

---

# Web Application Resource for Accessing Relational Entities

1.1 Release Installation Guide

Andre Michel Gauthier, Gauthier Services Limited

<nomad@gserv.co.uk>

Copyright © 2003 Gauthier Services Limited

This document is intended to provide basic installation instructions for WARFARE. Once all of the prerequisites have been satisfied then the rest should be straight forward.

## Table of Contents

|                                      |   |
|--------------------------------------|---|
| Terms and Conditions .....           | 1 |
| Pre-Requisites .....                 | 3 |
| Installing the Databases .....       | 3 |
| Installing the Web Application. .... | 4 |
| Upgrading Databases .....            | 5 |
| Upgrading Web Application .....      | 6 |
| Modules .....                        | 6 |

## Terms and Conditions

WARFARE is distributed under the Artistic License as I would like the framework to retain some resemblance to it's original form of being very small and lightweight. Others are encouraged to enhance the framework via the plug-in method as documented in the User Guide. Plug-ins can then be distributed separately under thier own licence or incorporated into the standard distribution.

One of the demo applications uses jfreechart which is distributed under the LGPL license. Please refer to LICENCE.jfreechart.txt and README.jfreechart.txt for details of this library, it's licence and where to get it from.

### The Artistic License

#### Preamble

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

#### Definitions:

- \* "Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.
- \* "Standard Version" refers to such a Package if it has not been

modified, or has been modified in accordance with the wishes of the Copyright Holder.

\* "Copyright Holder" is whoever is named in the copyright or copyrights for the package.

\* "You" is you, if you're thinking about copying or distributing this Package.

\* "Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

\* "Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b) use the modified Package only within your corporation or organization.

c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b) accompany the distribution with the machine-readable source of the Package with your modifications.

c) accompany any non-standard executables with their corresponding Standard Version executables, giving the non-standard executables non-standard names, and clearly documenting the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package.

7. C or perl subroutines supplied by you and linked into this Package shall not be considered part of this Package.

8. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

9. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

## Pre-Requisites

- PostgreSQL 7.4.2 or Later
- Tomcat 4.1.12 or later
- Ant 1.5, with xalan.jar installed in the `${ant.home}/lib`

## Installing the Databases

**Create the database instance (vm).** The database creation scripts expect there to be a database called vm this can be created using the `createdb` command for postgres.

### Example 1. Creating a database

```
createdb vm
```

You will need to have created a database user that is able to create a database using the `createuser` command. This command should be run as the postgres user.

### Example 2. Creating a user

```
createuser -d nomad
```

**Build the database.** From the sql directory run ant. This will in turn create all the tables and populate them with the views required for the view manager application. The build.xml specifies a number of properties to connect to the database.

- jdbc.driver=org.postgresql.Driver
- jdbc.classpath=/usr/local/pgsql/share/java/postgress.jar
- jdbc.dburl=jdbc:postgresql://127.0.0.1:5432/vm
- jdbc.user=nomad
- jdbc.pass=

These parameters can be overridden on the ant command line with -D

### Example 3. Building the Database

```
ant -Djdbc.user=barney
```

## Installing the Web Application.

**Deploy the WAR file.** Simply copy the data.war file from the dist directory into the webapps directory of your Tomcat installation. This should be automatically unpacked and a default context created once you start Tomcat.

**Configure the DataSources.** The demo application expects there to be four datasources available. The preferred method for providing access to the Databases is to create datasources in the application context within Tomcat. You should refer to the Tomcat documentation to find out how to do this if you are unsure. The required JNDI names for the datasources are:

- jdbc/vm should connect to the vm database
- jdbc/manager should connect to the vm database (this is to enable updates via the management interface to update a master database whilst allowing the configuration to be read from a slave under high load environments)
- jdbc/pm should connect to the pm database
- jdbc/intra should connect to the intra database

An example context xml file is provided (dist/warfare.xml) with these data sources configured. You should edit this file to suit your own Postgress install. This file can then be copied into the webapps directory. You will also need to copy the jar file containing the JDBC drivers for Postgress into the common/lib directory of your Tomcat installation (this is supplied with Postgres). If you are not installing the demo applications then you will only need to create the vm and manager connections.

**Post Install Configuration.** If you want to be able to save template files edited within the View Manager application you will need to configure the location where files should be saved to.

