Graffito Site v.1.0-a1-dev

Project Documentation

The Apache Software Foundation 02 August 2005

Table of Contents

1.1 Overview

1.2 Graffito Features

Graffito Features

Here is a summay of the main Graffito features.

CMS Repository Services

Creating flexible CMS services which can transparently support different CMS repositories. By this way, your application can access to heterogenous content store. It should be possible to support CMS repositories which are providing a Java API or a WEBDAV layer or the newest JSR 170).

Features	Description
Model Service	Manage all CMS objects (documents, folders, ...). Create, retrieve, update and delete content. Store links between contents.Provides an infrastructure for recording information about content.
Lock Service	Manages concurrency access to the CMS objects. Enable remote, collaborative authoring of any media type. Resources can be locked as "exclusive" or "shared".
Version Service	Manages content versionning. Infrastructure for versionned resources : check in/ check out with comments and metadata; versionning graphs and histories; browse and retrieve old versions.
Security Service	Manages "fine grained" permissions across the complete content tree. Each CMS service implementations has to be integrated with the Jetspeed security services or another security service; Grant permissions are based on the association between a content reference,an action type (eg. : read, write) and an actor type (user, group or role).
Search & indexing Service	Search documents and folders via full text search and/or metadata criteria.; Possibility to implement a new query language. Minimal query language can be : boolean search for property existence or value; or search for substring in a resource content. Indexing documents can vary in function of the language and the media type.
Workflow Service	Manages the different publication activities. Content can be attached to a workflow, which is a series of tasks what must be completed. Workflow can consist of any number of manual tasks, performed by users and groups, as well as automated tasks which are performed. Workflow tasks can be assigned to groups, in which case any member of a designated groups may approve, edit, or re-route content as appropriate. The default workflow can be customized and linked for reuse in other projects. Users are notified of new tasks via email notifications or notification portlet.
Event Service	Triggers events when content is created, update,deleted or retrieved; Events can be customized in order to index the document or makes auditing, ...
Integration Service	Manages external repositories consolidated into one virtual content tree. The user doesn't worry about the physical content location.

Graffito Content Store

In short term, we want to support the following content backend :

A simple DB schema based on OJB. WEBDAV server like Slide. JCR based repository like JackRabbit.

CMS portlets

Accessing thoses CMS Services across predefined JSR-168 portlets. Providing or integrate CMS editor portlets in order to update the CMS repository content. Managing the content publication across a customized content workflow.

So, we plan to build portlets for the following usages :

1.Administration : managing the content tree (add, delete folders & documents, ...), setting permissions, managing external server references, ...

2.Content browser & viewer.

3.Content workflow/notification.

4.Content editor.

5.Specialized portlets like news, forum and many others which have to store their content into a content repository.

Other Features

Support for any media type (XML, HTML, binaries like Ms Word, ...).

Applications like News, Forums, ...

Tools like scheduler, report generator, asynchronous publication, ...

Graffito tags and templates

1.3 News and Status

Status

2005 June : Incubation status

Graffito is still under incubation. See on the Graffito incubation page to get more information.

2 new committers join the team (Oliver and Sandro). They are mainly focus on the JCR Mapping tools.

Version 1.0-a1-dev is on good way, some portlets are availables, see the maven report to follow the developement process.

2004 December : Graffito - Incubator

Graffito developement is just incubating in the ASF. The code is not 100% stable and not all Graffito features are implemented.

Obviously, you are welcome to send your comments and to contribute !

1.4 Roadmap

Roadmap

1.0-a1-dev

1. Model Service.

2. Simple Version Service.

3. Search service based on Lucene.

4. Security service.

5. Simple demo portlets.

6. OJB Graffito Store implementation.

7. Jetspeed 2 integration.

1.0-a2-dev

1. JCR Integration

2. Webdav integration

1.0-a3-dev

1. Lock Service.

2. Event Service.

3. Notification service.

4. Workflow service.

5. Portlets & tools :

* Admin portlets

* Editors & viewers

6. Graffito applications like GraffitoNews, GraffitoForum, GraffitoScheduler

1.0-a4-dev

More adanced features like replications, more robust admin tools, Support editor like MsWord 2003 & Open Office, ...

