What's new in Telosys 1.0.0 ?

1/ DAO registries :

The DAO instances can be stored and managed in a DAO registry.
A DAO registry acts as a lightweight container and provides the DAO instance associated with a given bean object.
The DAO are managed as “mono-instance” objects.
The registry provides a single (and shared) instance for a given bean type.
There’s a DAO registry for each database

If the application uses a specific DAO registry, its class name can be defined in the “telosys.properties” file :
daoregistry = demo.env.DAORegistryDB${DBID}
A dynamic DAO registry can be used to avoid a specific registry implementation.
In this case, the DAO class is determined dynamically by using a class pattern defined in the “telosys.properties” file :
daoclass = demo.dao.db${DBID}.${BEANNAME}DAO

2/ New servlet to call DAO in REST mode :

DAO are now accessible directly via http in “REST mode”.
Example :
http://localhost:8080/myapp/dao/1/Agency/load.xml?Code=2
Uses the DAO associated with the bean class “Agency” to load the bean from the database ”1”
according with the given primary key parameters (here “Code”) and returns the response in XML format.
This feature is just a servlet, thus can be activated or not depending on the servlet declaration in the web.xml.

3/ “ScreenManager” replaces “ScreenDataAccessor”

The old “ScreenDataAccessor” interface has been renamed to “ScreenManager”
The “StandardScreenDataAccessor” class has been renamed to “StandardScreenManager”
The “StandardScreenDataAccessor” still exists for backward compatibility (it’s just a void class extending “StandardScreenManager”)

NB : there’s just <u>one method impacted</u> by this renaming :
ScreenContext.**getScreenDataAccessor()** doesn’t still exist.
It has been replaced by ScreenContext.**getScreenManager()**
which returns a ScreenManager (instead of ScreenDataAccessor)

4/ XML mapper class name pattern

A more powerful class name pattern is now available for XML mappers.
It can be specified in the “telosys.properties”
Example :
mapperclass = demo.xmlwrapper.${BEANNAME}XmlWrapper
The “old fashion” class definition (package + suffix) is still active for backward compatibility.

5/ More flexible configuration files loading

The origin of each configuration file can be specified in “telosys.properties”.
It allows different types of loading for each file (Web App resource, File System and Class Path)
The origin can be specified by adding “.origin” to the property name.

Examples for “ScreenConfFile” :

Loading from File System :
```
ScreensConfFile = /mydir/aaa/bbb/conf/screens.xml
ScreensConfFile.origin = FILESYSTEM
```

Loading as Web App resource :
```
ScreensConfFile = /WEB-INF/conf/screens.xml
ScreensConfFile.origin = WEBAPP
```

Loading by Class Path :
```
ScreensConfFile = screens.xml
ScreensConfFile.origin = CLASSPATH
```

6/ New StandardScreenProcedures class

StandardScreenProcedures is a new abstract class designed to be specialized for each screen (just like StandardScreenManager and StandardScreenTriggers).

7/ ScreenApplication and Servlet Context

The ScreenApplication has been detached from the ServletContext
ScreenApplication and ServletContext are accessible directly by “**getScreenApplication()**” and “**getServletContext()**” in classes which extends StandardScreenManager, StandardScreenTriggers and StandardScreenProcedures.

In other cases, they can be retrieved by
```
ScreenApplicationManager.getScreenApplication() ;
ScreenApplicationManager.getServletContext();
```