

Pattern for application services creation and integration

Do It The jAPS Way

**Eugenio Santoboni, AgileTec s.r.l. <e.santoboni@agiletec.it>
William Ghelfi, Tzente s.r.l. <w.ghelfi@tzente.it>
Matteo Minnai, Tzente s.r.l. <m.minnai@tzente.it>**

Pattern for application services creation and integration

by Eugenio Santoboni, William Ghelfi, and Matteo Minnai

Publication date 2010-03-24

Copyright © 2010 Tzente s.r.l.

Legal Notice

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the Appendix entitled "GNU Free Documentation License".

The copyright holders make no representation about the suitability of this Document for any purpose. It is provided “as is” without expressed or implied warranty. If you modify this Document in any way, identify your resulting Document as a variant of this Document.

Table of Contents

1. Scope of the document	1
Introduction	1
Target audience	1
Prerequisites	1
Resources	1
2. Introduction	2
3. Architectural model of the jAPS 2.0 framework	4
Portal View	5
Administration View	5
Distribution of the components of the system layers	5
4. How to create a jAPS2 service	6
Business Layer and Data Layer	6
Implementation	6
JUnit tests for the Data Layer and Business Layer	8
How to extend existing jAPS-Managers	9
Presentation Layer - Administration Area	10
Implementation	10
Internationalization and localization	12
Testing Actions with JUnit	12
Creation of the jsp for the Administration area	13
Creation of a new voice in the Administration Area	13
Modify the existing Administration area interfaces	14
A. GNU Free Documentation License	15

List of Examples

4.1. Package Manager Card	6
4.2. name of the Manager Card	6
4.3. Manager Card class	6
4.4. Implementation of the init method	7
4.5. Definition of the name of a manager through a constant	7
4.6. Methods signature	8
4.7. DAO Class	8
4.8. Full definition of the Manager	8
4.9. Extension of the User Manager using the same id of the core service (UserManager)	10
4.10. Create a package for the Action classes for the Card service	10
4.11. Package containing the class Action for Card management	10
4.12. Definition of the action bean	11
4.13. Action definitions in the file card.xml	11
4.14. Example of actions definition in the file PortalExample.xml	12
4.15. jsp of the card manager service	13
4.16. Declaration of the Menu	13
4.17. Redefinition of the "Pages Tree" interface	14

Chapter 1. Scope of the document

Introduction

The aim of this document is to give a detailed description of the architectural model of jAPS 2.0 and the steps to follow to create a new application service.

Target audience

This document is for developers aiming to build a new *Application Service* su jAPS 2.0.

Prerequisites

In order to take the maximum advantage of the present document, it is necessary to have a basic knowledge about: the Java platform, the Eclipse IDE, the Apache Tomcat servlet container, the PostgreSQL DBMS and the jAPS 2.0 framework.

Resources

Additional informations may be obtained through the following mailing-lists:

- <japs-devs@lists.sourceforge.net>, focused on developers
- <japs-users@lists.sourceforge.net>, focused on final users

Is it also possible to refer to the documentation found in the:

jAPS 2.0 Project - Developer Website [<http://dev.japsportal.org/>]

Chapter 2. Introduction

We define the jAPS Manager as a part of the jAPS Core which implements a basic system functionality. A jAPS Manager is also the main handler of that particular functionality.

The main services belong to one of the following groups:

Basic services:

- *AuthenticationProviderManager*: authenticator service..
- *BaseConfigManager*: configuration service. Load the configuration parameters from system database, making it available to the invoker.
- *CacheManager*: cache handler service.
- *CategoryManager*: category handler service.
- *ControllerManager*: this service supervises the execution of a request coming from the client.
- *GroupManager*: group handler.
- *I18nManager*: this service returns the localized strings upon request
- *KeyGeneratorManager*: this service superintends the generation of primary keys
- *LangManager*: this service handles the various languages of the system
- *NotifyManager*: event notification dispatcher service
- *PageManager*: page handler
- *PageModelManager*: this service handles the various page models
- *RoleManager*: role manager
- *ShowletTypeManager*: this service manages the showlets (ShowletTypes) types defined in the system
- *UrlManager*: this manager creates a URL to page of the Portal from essential informations.
- *UserManager*: account manager

CMS services (served by the jACMS plugin):

- *ContentManager*: contents manager
- *ContentPageMapperManager*: this service manages the map of the contents published in the pages of the portal
- *LinkResolverManager*: this manager resolves the symbolic links
- *ResourceManager*: resources (audio, video, images etc.) handler
- *SearchEngineManager*: this service creates the indexes of all the objects which will be later parsed by the search engine.