1.5 Support

Supporting organisations

Until now, there are 2 companies supporting the Graffito project :

- BlueSunrise - Contact : david@bluesunrise.com
- Sword - Contact : christophe.lombart@sword-technologies.com

2.1 Build all Graffito subprojects

2.2 Deploy Graffito into Jetspeed 2

Deploy Graffito into Jetspeed 2

Review your build.properties

New properties are required to deploy into Jetspeed 2.

Please review your ${USER_HOME}/`build.properties` file with the properties found in the sample file here (See the Jetspeed 2 section) .

Get the Jetspeed 2 M3 distribution

- Get and install Jetspeed 2 M3. The M3 distribution can be found here . This distribution contains Tomcat and the Jetspeed 2 demo application. Don't use the lastest Jetspeed code from SVN.
- If you want to change the database, read carefully the instructions found into the J2 M3 distribution. Check if your database is supported on this page .

Deploy Graffito

- Build Graffito. See Build all Graffito subprojects page .
- Execute `maven allDeploy` from the Graffito root directory.

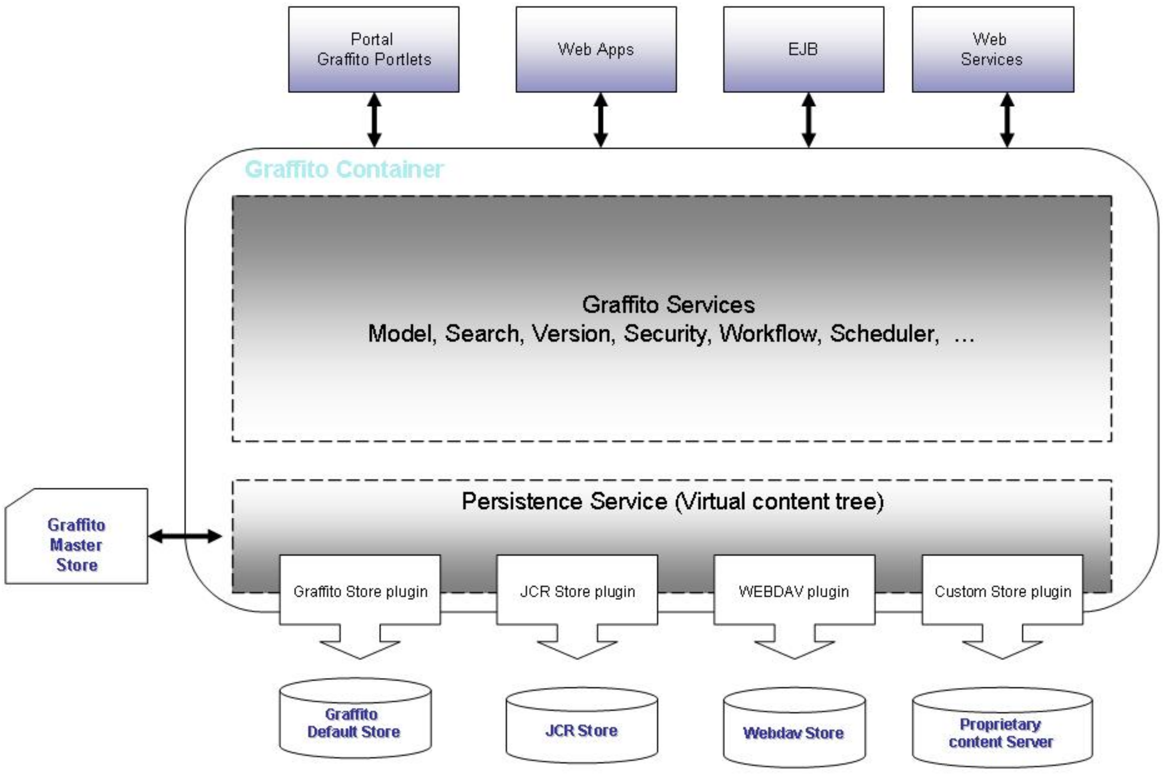
Start Jetpseed on tomcat

- Start Tomcat and connect to jetspeed with `http://localhost:8080/jetspeed`.
- Login as "admin" with password "admin". This is the only user that can manage the Graffito content. If needed, you can add new permissions in the Browser Portlet.

3.1 Portlets

4.1 Architecture

Graffito Engine Architecture



Architecture Neutral

Thanks to the Spring framework, Graffito can be used in different kind of architectures, for examples :

1. Portal server : Graffito is the default Jetspeed CMS engine.
2. Any other web applications can used Graffito.
3. EJB components.
4. Web services.

