINA_HOME/webapps/WebDir`)

2. **At any directory that has been defined as a** `context` **with** `docBase="/.../WebDir"` **in the configuration file:** `CATALINA_HOME/conf/server.xml`

   Contexts are defined by means of the following declaration in `server.xml`:

```
<Context path="/mywebap" docBase="/home/josepma/WebDir"
         reloadable="true" debug="0"/>
```

   The directory `WebDir` (root directory) has been located in `/home/josepma` and is accessed:

   `http://ingrid.udl.net:8080/mywebap`

   `mywebap` acts as the *web application name*

   This option, although legal, is discouraged since Tomcat 6.0

3. **At any directory that has been defined within a `.xml` file in the Tomcat `conf` directory**

   `CATALINA_HOME/conf/engine_name/host_name/webap_name.xml`

   Example:

   The location of the application `mywebap` may be described by means of the following file:

   `CATALINA_HOME/conf/Catalina/localhost/mywebap.xml`

   with the following contents:

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


 <Context path="/mywebap" docBase="/home/josepma/WebDir"
   debug="0"/>
```

Application deployment. Web ARchive file

All the files that constitute a web application can be packaged and compressed into a `jar` file which has **war** extension and which is called **web archive file**

In Tomcat, **war files** are automatically deployed in the following cases:

1. If the war file (e.g., `test.war`)is located in the directory:

   `CATALINA_HOME/webapps`

   When Tomcat starts up, a directory `CATALINA_HOME/webapps/test` is created and the contents of `test.war` is unpackaged to this directory

2. If the `docBase` property of a `context` declaration (in the file `CATALINA_HOME/conf/server.xml` or in a context file at `CATALINA_HOME/conf`), contains a `war` file:

```
<Context path="/servletTest"
         docBase="/home/josepma/test/test.war"
         reloadable="true" debug="0" />
```

   When Tomcat starts up, a directory `CATALINA_HOME/webapps/servletTest` is created and the contents of `test.war` is unpackaged to this directory

An example of web application deployment with servlets

Example location:
`introapweb/examples/ex5.1`

This example shows how to deploy the application described in example 3.3

That application is composed of:

- A static html page `formParam.html` (an html form)
- A servlet class `Parameter.java` (which must be compiled to `Parameter.class`)

When the form is sent, the servlet class `Parameter.class` should be executed in the server

In this example we will explore how to compile and map a servlet in a web application

Deployment procedure:

1. **Select a physical directory where the application will be developed**

   We choose `/home/josepma/web`

2. **Create the directory structure for the web application:**

   - Directory `/home/josepma/web/WEB-INF`
   - Directory `/home/josepma/web/WEB-INF/classes`
   - Directory `/home/josepma/web/WEB-INF/lib`

3. **Put the file** `formParam.html` **in the directory** `/home/josepma/web`

4. **Compile the servlet** with:

   ```
   ==> javac -classpath SERVLET-JAR-PATH Parameter.java
   ```

   SERVLET-JAR-PATH is the path of the `servlet-api.jar` file. This file contains the package `javax.servlet` which is necessary to compile servlets

   For Tomcat 6.0, this path is `CATALINA_HOME/lib/servlet-api.jar`

   The result of this compilation process is the file `Parameter.class`

5. **Put** `Parameter.class` **into the directory**

   `/home/josepma/web/WEB-INF/classes`

6. **Create the file** `/home/josepma/web/WEB-INF/web.xml` containing (at least) the following data:

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

<web-app>
<servlet>
    <servlet-name>Servlet1</servlet-name>
    <servlet-class>Parameter</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Servlet1</servlet-name>
    <url-pattern>/serv</url-pattern>
</servlet-mapping>
</web-app>
```

   This file defines the servlet class and maps it to the label `/serv`

7. **Create the file** `ex5.1.war` **in the directory** `/home/josepma/web`

   From this directory issue the following command:

```
==> jar cvf ex5.1.war *
```

8. **Copy the file** `ex5.1.war` **to** `CATALINA_HOME/webapps`

   Tomcat will deploy the web application at this moment. A new directory:

   `CATALINA_HOME/webapps/ex5.1`

   with the same contents as `ex5.1.war` will be created

9. **Access the web application with the following address:**

   `http://localhost:8080/ex5.1/formParam.html`

10. **Access the servlet with the following address:**

    `http://localhost:8080/ex5.1/serv`

    Check that the html form `formParam.html` contains the following line:

```
<form action="http://localhost:8080/ex5.1/serv"
      method="get">
```

    Or, better, with a relative path:

```
<form action="serv"
      method="get">
```

Deployment of a war file with the Tomcat manager

The copy of the war file to `CATALINA_HOME/webapps` is not always possible because that directory may be write protected.