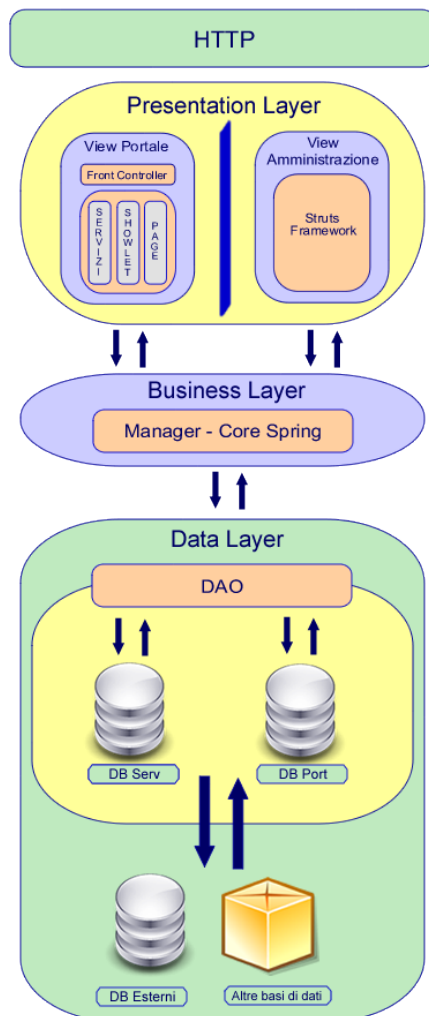
The services defined in the system are initialized during system start-up through the Factory provided by the Spring Framework.

It's important to underline that each service has one and only one instance. The invocation of a service can be obtained in either two ways: through the "Dependency Injection" technique favored by Spring

or using the appropriate elements of the system like `ApsWebApplicationUtils`. Every jAPS manager is described through a specific interface and every object class access a service *always* using the appropriate interface, *never* invoking the class directly.

The manager (o jAPS Manager) is the only linking point between the system data -whatever their origin is- and the functionality which use them. An example of service is the `PageManager` which manages the tree of the portal pages. Every operation involving the pages, such as addition, removal, displaying and so on is handled by the `PageManager`.

Chapter 3. Architectural model of the jAPS 2.0 framework



Architectural model of the jAPS 2.0

To fully understand this document it's necessary to describe the architectural model of jAPS: jAPS is mainly composed by 3 layers:

- *Data Access Layer*: It is composed by all the elements which superintend the Persistence Layer. The main component are the DAO classes (Data Access Object) which are the only linking element between the framework and the data sources (Database, Filesystem, LDAP service directory etc.)
- *Business Layer*: This is the core of the system. Here the concept of *jAPS service* as manager of every macro functionality, takes place. This layer is built upon the Spring Framework, whose listener, during the system start-up, initializes all the services and injects them in the web application context as beans. The Business Layer utilizes the Data Access layer to get the data needed, gives to the higher layer (the Presentation layer) the elements to display and supports it in the execution of actions.
- *Presentation Layer*: The aim of this layer is to build the graphic interfaces which represent the mean through which the users interact with the system. This layer gives a pure View layer (that is, a jsp without any business logic) and a "slim" controller (which checks the consistency of the data submitted and serves the data produced); both of them provide support to the layer below, the Business Layer. In the jAPS framework this layer is divided in two parts: the Portal View (referred

as Front-end) and the Administration View (Back-end). These views, which differs by functionality and architecture, are completely independent from each other.