Virtual Content Tree and Graffito Master Store

Graffito groups together different stores into the same virtual content tree. By this way, your application can access to heterogenous content servers. The principal store required by the Graffito engine is the Graffito Master Store. It contains all store references and knows how to mount thoses stores into the virtual content tree. Graffito Master Store can also contain some CMS objects, permissions setting, scheduler info, ...

The Graffito Engine

The Graffito engine is a container based on 2 distinct layers : The "Graffito services" layer and the "Persistence service" Layer.

The first one, the "Graffito Services" deals with an high level of abstraction. It is composed of different Spring components which are providing an generic API. This API is independant of any CMS content store. Each components solves one of the CMS aspect. For example, the model service manages the CMS objects lifecycle (instantiate, create, update, remove) and the search service gives a way to search content across the complete content tree, ... In order to have more abstraction on the content location, the "Graffito Services" layer dispatchs all requests to the "Persistence Service" layer.
The core Graffito services are :

1. The Model Service.
2. The Version Service.
3. The Security Service.
4. The Search Service.
5. The Index Service.
6. The Workflow Service.
7. The Event handler Service.

Thanks to the Spring assembling script, it is possible to add new Graffito services.

The "Persistence Service" layer is the key component for building the virtual content tree. After reading all store references from the Graffito Master Store, it can mount each of them into the content tree. Mounting a store is based on an URI mapping. All servers are associated to an URI prefix. So, a CMS object is stored into the store which has the same URI prefix. In the current version it is possible to mount a server only on the root level. Later, we plan to mount a server anywhere in the content tree. The "Persistence Service" is based on a plugin design. By this way, it is possible to have different kind of stores (Webdav, Graffito Store or proprietary store). The "Persistenct Service" knows which plugin to use with which server. The plugin goals is to convert a Graffito request (eg. create a document) into a specific CMS API.

The Graffito Store plugin

Graffito Store plugin can be used to access to a predefined DB schema provided with the Graffito distribution. This plugin implementation is based on OJB.

The JCR plugin

[TO BE COMPLETED]

The Webdav plugin

[TO BE COMPLETED]

4.2 Graffito Content Store Setup

Graffito Master Content Store

The principal store required by the Graffito engine is the Graffito Master Store. It contains all store references and other meta information (see the Graffito architecture description). By default, Graffito Master Store is a RDBMS access through OJB. This page explains how to configure Graffito to use your favorite RDBMS product.

TO DO : Explain how to configue a Graffito Store

Graffito WEBDAV Content Store

Graffito JCR Content Store

Define your own Content Store

4.3 Maven goals

Maven Goals

In addition to the standard maven goals, the following goals have been defined. In order to be executed correctly, the Graffito maven goals are using some paramaters defined in your build.properties file (see the page Building Graffito).

Goals defined for the complete project

You can execute the following maven goals from the Graffito root folder (svn trunk folder).

Goal	Description
allClean	Clean all subprojects
start.test.server	start an HSQL test server used for the unit test based on HSQL DB
allBuild	Build all subprojects
allDeploy	Deploy all subprojects into a Jetspeed 2 portal
allSite	Build the complete Graffito site

Goals defined for the API subproject

Goal	Description
deploy	Deploy the API jar into a Jetspeed 2 portal

Goals defined for the Components subproject

Goal	Description
db.scripts.gen	Generate the SQL scripts used by the components.
db.create.test	Create the TEST DB
db.drop.test	Drop all tables in TEST DB
db.delete.test	Delete all tables in the TEST DB
db.reinit.test	db.drop.test + db.create.test

Goal	Description
db.create.production	Create the PRODUCTION DB
db.drop.production	Drop all tables in PRODUCTION DB
db.delete.production	Delete all tables in the PRODUCTION DB
deploy	Deploy the compoment jar into a Jetspeed 2 portal

Goals defined for the Engine subproject

Until now, only unit tests and jar are done. We plan to make a goal to build a default web application which contain the Graffito engine ready to run.

Goals defined for the J2-Deploy subproject

Goal	Description
deploy	Deploy all configuration files into a Jetspeed 2 portal

Goals defined for the JSR-168 Portlet Application subproject

Goal	Description
deploy	Deploy portlet war into a Jetspeed 2 portal
hotdeploy	Hot deploy portlet war into a Jetspeed 2 portal
undeploy	Undeploy portlet war into a Jetspeed 2 portal