In this case the deployment of a web application can be done using the **Tomcat manager web application**

Deployment procedure with the Manager application:

1. At `CATALINA_HOME/conf/tomcat-users.xml` add the following line

```
<user username="someuser"
 password="somepassw"
 roles="manager"/>
```

   Now, the user `someuser` has the manager rights

2. At the Tomcat welcome page (e.g., `http://someServer:8080`) select the `Tomcat Manager` link and log in with the username and password stated at previous step

3. Deploy the `.war` file in the section *war file to deploy* of the manager page

   This action will copy the .war file to the folder stated in the appBase attribute of the `Host` section in the `CATALINA_HOME/conf/server.xml` file

   The `autoDeploy` attribute should be set to true

```
<Host name="localhost" appBase="webapps"
     unpackWARs="true" autoDeploy="true"
     xmlValidation="false" xmlNamespaceAware="false">
```

   The appBase attribute has by default the value of `webapps` (which points to `CATALINA_HOME/webapps`)

Automation of web application building and deployment

- **The process of building and deploying a web application may be automated**

- **The automation process is achieved by the use of the `ant` application**

  The `ant` application is a **build tool** that interprets and executes a file called `build.xml` which contains the instructions that are required to build and deploy the web application

- `ant` is an open-source project which is a part of **Apache Jakarta project**. It has a similar purpose to `make`, however there are some important differences between both tools

  In the Java world, `ant` is usually the preferred option

Process to automate the building and deployment of a web application

1. Create a development file structure

The web application is usually developed using a file structure similar to the following:

- `src`
  It contains the source code of the java classes of the web application (servlets, beans, classes to define tags...)
- `dd`
  It contains the deployment descriptors (e.g., `web.xml`)
- `web`
  It contains html, jsp, image files...
- `build.xml`
  The `build.xml` file contains the procedure to build and deploy the web application
  The syntax and semantics of the `build.xml` file is beyond the scope of this module.
  See `http://ant.apache.org` for a description

2. **At the root directory execute the `ant` build tool**:

   ```
   ==> ant <some-objective>
   ```

   This command will execute the commands contained in the build.xml file which are necessary to achieve that objective

   The build.xml file contains a default objective which is executed if the objective at the command line is ommitted:

   ```
   ==> ant
   ```

   Usually, the default objective builds and deploys the web application

   The building and deployment processes lead to two issues:

   - A new directory usually called `build` is created. That directory contains the compiled java classes and possibly the required java libraries
   - A file `webAppName.war` is created and copied to `CATALINA_HOME/webapps`
     Now the application is deployed and can be used

3. **Access the web application**

   `http://myserver.net:8080/webAppName`

Automation of web application building and deployment. Example

Example location:
`introapweb/examples/ex5.2`

- **General description**

  This is the example 5.1 with the building and deployment processes automated using the `ant` build tool

- **Building and deployment process**

  From the root directory (`introapweb/examples/ex5.2`) execute:

  ```
  ==>ant
  ```

  Access `http://localhost:8080/ex5.2/formParam.html`

- **Cleaning the generated staff**

  Execute:

  ```
  ==>ant clean
  ```

  The `build` directory, the `ex5.2.war` file and the `CATALINA_HOME/webapps/ex5.2.war` will be deleted

Other issues: package-structured servlets

If the servlet classes are organized into Java packages, this must be reflected in the directory hierarchy under `/WEB-INF/classes/`

For example, a Java class named:

```
com.mycompany.mypackage.MyServlet.class
```

would need to be stored in a file named:

```
/WEB-INF/classes/com/mycompany/mypackage/MyServlet.class
```

In addition, the first line of the file `MyServlet.java` will be the following:

```
package com.mycompany.mypackage;
```

Other issues: package-structured servlets

The file WEB-INF/web.xml:

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

<web-app>

<servlet>
    <servlet-name>Servlet2</servlet-name>
    <servlet-class>com.mycompany.mypackage.MyServlet
     </servlet-class>
</servlet>

<servlet-mapping>
     <servlet-name>Servlet2</servlet-name>
     <url-pattern>/serv2</url-pattern>
</servlet-mapping>

</web-app>
```

References

- Servlet 2.3 specification

  `http://jcp.org/aboutJava/communityprocess/final/jsr053/`

- `web.xml` file structure:

  `http://java.sun.com/webservices/docs/1.0/tutorial/doc/JavaWSTutorialTOC.html`