Every application service must be developed in the total respect of the architectural schema above, placing every part in the right layer. The presence of the elements of the new service in all of the three layers depends on characteristics of the service itself. The typical service which needs the usual addition, removal, editing and searching operations will have elements in each layer - take as reference the "Personal Card" management service explained further in this document and found implemented in the "Portal Example" demo. Moreover the "Personal Card" service has customized elements in the "View" layer, both in the Portal and the Administration area. The LDAP plugin, on the contrary, has elements in the Business layer only.

Portal View

This is the part of the presentation layer where the results of the queries to system services are displayed mainly through the Showlets. Showlets are the preferred method to use to make the system services interact with users. The tasks of the Portal View are to provide services based on the current user permissions (every element of the Portal Layer incorporates the rules which govern the access to services) and to serve content as fast as possible (using content caching mechanism). The portal view is handled by a specialized servlet (ControllerServlet) whose primary target is to invoke a precise succession of services (coherency of the URL, user privilege checks etc) which will finally result in the rendering of the requested page.

Administration View

This is the area reserved for administration of the various elements of the Portal (Pages, Contents, Resources etc.) whose access is reserved to a restricted pool of users. The view of the Administration Area, comprehensive of the controller logic) has been completely redesigned: the reference framework, firmly tied to the Spring framework, is now Struts2. The View has been modified to meet the (Italian) Public Administration requisites of accessibility - taking as a firm point the respect of *all* the W3C standards.

Distribution of the components of the system layers

The source files, with the exclusion of the test packages and certain supporting elements (eg. the static resources and the templates directories etc.), are enclosed in two packages:

- `com.agiletec.aps` : here can be found all the elements of the Data Access Layer, Business and presentation Layer (the last *limited* to the Portal View only)
- `com.agiletec.apsadmin` : this package contains all the elements need to manage the presentation layer of the Administration View

A similar division exists in the directory `WEB-INF` of the web application: here are contained, including the supporting folders, the usual `aps` (which contains all the `jsp` and the `tld` files of the Portal View layer) and `apsadmin` (containing all the `jsp` files belonging to the Administration Area).

Chapter 4. How to create a jAPS2 service

The following paragraphs explain in detail how to create a new service in the jAPS 2.0 framework.

The main objective of the present document is to allow the jAPS-Developers a fast development of new services to *integrate* with existing ones, without modifying the Base Core sources (java classes, jsp files, configurations, etc).

Business Layer and Data Layer

During the process of the creation of a new service, the following procedure starts from the implementation of the Business and Data Layer of the new functionality.

Active elements: the classes involved are `<NAME_OF_THE_HANDLED_OBJECT>Manager` (the name of the service) which must extend the `AbstractService` class. In a similar manner, if the DAO classes are needed, the `<NAME_OF_THE_HANDLED_OBJECT>DAO` must extend the `AbstractDao` class.

Implementation

Create the package, external to the core classes, respecting the same schema used by the core.

Example 4.1. Package Manager Card

If the new service is called *Card*, the resulting name will be `it.projectname.aps.system.services.card`.

Create an interface, namely *Firma del Servizio*, which respects the following syntax `I<NAME_OF_THE_HANDLED_OBJECT>Manager`.

Example 4.2. name of the Manager Card

`ICardManager`

This interface includes all the public methods (and the costants, if present) of the service which will be accessible from the outside. *Every use of the implemented methods must happen through the invocation of this interface.*

Create the class of the service `<NAME_OF_THE_HANDLED_OBJECT>Manager` which extends in turn the `AbstractService` and implements the methods declared in the interface seen before.

Example 4.3. Manager Card class

```
public class CardManager extends AbstractService implements ICardManager {
    ....
}
```

Take care to implement the `init` method of the abstract service (which allows the correct initialization of the service), and the methods declared in the interface properly

Example 4.4. Implementation of the init method

```
/**
 * Service initialization
 */
public void init() throws Exception {
    .....
}
```

Add, in the class (or interface) <PROJECT_NAME>SystemConstants contained in a sub-package `aps.system` of the project, the constant <NAME_OF_THE_HANDLED_OBJECT>_MANAGER which uniquely identifies the name of the service within the project.