- Locate the viewmanager.properties file within the war file or exploded webapp directory in Tomcat
- Locate the line containing files.savedir
- Replace the value for this property with the directory in which you wish to save files to. The entry recommended is the WEB-INF/classes directory under the directory where tomcat has un-packed the war file. This is the location where most of the templates in the demo reside and therefore any changes will be found. You can specify any directory but that directory must be on the CLASSPATH of tomcat for any saved files to be located by the application framework. An alternative location not on the CLASSPATH can be used if you wish changes to be verified and controlled before being deployed. I.e. save files in /tmp and then when files have been approved copy them manually into the WEB-INF/classes directory. Templates can also be located inside a jar file in the WEB-INF/lib directory to enable stricter version control on deployed templates if desired.
- Locate the lines referring to the scheduler config
- Alter the value of the saveto parameter to the location of the scheduler.xml file in your installation. Its location within the WEB-INF/classes directory causes a default configuration of tomcat to re-load the context every time this file changes. This will force you to re-log in each time you change the configuration in the admin application. You can change the location completely to be outside of the webapp directory but you will need to manually restart tomcat to re-load the context to get the scheduler to re-read the configuration file.

## Upgrading Databases

**Pre-Upgrade Actions.** Perform a dump of any databases you have made updates to incase the upgrade fails and you need to recover the previous version. The command `pg_dump $DBNAME` will perform this for you. E.g `pg_dump pm > pm.dmp`. This can then be recovered by `pg_restore` if the upgrade fails.

**Migrate the schema if required.** If you have a 1.0 database that you have been using then the schema will need to be migrated to work with the new 1.1 release. This can be done by running the ant target migrate within the sql directory. This target will make changes to the database schema so you will no-longer be able to use a 1.0 release of warfare with this database. Most of the changes have been done to preserve functionality within the application, however a complete change to the way menus are generated means that all previously existing menus will have been deleted.

**Re-Migration of previous 1.1 releases.** If you already have a migrated alpha 1.1 release it may be necessary to re-run the migration script via the psql command to add any further changes into the schema. This can be done by typing `psql vm < migrate.sql` at a unix command prompt. This will produce errors for any changes that have already been made to the schema but should work. (Nasty hack I know but the schema will not be frozen until I enter Beta phase and this was the only way I could find to do the incremental changes without having different migration scripts for each version increment).

**Run the update scripts.** From the sql directory run ant update. This will remove any old views and populate the view manager with a new set of views for this release. The build.xml specifies a number of properties to connect to the database

- `jdbc.driver=org.postgresql.Driver`
- `jdbc.classpath=/usr/local/pgsql/share/java/postgress.jar`
- `jdbc.dburl=jdbc:postgresql://127.0.0.1:5432/vm`
- `jdbc.user=nomad`
- `jdbc.pass=`

These parameters can be overridden on the ant command line with `-D`

#### Example 4. Updateing the database

```
ant -Djdbc.user=barney update
```

## Upgrading Web Application

**Pre-Upgrade Actions.** Take a complete copy of the application directory from the \$TOMCAT\_HOME/webapps directory. E.g. If you installed WARFARE with the default name of data then cp -R \$TOMCAT\_HOME/webapps/warfare databackup where \$TOMCAT\_HOME is the top level directory of your tomcat installation.

**Applying the update.** Change directory into the application directory where you have WARFARE installed. E.g. If you installed WARFARE with the default name of data then cd to \$TOMCAT\_HOME/webapps/warfare where \$TOMCAT\_HOME is the top level directory of your tomcat installation. Switch user to the user that tomcat runs as and run jar -xvf \$DISTHOME/dist/warfare.war where \$DISTHOME is where you unpacked the distribution tar file. If you have added any datasources to the WEB-INF/web.xml you should copy the web.xml from the backup directory you created earlier into the WEB-INF directory.

**Post Upgrade Configuration.** If you want to be able to save template files edited within the View Manager application you will need to configure the location where files should be saved to.

- Locate the viewmanager.properties file within the war file or exploded webapp directory in Tomcat
- Locate the line containing files.savedir
- Replace the value for this property with the directory in which you wish to save files to. The entry recommended is the WEB-INF/classes directory under the directory where tomcat has un-packed the war file. This is the location where most of the templates in the demo reside and therefore any changes will be found. You can specify any directory but that directory must be on the CLASSPATH of tomcat for any saved files to be located by the application framework. An alternative location not on the CLASSPATH can be used if you wish changes to be verified and controlled before being deployed. I.e. save files in /tmp and then when files have been approved copy them manually into the WEB-INF/classes directory. Templates can also be located inside a jar file in the WEB-INF/lib directory to enable stricter version control on deployed templates if desired.

## Modules

**What are Modules.** Modules are a means of distributing and installing applications that run within WARFARE as separate deliverables. This means that they can easily be installed into an existing WARFARE install and be upgraded without major disruption to other applications. The intention is that each module is entirely self contained and can create its' own database and insert its' own views as well as install any content into an existing webapp.

