Contents

Appendix A

Traffic danger web system deployment

A.1 Environment setup

Java applications are typically compiled to bytecode that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Because of Java platform independence, the system can be deployed on various environments.

For the project to be deployed, following applications should be pre-installed:

- PostgreSQL database engine,
- Java Runtime Environment (JRE),
- Apache Tomcat servlet container.

A.1.1 PostgreSQL installation

WINDOWS

Get the latest version of PostgreSQL executable package from `http://www.postgresql.org/download/windows/`. After downloading simply double click on that package. The installer will do the job.

LINUX

Get the latest version of PostgreSQL sources. For the date of writing that file latest version of PostgreSQL was v9.0beta2. Sources can be obtained by anonymous FTP from `ftp://ftp.postgresql.org/pub/source/v9.0beta2/postgresql-9.0beta2.tar.gz`. After downloading, unpack the file:

```
gzip -dc postgresql-9.0beta2.tar.gz | tar xf -
```

This will create a directory `postgresql-9.0beta2` under the current directory with the PostgreSQL sources. Enter into that directory and execute installation procedure:

```
./configure
gmake
su
gmake install
adduser postgres
mkdir /usr/local/pgsql/data/
chown postgres /usr/local/pgsql/data/
su - postgres
/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data/
/usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data/ >logfile 2>&1 &
/usr/local/pgsql/bin/createdb test
/usr/local/pgsql/bin/psql test
```

A.1.2 Java Runtime Environment (JRE) installation

Download the Java 2 Standard Edition Runtime (JRE) release version 5.0 or later, from `http://www.java.com/en/download/manual.jsp` and install the JRE according to the instructions included with the release.

Set the environment variable named `JRE_HOME` to the pathname of the directory into which you installed the JRE, e.g.

LINUX

```
# for Bourne, bash, and related shells
export JRE_HOME=/usr/local/java/jre5.0

# for csh and related shells
setenv JRE_HOME=/usr/local/java/jre5.0
```

WINDOWS

```
set JRE_HOME=C:\jre5.0
```

You can also use the full JDK rather than just the JRE. In this case set the `JAVA_HOME` environment variable to the pathname of the directory into which you installed the JDK.

A.1.3 Apache Tomcat installation

Download the latest version of Tomcat binary distribution from `http://tomcat.apache.org/`, appropriate for your system and unpack the file into convenient location so that the distribution resides in its own directory:

LINUX

```
cp apache-tomcat-6.0.26.tar.gz /usr/local/apache/
cd /usr/local/apache/
gzip -dc apache-tomcat-6.0.26.tar.gz | tar xf -
```

Set the `CATALINA_HOME` environmental variable (it is used to refer to the full pathname of the release directory):

LINUX

```
# for Bourne, bash, and related shells
export CATALINA_HOME=/usr/local/apache/apache-tomcat-6.0.26

# for csh and related shells
setenv CATALINA_HOME=/usr/local/apache/apache-tomcat-6.0.26
```

WINDOWS

```
set CATALINA_HOME=C:\apache\apache-tomcat-6.0.26
```

Startup Tomcat:

LINUX

```
$CATALINA_HOME/bin/startup.sh
```

WINDOWS

```
%CATALINA_HOME%\bin\startup.bat
```

After starting the default web applications included with Tomcat will be available at address: `http://localhost:8080/`

If Tomcat is not responding, the reason can be another web server (or process) running and using provided port `8080`. The solution is to edit `$CATALINA_HOME/conf/server.xml` configuration file and change the default port.

Shutdown Tomcat:

LINUX

```
$CATALINA_HOME/bin/shutdown.sh
```

WINDOWS

```
%CATALINA_HOME%\bin\shutdown.bat
```

A.2 Loading and configuration

Open the *Tomcat Manager* available under the *Tomcat Administration* panel at address `http://localhost:8080/manager/html/`:

If you are not authorized to view this page, you should probably examine `$CATALINA_HOME/conf/tomcat_users.xml` file and, if necessary, define a new user with appropriate rights, e.g. `<user username="root" password="toor" roles="manager-gui" />`.