Example 4.5. Definition of the name of a manager through a constant

```
public interface MyProjectSystemConstants {

    public static final String CARD_MANAGER = "CardManager";

    .....

}
```

Add the service in a new configuration file, which will be later parsed by Spring. The configuration files must be inserted in a directory under the `/WEB-INF/<PROJECT_NAME>/conf/` following the same pattern used for the configuration files of the core.

The new Manager must be inserted in the Spring context using a syntax similar to the one shown below:

```
<bean id="CardManager"
      class="it.projectname.aps.system.services.card.CardManager"
      parent="abstractService" >
</bean>
```

where the id is the value of the constant defined previously.

Important

Care must be taken in the definition of the bean since it must *not* match any other existing ids in the system unless we intend to extend an existing service on purpose.

Make the system aware of the new service by instructing Spring to load every xml file in the configuration directories of your service. This is typically done editing the `/WEB-INF/web.xml` and adding to the xml attribute `param-value` of the parameter `contextConfigLocation` the file pattern string `WEB-INF/<PROJECT_NAME>/conf/**/*.xml`. This definition must be added in the last position. The same pattern must be inserted in method `getSpringConfigFilePaths` of the class `test.com.agiletec.aps.ConfigUtils`. This class is used to setup the proper environment for the test suites; again, the definition must be placed in the last position.

If the new service uses a DAO (Data Access Object) so that it adds new elements in the Data Layer, the first thing to do is to crate an interface *Firma del DAO* using this declaration

I<NAME_OF_THE_HANDLED_OBJECT>DAO and add, in the class which implements that interface, an instance variable of the same type of the newly created one. This variable must have both getter and setter, with the former being rigorously public.

Example 4.6. Methods signature

```
public void setCardDao(ICardDAO cardDao);
protected ICardDAO getCardDao();
```

Create the DAO class <NAME_OF_THE_HANDLED_OBJECT>DAO which implements the interface just created. If we are willing to use a standard JDBC, the class DAO just created must extend the class AbstractDAO.

Example 4.7. DAO Class

```
public class CardDAO extends AbstractDAO implements ICardDAO {
    ....
}
```

Inject the new DAO in the bean of the service previously described.

Example 4.8. Full definition of the Manager

```
<bean id="CardManager"
    class="it.projectname.aps.system.services.card.CardManager"
    parent="abstractService" >
    <property name="cardDao">
        <bean class="it.projectname.aps.system.services.card.CardDAO" >
            <property name="dataSource" ref="dataSourceBeanName" />
        </bean>
    </property>
</bean>
```

NOTE: inject the datasource having care to choose the proper reference between the default "portDataSource" or "servDataSource" (which always exist in a jAPS installation) and the new data sources eventually created for the new service.

jUnit tests for the Data Layer and Business Layer

Every service in the DAO *must* be tested in its public methods. In other words it's necessary to:

- create a java class named <PROJECT_NAME>ConfigUtils (in the package test.it.projectname) which extends the class ConfigUtils; the methods getSpringConfigFilePaths and closeDataSources must be extended as well. The former provides the path for the configuration files of the new service needed by Spring, the latter handles the database connection closure of the new datasources.
- create a java class <PROJECT_NAME>BaseTestCase (in the package test.it.projectname.aps) which extends the class

`test.com.agiletec.aps.BaseTestCase`. Override the method `getConfigUtils` so that it returns an instance of `<PROJECT_NAME>ConfigUtils` (that is, the class previously created).

- Create the test classes `Test<NAME_OF_THE_HANDLED_OBJECT>Manager` and, if needed, the `Test<NAME_OF_THE_HANDLED_OBJECT>DAO` in the package `test.it.projectname.aps.system.services.<NAME_OF_THE_HANDLED_OBJECT>`. Such classes must extend the classes previously created. Remember to test all the public methods of the new service.

