

INFOGLUE CMS
(www.infoglue.org)

Review

1) Maintenance: difficulty=EASY/MEDIUM

InfoGlue is a CMS application based on the MVC model.

Very clear separation of contents / layout / logic.

Static contents and graphics can be created / updated very easily: contents (texts, images, documents) can be created/updated by anyone, while website layout and graphics can be created/updated easily by web designers (HTML is sufficient).

Combining layout, navigation and contents (i.e. managing how/where the contents/information are displayed) is done through templates, which require some knowledge of JSP and some understanding of the InfoGlue API classes and taglibs.

2) Templating / web design customization: quality=VERY GOOD, difficulty=MEDIUM

Templating language: either JSP or Apache velocity.

Templates in InfoGlue are considered as “contents” but there is a dedicated “content type” (templatePage) for them and they differ sensibly from other types of contents (texts, formatted text pages with images, images, documents...). Templates can contain HTML, JSP, JSTL tags and other custom tags and they typically contain “slots” where the created “normal” contents (and links to them) can be inserted. (Slots can also contain other templates, which allows to “distribute” the layout across small basic modules that can be than combined in different ways, making it very easy to change the whole layout of a website without changing all the templates).

Using JSP and/or JSTL, it is very easy to re-use an existing website layout or existing web pages.

Using JSTL tags and InfoGlue custom tags allows for very clean template pages that are very powerful in managing both the static layout and the dynamic positioning of contents and navigation items.

InfoGlue templates are very re-usable as they can accept parameters and therefore use the same layout logic to produce different displays.

3) Website structure (storyboard) implementation: VERY GOOD

The structure management is clearly separated and independent from the content management: the web structure is made of sitenodes that can “include” both templates (presentation) and contents, or better yet include templates and pass them parameters (properties) so that they include the correct contents.

The website structure and navigation logic can be easily modified without affecting the contents and the graphics.

4) Portability: quality=GOOD, difficulty=EASY

Fast and easy to port to a new environment.

(There is a built-in import/export functionality in InfoGlue, but it fails in many cases)

The export/import is easy and smooth if done between two very similar server environments (particularly: same Tomcat version, same core libraries versions, same MySql version) and

it only requires:

- a) a **dump of the database** (see notes below: Portability – Database dump)
- b) an **InfoGlue setup** on the new server (if a dedicated instance of InfoGlue is not working there already)
- c) some minor environment-related changes to the .properties files

But an export/import can also be tried between different server environments providing some conditions are met (see notes below: Portability – Different server environments):

5) **Compliance with standards / best practices: very GOOD**

- a) Cross-platform. Language: Java. Server environment: any J2EE servlet container.
- b) Can use different databases (MySQL, Oracle, PostGres...)
- c) Uses persistence (Hibernate)
- d) Uses caching
- e) Fully exploits XML:
 - Contents are stored in the database as XML
 - Content type definitions are stored in the database as XML schemas for the type of content defined
- f) Compliant with the JSR 168 standard for portlets

6) **Installation: EASY**

- a) “Wizard” setup: an easy procedure (both GUI version and command line version) that asks for configuration parameters (Tomcat path, database type, database path, database user etc.) and sets everything up (a batch procedure executes all the steps listed under (b))
- b) Manual setup:
 - Create a dedicated db
 - Deploy the WAR files (at least the CMS application and the deliver working application, but commonly also the deliver live application) under Tomcat
 - Create context files under TOMCAT_HOME/conf/Catalina/localhost/ with the same names as the applications
 - Modify the localConfigs/database.xml file providing the database settings
 - Modify the webapp .property files (cms.property under the CMS application, deliver.cms under the other applications) providing the correct environment settings (paths, server etc.)
 - Copy pluto files from installation directory to: /usr/share/tomcat5/shared/lib/

NOTES

1) **Portability:**

- SIMILAR SERVER ENVIRONMENTS – DATABASE DUMP:

Importing a dump created on a system where `lower_case_table_names` is set to 1 (or on Windows) [*best solution: before dumping on Windows, start MySQL with parameter – `lower_case_table_names=2`: this will dump table and field names with correct case; you can also try: first setup InfoGlue and then import the db: the installer stops if tables have lower case names*].

- to create a new database:

```
mysqldump –user=... --password=... egfar_ig_old
```

```
mysql -user=... --password=... egfar < egfar_ig_old
```

- to replace an existing database:

```
mysqldump -user=... --password=...
```

```
-add-drop-table --ignore-table=OS_PROPERTYENTRY egfar_ig_old
```

```
mysql -user=... --password=... egfar < egfar_ig_old
```

(Important: if InfoGlue is already installed and working on a system, in order to import a database from another rinstallation without changing the application settings (particularly the host and path information), *exclude the OS_PROPERTYENTRY table from the dump.*)

(Important: execute a query on the “cmDigitalAsset” table in order to *change the absolute local base path of digital assets*, which – in a standard installation - is the TOMCAT physical path: e.g. “UPDATE `cmDigitalAsset` SET `assetFilePath` = replace(`assetFilePath`, '/usr/local/jakarta-tomcat', 'C:\\Programmi\\Apache Group\\Tomcat 5.5')”)

(Other useful queries in order to reset the environment – if needed: “DELETE FROM `cmPublicationDetail`”; “DELETE FROM `cmPublication`”, “UPDATE cmContentVersion SET stateId = 0”)

- DIFFERENT SERVER ENVIRONMENTS:

An export/import can be tried between different server environments providing some conditions are met:

a) A **dump of the database** can be successfully imported to the new environment.

In the case of MySQL, the problems that could arise are: importing a newer MySQL version dump to an older MySQL version [*solution: add –skip-opt to mysqldump command or –compatible=MYSQL40; but be careful: mysqldump from MySql 5.0 with –compatible=MYSQL40 doesn't include the auto_increment information, which must be added manually!*];

b) (For import to an existing InfoGlue environment) the **Tomcat version** in the new environment matches the Tomcat version of the InfoGlue import (InfoGlue installs different libraries according to the Tomcat version and code written in the components will probably use those libraries)

c) The **dependancies** (classes, libraries, taglibs, servlet container) for the java code used in the templates written for the repository are all present in the correct version in the new environment and. This means that if for instance the repository was created in a Tomcat 5.0 environment using JSTL 1.1 and you are trying to port it to a Tomcat 4.x environment you will have to replace all the taglibs import directives to point to the JSTL 1.0 taglib urls and change code accordingly.