**Installing Modules in to WebApps.** Modules can be installed en-masse by switching to the modules directory and running ant. The default target for ant is to install all modules into the war file located in the dist directory of the release. You can however install all the demos elsewhere by specifying a war file on the command line (e.g. ant -Dwebapp.war=myapp.war). It is also possible to install modules into an exploded war file by specifying the directory and a target of install (e.g. ant -Dwebapp.dir=mywebappdir install). Modules can be installed individually in the same way by changing directory into the module directory and executing the same set of commands.

**Installing the database componants of Modules.** The database componants of a module typically include the SQL statements to create any view and the statements to create the database that this module provides access to. The build.xml within the modules directory can be invoked with the target install-db to install the database com-

ponents of all the modules in this release. You can also only install the modules that you want by changing directory into the module directory you want to install and run ant with a target of install-db. You can override the parameters used to connect to the vm database using the same method documented for uninstalling the main database components of WARFARE.

**Updating Modules in WebApps.** Modules are updated in a similar way in which they are installed. This assumes that you have already installed the module once and will not re-create ant database schemas created by the initial install or update the web.xml file in the destination web-app. Running ant in the modules directory with a target of update-war will update the war file in the dist directory of the release. You may specify an alternate war file on the command line (e.g. ant -Dwebapp.war=myapp.war update-war). You may also update an exploded war file by specifying the directory and the target update (e.g. ant -Dwebapp.dir=mywebapp update). If you update the war file in the release in-order to upgrade an existing warfare install you will need to backup your web.xml from your current release as the update will not update the web.xml in the release war file.

**Updating the database components of Modules.** The build.xml within the modules directory can be invoked with the target update-db to install the database components of all the modules in this release. You can also only update the modules that you want by changing directory into the module directory you want to update and run ant with a target of update-db. Database connection parameters can be overridden as documented above.

**Included Modules.** The new data.war file contains a bare skeleton web application that is configured to run WARFARE. The Project Management demo and the Intranet demo are available as modules that can be installed. If these are not installed then these demos will not be available in the warfare web application. The Project management demo has some links to views within the Intranet Demo, if the Intranet demo is not installed prior to the project management demo then an error will be displayed when the installer tries to create these links. The Project Management demo will however still work it will just mean that these links will not be created. If you later install the Intranet demo you can then update the Project Management demo and these links will be created. The System Configuration Application Matrix (SCAM) is a sample application to keep an inventory of software installations and configurations. A test module is also present, this contains little bits and pieces that I have been testing. The primary purpose of this module is to provide a place to test new features and experimental ideas.

**The Intranet Demo.** The Intranet demo expects there to be a database named intra that should be created prior to installing the database for the Intranet module. There are some additional parameters for specifying the database to install the components of the Intranet demo.

- intra.driver=org.postgresql.Driver
- intra.classpath=/usr/local/pgsql/share/java/postgress.jar
- intra.dburl=jdbc:postgresql://127.0.0.1:5432/intra
- intra.user=nomad
- intra.pass=

These parameters can be overridden on the ant command line with -D

**The Project Management Demo.** The Project Management demo expects there to be a database named pm that should be created prior to installing the database for the Project Management module. There are some additional parameters for specifying the database to install the components of the Project Management demo.

- pm.driver=org.postgresql.Driver
- pm.classpath=/usr/local/pgsql/share/java/postgress.jar
- pm.dburl=jdbc:postgresql://127.0.0.1:5432/pm
- pm.user=nomad
- pm.pass=

These parameters can be overridden on the ant command line with -D

**The SCAM Demo.** The SCAM demo expects there to be a database named scam that should be created prior to installing the database for the SCAM module. There are some additional parameters for specifying the database to install the components of the SCAM Demo.

- scam.driver=org.postgresql.Driver
- scam.classpath=/usr/local/pgsql/share/java/postgress.jar
- scam.dburl=jdbc:postgresql://127.0.0.1:5432/pm
- scam.user=nomad
- scam.pass=

These parameters can be overridden on the ant command line with -D

**The Test Demo.** The Test demo makes use of views that have been defined in the intranet demo which therefore will need to be loaded to make use of the Test demo. The Test Demo also sets up a connection to a JMSQueue and a JMSTopic. If you are interested in the JMS Functionality of WARFARE then you should have a running instance of OpenJMS on the same server as WARFARE is installed on. The views make use of Queues and Topics that have been defined by the default install of OpenJMS so you shouldn't need to do any additional configuration to OpenJMS.