To check a service we have obviously to invoke it in every test class; this is done with the following code:

```
ImyServiceManager myServiceManager =
    (ImyServiceManager)
    this.getService(MyProjectSystemConstants.MY_SERVICE_MANAGER);
```

To test a DAO is necessary to create it as Spring does, passing to it the `datasource` and as every requested bean.

```
DataSource dataSource = (DataSource)
    this.getApplicationContext().getBean("dataSourceName");
MyServiceDAO myServiceDao = new MyServiceDAO();
myServiceDao.setDataSource(dataSource);
```

Two databases, namely `jAPStestPort` and `jAPStestServ`, are provided for testing purposes. They reflect their "production" counterparts, the `jAPSPort` and `jAPSServ`. If the new service requires additional databases they all must have a test and a production version as well.

For every method of the service to test a corresponding method in the appropriate test class must be created:

```
public void testNomeMetodoDaTestare() {
    .....
}
```

When creating test methods it's important to plan the restore of the data in the state they were prior the execution of the test(s), whatever the result is. This assures the coherence and the correctness of the following test. You don't want a failed test to cause a succession of failures in different classes which previously were just fine.

How to extend existing jAPS-Managers

If the newly born service alters existing managers (by either integrating or modifying functionalities) you are strongly advised *to avoid modifying the core!* Create inside the package `it.projectname.aps.system.services` of your project, a new manager which extends the existing one. In the Spring configuration file of your service the id of the service must perfectly match the one of the existing service (of the core of jAPS) that we are going to extend.

Example 4.9. Extension of the User Manager using the same id of the core service (UserManager)

```
<bean id="userManager"
  class="it.projectname.aps.system.services.user.UserManager"
  parent="abstractService" >
  <property name="userDAO" >
    <bean class="it.projectname.aps.system.services.user.UserDAO">
      <property name="dataSource" ref="servDataSource" />
    </bean>
  </property>
  <property name="configManager" ref="BaseConfigManager" />
</bean>
```

In the previous example, the new bean id `userManager` substitutes, having the same name, the one of the core of jAPS. Remember to insert all the properties found in the declaration of the core bean in the new one.

Presentation Layer - Administration Area

The most of the Application services will need an administration interface. In the core of jAPS, the class which superintend the interface mechanism are all grouped inside the package `com.agiletec.aspadmin`. This package in turn contains other sub packages organized (and separated) by functionality; each serves a well determined function whose controls are displayed in the Administration Area. The new service must present the sources to manage the back-end interfaces developed following the same structure used in the core.

Implementation

Create the package -outside the Core path!- respecting the schema used in jAPS 2.0, as stated earlier.

Example 4.10. Create a package for the Action classes for the Card service

Suppose to have the need to create Actions to handle the *Cards* (defined in the homonym class): the resulting name of the package will be: `it.projectname.apsadmin.card`.

Create the java interface *Firma delle Action* which respectes the syntax `I<NAME_OF_THE_HANDLED_OBJECT>Action`.

Example 4.11. Package containing the class Action for Card management

`ICardAction`.

This interface presents all the public methods and eventually the constants, which will be implemented in the service class. Every method presented in the interface is an action which can be executed.

If our service provides some search function of the object handled by the service we have to create an additional java interface, namely `I<NAME_OF_THE_HANDLED_OBJECT>FinderAction`. This is the gate to the finder action class.

Create the action class named `<NAME_OF_THE_HANDLED_OBJECT>Action` which extends the `BaseAction` and implements the interface above. If needed, create the finder action class which manages the search operations.

Any action class must have a corresponding Spring configuration file; the syntax to use is close to the one shown in the example below.

Example 4.12. Definition of the action bean

```
<bean id="cardAction" scope="prototype"
      class="it.projectname.apsadmin.card.CardAction"
      parent="abstractBaseAction" >
</bean>
```

Important

The scope of the bean of the action classes must be "prototype" and care must be taken when defining the bean: it must not match any other bean id of the core, unless we are extending an existing service, as we have already seen.

Insert the configuration file in a directory named /WEB-INF/<PROJECT_NAME>/apsadmin/conf/.

