The Internet store project: practical details

Marcin Skubiszewski

9th March 2003

1 How to run software related to the project

This section describes how to run software needed by the Internet store project.

The project uses two servers: PostgreSQL (the database server) and Tomcat (the web server): the Internet store is only accessible to customers while the two servers are running. Sections 1.1 and 1.2 explain how to initialize and run the two servers.

Section 1.3 describes the locations and names of files that students need to access and modify in order to complete the project. Section explains how to compile Java programs that are part of the store.

1.1 Initializing and running the PostgreSQL database

In order to start the PostgreSQL database server, type to the shell

```
postgresql-perso start
```

This command starts a database attached to your account (other students access their respective databases, not yours).

To stop the database, type

```
postgresql-perso stop
```

Data belonging to your database are stored in the central server `trap.pl`, not in your computer. Therefore, every user can only run a database on one computer at a time: while a database runs for a user on some computer: after the execution of `postgresql-perso start` on a computer by a given user, and before the subsequent `postgresql-perso stop`, the user in question cannot run the database on a different computer.

But once you have typed `postgresql-perso stop` on one computer, your can type `postgresql-perso start` on another one.

When you start your database for the first time, you need to initialize it, by typing the following command to the shell:

```
schema.sh
```

You have to start the database first, and only then to initialize it: run `postgresql-perso start` first, and `schema.sh` second. The other way around will not work.

The `schema.sh` command sets up the database schema (i.e., organizes the database) so that it corresponds with the Internet Store project.

Attention: the `schema.sh` command also empties the database: it suppresses all existing articles, customers, pending orders etc. Therefore, only run `schema.sh` once, when you begin working on the project. Once you started filling the database with useful content, never run `schema.sh` again.

1.2 Initializing the Internet store and running the Tomcat web server

1.2.1 Initialization

Before you start working on the Internet store, you need to set up the environment for the Tomcat web server. For this purpose, type to the shell:

```
tomcat-init
```

Once you have started working on the project, you must not type the above command again: while initializing the environment of Tomcat, the command will destroy all the work that you did so far.

To avoid disasters, the **tomcat-init** command asks for confirmation, and only then proceeds.

1.2.2 Compilation

Once Tomcat has been initialized, you need to compile certain pieces of the source code belonging to the project. For this purpose, type the following two commands:

```
cd ~/classes
tomcat-javac
```

Whenever you compile source code, you need to check the output printed by `tomcat-javac` for error messages. If error messages are present, you must understand them and fix source files accordingly, then to compile again.

1.2.3 Running Tomcat

After a successful compilation, you can start Tomcat. For this purpose, type the following command to the shell:

```
tomcat-startup
```

Note the difference between PostgreSQL and Tomcat: PostgreSQL needs to be started first, and then to get initialized; Tomcat needs to be initialized first, and only then it can start running.

When you modify the Internet store, it is often necessary to restart Tomcat; otherwise, the modifications will not be taken into account. To restart Tomcat, type the following two commands to the shell:

```
tomcat-shutdown
tomcat-startup
```

(the first command stops Tomcat, the second one starts it again).

Note: In Tomcat, everything is slow initially: When you start tomcat for the first time after a `tomcat-init`, Tomcat can spend up to 3 minutes initializing itself. Similarly, when a new JSP file is used by Tomcat for the first time (after the initial creation or a modification of the file), Tomcat needs some time (say, up to a minute) to compile the file.

Therefore, whenever Tomcat does not respond immediately, you should wait for a few minutes before considering that there is a problem.

1.3 Accessing files related to the project

1.3.1 General information

With the exception of the database schema (which students should not modify), all the files related to a student's project are stored in the directory

```
~/web
```

Symbol "~" represents the home directory of the student in question (of course, this directory is different for every student).

The directory `~/web` contains many files in addition to those that are part of the project. These additional files were created when Tomcat and Struts were installed. They are necessary for the project. The rule is: if you do not understand the role of a file in `~/web`, do not delete or modify this file.

1.3.2 The files that we use

The project consists of the following kinds of files:

- **JSP (Java Servlet Page) files,** that describe how various webpages or webpage fragments should be displayed. We write JSP files using a mix of HTML and Java. The names of JSP files end in `.jsp`

  The JSP files are located in directory

  ```
  ~/web/webapps/sudety
  ```

  To access this directory, you can employ the following symbolic link:

  ```
  ~/jsp
  ```

- A **Stylesheet,** written in CSS. The names of stylesheets end in `.css`

  We only use one stylesheet, its pathname is

  ```
  ~/web/webapps/sudety/style.css
  ```

  or, using a symbolic link,

  ```
  ~/jsp/style.css
  ```

- **Java program files**, that represent our business logic (for example, Java programs search for articles, registers new customers, etc.). The names of Java program files end in `.java`.