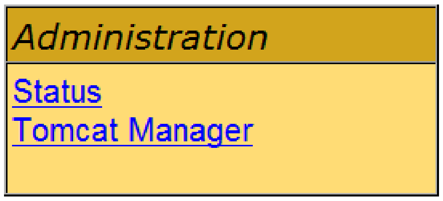


Figure A.1 Tomcat administration panel

Under the *WAR file to deploy* section shown in Figure A.2, deploy `traffic_web-1.0.0.war` archive, containing prebuild web application for traffic danger web system.

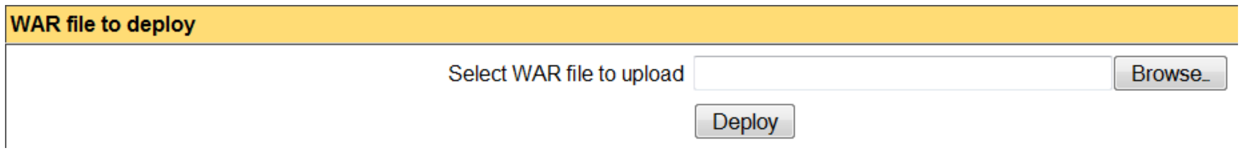


Figure A.2 WAR archive deployment section

After deployment is complete, open for edition `$CATALINA_HOME/webapps/traffic_web/WEB-INF/web.xml` file and change value of `OntologyURI` parameter for appropriate path indicating ontology file for traffic danger `TrafficDanger.owl`.

The URI can identify local or remote resource, for example:

- `file:///C:/Users/jwa/masters/TrafficDanger.owl` (local Windows location),
- `file:///home/jwa/masters/TrafficDanger.owl` (local Linux location),
- `http://host/TrafficDanger.owl` (remote location).

Final step is about SQL scripts execution, required for database creation. Run the following scripts in the given order: `postgres_traffic_user.sql`, `postgres_traffic_database.sql`, `postgres_traffic_schema.sql` and finally `postgres_traffic_data.sql`. The last one actually contains sample data, so its execution is optional.

LINUX

Change user to `postgres`, and change catalog to directory containing database scripts and invoke following commands:

```
psql -f postgres_traffic_user.sql
psql -f postgres_traffic_database.sql
psql -f postgres_traffic_schema.sql traffic
psql -f postgres_traffic_data.sql traffic
```

After that, system should be ready to cooperate under address `http://localhost:8080/traffic_web-1.0.0/board.html`.

Appendix B

Additional tools installation instructions

B.1 Protégé installation

B.1.1 Editor installation

The installation instructions can be found on Protégé home page [4]. First, download the latest version of Protégé tool from the Protégé home page. Choose installation variant based on your system type.

WINDOWS

After downloading double-click `install_protege_4.0.2.exe`.

If you do not have a Java virtual machine installed, be sure to download the package above which includes one.

LINUX

After downloading open a shell and go to the directory of downloaded installer. At the prompt type:

```
sh ./install_protege_4.0.2.bin
```

If you do not have a Java virtual machine installed, be sure to download the package above which includes one. Otherwise you may need to download one from Sun's Java website [6] or contact your OS manufacturer.

ALL OTHER PLATFORMS

1. Instructions for Unix or Unix-like operating systems
   - For Java 2, after downloading, type

     ```
     java -jar install_protege_4.0.2.jar
     ```
   - For Java 1.1, after downloading, type

     ```
     jre -cp install_protege_4.0.2.jar install
     ```
   - If that does not work, try

     ```
     java -classpath [path to] classes.zip:install_protege_4.0.2.jar install
     ```
   - If that does not work either, on sh-like shells, try