Once again, make spring aware of the new action by adding the following string WEB-INF/<PROJECT_NAME>/apsadmin/conf/**/**/*.xml in the xml attribute param-value of the parameter contextConfigLocation located in the file /WEB-INF/web.xml. This definition must be placed in the last position.

Create, at the same level of the interfaces and classes, a xml file which contains the definitions of the actions previously implemented. These definitions follow the Struts2 rules; there is one definition for every action which can be triggered by users from the user interface.

Example 4.13. Action definitions in the file card.xml

```
<struts>
<package name="portalExample_do/Card"
  namespace="/do/Card" extends="japs-default">
  <action name="list" class="cardFindingAction">
    <result type="tiles">admin.Card.list</result>
    <interceptor-ref name="japsDefaultStack">
      <param name="requestAuth.requiredPermission">superuser</param>
    </interceptor-ref>
  </action>
  .....
  <action name="edit" class="cardAction" method="edit">
    <result type="tiles">admin.Card.entry</result>
    <interceptor-ref name="japsDefaultStack">
      <param name="requestAuth.requiredPermission">superuser</param>
    </interceptor-ref>
  </action>
  ....
</package>
</struts>
```

Note: the name of the Struts2 package must present as prefix the name of the project.

Make use of the stack interceptors defined in the file struts.xml:

- japsDefaultStack: this is the default for the actions of the Administration view which need specific permissions to be executed (eg. check for the user permission when accessing the Administration area). This stack does not enforce validation or range check of the submitted

parameters. This stack needs the explicit declaration of the permission needed to execute the action in the `requiredPermission` tag.

- `japsValidationStack`: extension of the `japsDefaultStack` with the addition of validation checks.
- `japsFreeStack`: This stack is to be used for actions both internal and external to the Administration area, which require neither permission nor validation checks.
- `japsFreeValidationStack`: Extension of the `japsFreeStack` stack with validation check enabled.

Create at the same level of the java interface and action classes the appropriate xml files to define the kind and the number of validation checks to perform. These validation files follow the Struts2 syntax for the validation.

Create a new `<PROJECT_NAME>-struts.xml` in the root of the source files or, in other word, in the same level of the directory where the `struts.xml` resides. This file must contain all the references to the configuration files of the new actions of the project.

Example 4.14. Example of actions definition in the file `PortalExample.xml`

```
<struts>
  <include file="it/myprojectname/apsadmin/card/card.xml"/>
</struts>
```

The xml file containing the definitions of the various actions of the project must be declared in the parameter `Struts2Config` within the `/WEB-INF/web.xml` file. As always the definition must be inserted in the last position.

Internationalization and localization

The property files reside in the same directory of the newly created Action classes; these files provide support for the Internationalization (i18n). These file must follow strictly the rules as specified in documentation released in the Struts2 framework website.

In the properties files must be inserted not only the static labels of the jsp files of the user interfaces, but all the labels correlated to the validation support. These labels must be provided for both English and Italian language (the file `package_it.properties` and `package_en.properties` serve this purpose).

As for the id of the service beans, the keys of the labels must not match any of the common resources keys contained in the files `global-messages_en.properties` and `global-messages_it.properties`.

To avoid problems, you are encouraged to subdivide the label in the following groups:

- on menu basis
- per titles (h1 e h2)
- static strings of the jsp files
- string used by the validation files

Testing Actions with jUnit

Create the proper environment and the classes to test new newly created actions. In other words:

- create a java class named `<PROJECT_NAME>ConfigUtils` (or use the class used to test the manager methods) which extends the class `ConfigUtils`; the methods `getSpringConfigFilePaths` and `closeDataSources` must be extended as well. The former provide the path for the configuration files of the new service needed by Spring, the latter handles the database connection closure of the new datasources.
- Create a java class `<PROJECT_NAME>ApsAdminTestCase` (inside the package `test.it.projectname.apsadmin`) which extends the class `test.com.agiletec.apsadmin.ApsAdminTestCase`, then override the method `getConfigUtils` (so that it returns an instance of the class previously created) and the `setInitParameters` (so to load the definition of your actions and all those defined in the same level of the `struts.xml`)
- Create the Action classes named `Test<NAME_OF_THE_HANDLED_OBJECT>Action` which extends the class of the previous step. If needed create the test class of the action which uses the search engine of your service; always test all the Actions!