  Java program files are not directly executed; they are compiled into pseudocode, then pseudocode is executed (pseudocode is not intended for understanding by humans). Pseudocode is stored in files whose names end in `.class`

  Our Java program files and the corresponding pseudocode are located in directory

```
~/web/webapps/sudety/WEB-INF/classes/sudety
```

To access this directory, you can employ the following symbolic link:

```
~/classes
```

- A **Struts configuration file**: this file describes the structure of the project and the rôle played by all other files. For example, struts configuration files describe which Java programs and which JSPs are to be called in a given situation. The Struts configuration file is written in XML, and its names end it `.xml`

  The pathname of this file is

  ```
  ~/web/webapps/sudety/WEB-INF/struts-config.xml
  ```

  or, using a symbolic link,

  ```
  ~/jsp/WEB-INF/struts-config.xml
  ```

- **The database schema**: this file describes the structure of the database that stores our data. It is written in a mix of shell and SQL. It is only executed once, while we initialize the store. Students do not need to modify this file.

  This file is located at

  ```
  /usr/local/bin/schema.sh
  ```

2 A few tips about using Linux

2.1 Setting up your graphical environment (KDE)

Your graphical environment should be KDE default (this is set up from the first-time wizard).

Your language should be American English (`en_US`). To obtain this:

- in the menu available from the bottom-left corner, choose ***Konfiguracja*** or ***Configuration***
- inside, choose ***KDE***
- inside, choose ***Personalization***
- inside, choose ***Country and Language***
- inside, choose ***United States of America***
- click ***OK***

It is convenient to have small window titles. To obtain this:

- in the menu available from the bottom-left corner, choose ***Configuration***
- inside, choose ***KDE***
- inside, choose ***LookNFeel***

- inside, choose ***Window Decoration***
- inside, choose ***B II***
- click ***OK***

To have a shell window with a small font (that takes littke space on your screen):

- in the menu available from the bottom-left corner, choose ***Run command***
- a window appears; in the window, type
  - `xterm -font 6x12` for a very small font, or
  - `xterm -font 6x13` for a moderately small font
- click ***OK***

2.2 Various Linux commands

Whenever you start a new shell, type "`set -P`". This command will cause the shell to behave sensibly in presence to symbolic links.

To read information about *command*, type "`man` *command*" (`man` stands for manual).

To change your account's password, use command `yppasswd`.

To list all files in your current directory, type `ls -FC` (short list) or `ls -l` (long, informative list).

To start a virtual terminal with a small font, and save space on your screen, type `xterm -font 6x12`.

To change your current directory, type `cd` *destination*, where *destination* is the new current directory that you want to use. For example,

```
cd ~/classes
```

will take you to the directory where your Java programs reside. Typing "`cd`" alone is equivalent to typing "`cd ~`", and takes you to your home directory.

Command "`pwd`" (print working directory) displays the currrent directory.

To edit a file, type "`emacs` *file*".

3 Saving your work with `RCS`

RCS is the *revision control system*. RCS can remember the history of every file on which you work, i.e., can remember the content of the file at various points in time. The content of a specific file at some point in the past is called *version* or *revision*.

RCS makes it easy to keep track of changes that you brought to your project. RCS is useful if

- you want to revert changes that you introduced into one of your files (because you have discovered that you did something wrong)
- you want to compare the current version of one of your files with a previous version (because you want to know what precisely you have changed recently).

For every file named `file` RCS creates a history file named `file,v` (for example, the history file for `Item.java` is `Item.java,v`).

RCS is no substitute for saving a copy of your work: because the RCS history files are stored next to the files with which they correspond, any problem that results in file damage will likely cause damage to all files: both your ordinary files and the corresponding RCS history files. As a result, RCS will be unable to restore any version (old or recent) of your files.

To register with RCS the current version of *file*, type "`ci -l` *file*". Once you have registered a version of a file with RCS, this version remains available forever.

RCS never spontaneously registers versions: RCS will only remember the content of *file* at the precise times when you typed "`ci -l` *file*". If, so far, you typed "`ci -l` *file*" ten times (at ten different points in time), then RCS holds ten different versions of *file*, numbered from 1.1 to 1.10.

To compare *file*, as it exists currently, with the last version known to RCS, type "`rcsdiff` *file*".

To compare *file*, as it exists currently, with the version registered under the number *1.n*, type "`rcsdiff -r`*1.n file*".

To compare versions *1.n* and *1.m* of *file* with each other, type "`rcsdiff -r`*1.n* `-r`*1.m file*".

To retrieve version *1.n* of *file*, and make it accessible under the temporary name *temp*, type "`co -p`*1.n file > temp*".