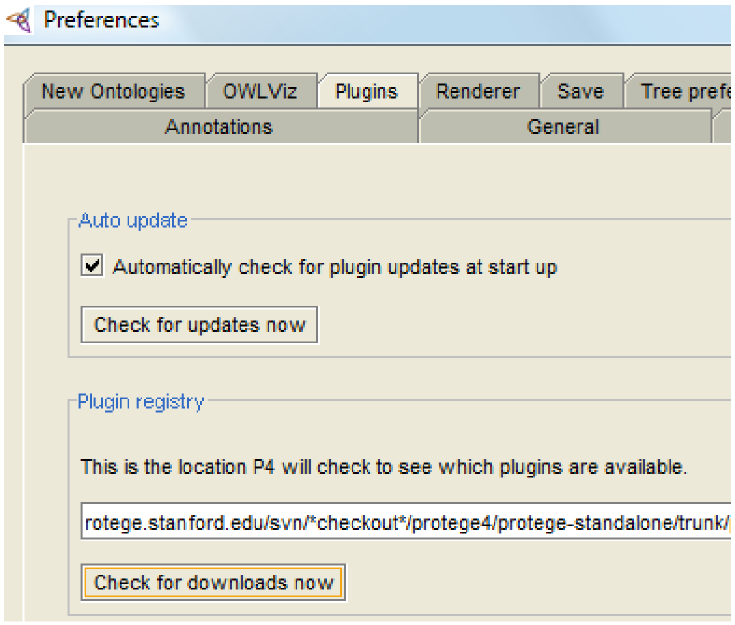
     ```
     cd [to directory where install_protege_4.0.2.jar is located]
     CLASSPATH=install_protege_4.0.2.jar
     export CLASSPATH
     java install
     ```
   - Or for csh-like shells, try

     ```
     cd [to directory where install_protege_4.0.2.jar is located]
     setenv CLASSPATH install_protege_4.0.2.jar
     java install
     ```
2. Instructions for other platforms
   - Be sure you have Java installed. You can download Java from Sun's website [6].
   - In a console window, change to the directory where you downloaded `install_protege_4.0.2.jar` to before running the installer.

Your operating system may invoke Java in a different way. To start the installer, add `install_protege_4.0.2.jar` to your `CLASSPATH`, then start the main class of the installer named `install`.

B.1.2 Plugins installation

After installation of Protégé there can be custom need of installation additional plugins. For plugins installation go to *File->Preferences->Plugins* tab and click *Check for downloads now* button (Figure B.1).



☑ Automatically check for plugin updates at start up

This is the location P4 will check to see which plugins are available.

rotege.stanford.edu/svn/*checkout*/protege4/protege-standalone/trunk/

Figure B.1 Plugins tab

For all additional instructions check the wiki page for Protégé [5]. That is very competitive and reliable source of all information that may be needed. It includes documentation, tutorials, sample ontologies, plugin libraries, etc.

B.2 Maven installation

Maven will be very helpful while working with sources of the project, because it can automatically download all dependencies tree from the network.

Maven is a Java tool, so you must have Java installed in order to proceed. Java Development Kit (JDK) is required (Java Runtime Environment (JRE) is not sufficient). Installation instructions provided below can be found on Maven home page [3].

Windows

1. Download the latest Maven archive from Maven home page and unzip the distribution archive, i.e. `apache-maven-2.2.1-bin.zip` to the directory you wish to install Maven 2.2.1. These instructions assume you chose `C:\Program Files\Apache Software Foundation\`.
2. Add the `M2_HOME` environment variable by opening up the system properties (`WinKey+Pause`), selecting the *Advanced* tab, and the *Environment Variables* button, then adding the `M2_HOME` variable in the user variables with the value `C:\Program Files\Apache Software Foundation\apache-maven-2.2.1`.
3. In the same dialog, add the `M2` environment variable in the user variables with the value `%M2_HOME%\bin`.
4. In the same dialog, update/create the `PATH` environment variable in the user variables and prepend the value `%M2%` to it (in order to make Maven available in the command line).
5. In the same dialog, make sure that `JAVA_HOME` exists in your user variables or in the system variables and it is set to the location of your JDK, e.g. `C:\Program Files\Java\jdk1.5.0_02` and that `%JAVA_HOME%\bin` is in your `PATH` environment variable.