Creation of the jsp for the Administration area

All the jsp files composing the user interfaces are located in the directory `/WEB-INF/<PROJECT_NAME>/apsadmin/jsp/`.

Example 4.15. jsp of the card manager service

- `cardFinder.jsp`: this is the interface for the search card service; the search itself is handled by the Action class `CardFinderAction`.
- `entryCard.jsp` generates the Card add/remove interface, handled by the Action class `CardAction`.

The (jsp) interfaces must be declared inside the template called `main.layout` in the file `/WEB-INF/apsadmin/tiles.xml` which specifies the configuration of the pages being invoked as a result of the action. Such configuration must obey the rules of Tiles2, a Struts2 plugin.

Define a new Tiles configuration file for the pages, `<PROJECT_NAME>-tiles.xml` inside the folder `/WEB-INF/<PROJECT_NAME>/apsadmin`. The pages must extend the `main.layout` and the single ids represent the result (in the form of tiles type) of every action. The tiles configuration must be declared within the parameter `org.apache.tiles.impl.BasicTilesContainer.DEFINITIONS_CONFIG` of the descriptor file `web.xml` of the web application. Again, it must be placed in the last position.

Creation of a new voice in the Administration Area

To add a new element in the Plugin menu create in the directory `/WEB-INF/<PROJECT_NAME>/apsadmin/jsp/common/template/` a file named `subMenu.jsp` which contains the new menu item referring to the new application service (a plugin, in this case) Then create a new bean (a Spring Object) with id `<SERVICE_NAME>SubMenu` which refers to the class `PluginSubMenuContainer`; this class has a property called `submenuFilePath` whose value is the path of the `subMenu.jsp` just created.

Example 4.16. Declaration of the Menu

```
<bean id="cardPluginSubMenu"
  class="com.agiletec.apsadmin.system.plugin.PluginSubMenuContainer" >
  <property name="submenuFilePath">
    <value>/WEB-INF/demo/apsadmin/jsp/common/template/subMenu.jsp</value>
  </property>
</bean>
```

Following carefully these steps the new menu item will be included in the Plugin menu in the administration area.

Modify the existing Administration area interfaces

If the Application service is going to modify existing interfaces for any reason (eg. integration of new modules, link or whatever) you are advised to *avoid any modification of the Core interfaces*. Create instead in the tiles configuration file `<PROJECT_NAME>-tiles.xml` a new definition with the same name of the core interface to override. So the element to modify is simply rewritten from scratch.

Example 4.17. Redefinition of the "Pages Tree" interface

copy the following definition in the files `<PROJECT_NAME>-tiles.xml`:

```
<definition extends="main.layout" name="admin.Page.viewTree">
  <put-attribute name="title" value="title.pageManagement" />
  <put-attribute name="extraResources"
    value="/WEB-INF/apsadmin/jsp/common/template/extraresources/pageTree.jsp" />
  <put-attribute name="body"
    value="/WEB-INF/<PROJECT_NAME>/apsadmin/jsp/page/pageTree.jsp" />
</definition>
```

where the `admin.Page.viewTree` is the id of the interface of the page tree handler.

The path of the jsp must be the same of the jsp files of the interface to extend *with the sole exclusion* of the root directory of your project.

Nei limiti del possibile, è sconsigliato utilizzare questa tecnica; nel caso di inserimento nuove funzionalità che integrano alcune preesistenti, è consigliato utilizzare la tecnica dei SubMenu dei Plugin per creare gli EntryPoint della funzionalità. Whenever it's possible please follow these directions; if the new service adds some new functionality extending the existing ones, a good practice is to use the submenu technique used for the plugins so to create the entry point for the new service.

Appendix A. GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. [<http://www.fsf.org/>]

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties — for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See Copyleft [<http://www.gnu.org/copyleft/>].

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.