Unix-based operating systems

1. Download the latest Maven archive from Maven home page and unzip the distribution archive, i.e. `apache-maven-2.2.1-bin.zip` to the directory you wish to install Maven 2.2.1. These instructions assume you chose `/usr/local/apache-maven/`. The subdirectory `apache-maven-2.2.1` will be created from the archive.
2. In a command terminal, add the `M2_HOME` environment variable, e.g.

   ```
   export M2_HOME=/usr/local/apache-maven/apache-maven-2.2.1
   ```

3. Add the `M2` environment variable, e.g.

   ```
   export M2=$M2_HOME/bin
   ```

4. Add `M2` environment variable to your path, e.g.

   ```
   export PATH=$M2:$PATH
   ```

5. Make sure that `JAVA_HOME` is set to the location of your JDK, e.g.

   ```
   export JAVA_HOME=/usr/java/jdk1.5.0_02
   ```

   and that `$JAVA_HOME/bin` is in your `PATH` environment variable.

Appendix C

Working with sources

When it comes to editing sources (for reasons like extending current functionality, fixing unexpected bug, etc.), we should prepare convenient developing environment. Project is written using Eclipse IDE, and is supported by Maven (installation steps in Section B.2). For Eclipse to support Maven, we should install M2Eclipse plugin. Installation is straightforward, because it is automatic, and can be found on M2Eclipse home page [2]. For me, that 3 mentioned tools are synonym of convenient developing environment in the Java world.

Traffic danger application consists of 3 projects:

- `traffic_ont` - contains logic responsible for interacting with ontologies,
- `traffic_domain` - contains logic responsible for interacting with database,
- `traffic_web` - integration part, contains web logic, and UI.

For opening the sources, we should import that 3 projects into Eclipse workspace. When edition is complete, we can compile and install project to our local repository using Maven. What is more, complete web archive will be created, with all dependent resources and libraries. Such prepared package can be then deployed through Tomcat manager (as shown in Section A.2).

Because of comprehensive configuration files and appropriate projects structure (compatible with Maven standard layout [1]), installation can be accomplished in 3 steps shown below (suppose, that all projects are located in `C:\workspace` directory):

```
cd C:\workspace\traffic_ont\
mvn clean install
cd C:\workspace\traffic_domain\
mvn clean install
cd C:\workspace\traffic_web\
mvn clean install
```

After execution of that trivial commands, in our local Maven repository, suppose it is `C:\.m2\repository\`, a subdirectory containing 3 projects will be created. Directory structure looks like that:

```
|--repository
   |--...
   |--agh
      |--traffic
         |--traffic_ont
            |--1.0.0
               |--traffic_ont-1.0.0.jar
               |--...
         |--traffic_domain
            |--1.0.0
               |--traffic_domain-1.0.0.jar
               |--...
         |--traffic_web
            |--1.0.0
               |--traffic_web-1.0.0.war
               |--...
```

The complete web archive `traffic_web-1.0.0.war` is now created, and ready to be deployed on server (see Appendix A).

Alternative way of quick loading Maven webapp project into Tomcat servlet container is available by invoking following command:

```
mvn tomcat:run
```

Bibliography

[1] Introduction to the Standard Directory Layout for Maven. `http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html`. Accessed September 2010.

[2] M2Eclipse home page. `http://m2eclipse.sonatype.org/`. Accessed September 2010.

[3] Maven home page. `http://maven.apache.org/`. Accessed September 2010.

[4] Protégé home page. `http://protege.stanford.edu/`. Accessed September 2010.

[5] Protégé wiki page. `http://protegewiki.stanford.edu/`. Accessed September 2010.

[6] Sun home page. `http://java.sun.com/`. Accessed September 2010.