Mizura Home

Powered by Open Sources Licences from: Adaptavist Theme Builder, Atlassian Suite, Aquafold Data Studio, Jet Brians IntellyJ IDEA, JProfiler, Yourkit Java Profiler

Macro desconhecida:'html' Macro desconhecida:'html'

Introduction

Welcome to Mizura.

Mizura is an extensible Java framework that provides a universal approach to synchronize technical and managerial artifacts in software enviroment projects.

Based in concept of the connectors, multiple repositories (like Atlassian Jira, Sparxsystems Enterprise Architect, Excel/Google Spreadsheets, Bugzillla, VersionOne, Mantis, CSV files and so) can be synchronized in both ways with many configuration capabilities.

The primarly goal of the Mizura is a seamless integration between Atlassian Jira and Sparxsystems Enterprise Architect (widely used tools in software development scenarios), keeping synchronized project tasks and the equivalent UML artifacts, like use usecases, requirements, activities and so. The default Mizura implementation should offer connectors for JIRA, EA, CSV, XML, Excel and databases tables.

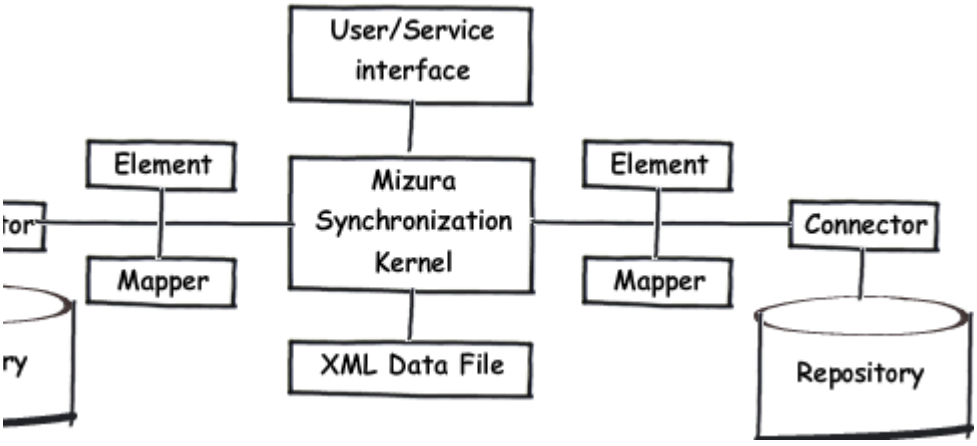
Learn more

With Mizura, you can automate and synchronize project management tasks with your UML models, enabling agile teams to really implement the concept of "points" or "stories". In another words, with Mizura, you can create an usecase (or requirement, or activities, or... anything!) in a Enterprise Architect and, them, automatically create - and always keep synchronized - the related issue in Jira, or vice versa. Using this approach, your usecases /requirements/so can be showed in kanban charts or priorized

Mizura works in a bidirectional way, creating issues from UML elements, or creating UML elements from issues. In anycase, the full synchronization is enabled.

Architecture

Bellow, the entire architecture of Mizura:



Each part as describe here:

- **Repository**: Is a datasource, and can be anything, like bugtrack tools (like JIRA, Mantis, Bugzilla, ClearQuest, Redmine, etc.), management tools (MSProject, Primavera, CASE tools (like Enterprise Architect, Rational, Together, Caliber, StarUML, etc.), flat files (CSV, XML), databases tables, spreadsheets (Excel, Calc, GDocs, etc.) or even live data (Webservices, REST).
- **Connector**: Is a gateway beetween the Mizura and the external world. Connectors are responsible for query and change objects in the repositories. Each repository must have one connector with CRUD (Create, Retrieve, Udpdate and Delete) support. For example, talking about database connectors, they must be responsible for Selects, Updates, Deletes and Inserts in the database tables. The standard Mizura package ships with 5 connectors: JIRA, EA, CSV, XML and database tables. You can easly create new connectors implementing only 7 API methods!
- **Element** : Is a synchronizable artifact, like an issue in bugtrack tools, an UML element (Use Case, Requirement, Activity, etc.) in CASE tools, a table row in database tables, a row line in CSV files, a XML element in XML files, a task in management tools an so. Each element as a universal identifier, so, it must be unique in your repository. Are these identifiers used to synchronize elements between repositories. Each repository can hold many different element types (ie. in CASE tools, an Use Case is an element type, an Requirement is another element type, and so.). Each element has a bunch of fields (like name, status, description, timestamp dates, and so). Each field as one type (like text, number, enumeration, array, logic types and so.). Any Java type is a valid field type in the Mizura.
- **Mapper** : A mapper is responsible to *map* an element from/to real representation in the repository. Effectivelly, each mapper has only two methods: `load()` (to initialize an element from your real state in the respository) and `store()` (to save back the element changes in the object repository). Specifically, each element type has one mapper. If the synchronization need a special treatment for an element type or attribues (ie. conversion operations), it is the mapper task realize this.
- **Synchronization kernel**: This is the Mizura kernel. I a few words, a configurable code block that continuously scan respositories changes and apply synchronizations when necessary. Mizura support two way synchronization beetween any repository, enabling eighteen distinct behaviors for each synchronization. (add, update, update if new, delete, purge, and so).
- **Data file**: Its a simple XML file that contain all configuration information necessary for synchronizations like connectors definitions, element and field types and, of course, the synchronized element IDs.
- **User/Service interface**: For now, Mizura its a service application, running in a background thread in the Operating System (ie. Windows Tray/Service or Linux Daemon). Only one instance of the Mizura can syncrhonize any number of local or remote repositories in the network. I a near future, a graceful user interface should be available in a desktop or web interface (at this time, we are not worrying about it yet, because the focus is the functionality, nor usability).

Current status and Future works

Mizura was launched in Jan/2010, like a personal effort from Marcelo Mrack for the 3PUP movement.

The first version (alpha - proof of concept) was released in March, 31 2010 with support to synchronize elements from EA to JIRA repositories.

During 2010, the project remained idle, with a focus on another activites (yeah, open source projects are very complicated ...)

But, in first half of 2011, the project was resumed.

We expect that a first functional release will be available in ~~September, 2011~~ ~~February 2014~~ July 2015, using Open Source licence (Apache 2.0).

Until that, follow the posts and latest releases in the project.

Management

Latest management messages

Since December 2014, we are seeking funds for the project. We expect news under this area until half of February 2015. The latest code is available here.

Latest worklog activities

04/11/2013 - mmrack - 1h - (MZ-41) Resolved Hon nor Behavior flags on Add, Update and Delete operations. See commit today.

08/11/2013 - mmrack - 3h - (MZ-40) At bus to Sta Cruz, codifing "sanitize" method in SynchImpl.

19/11/2013 - mmrack - 30m (MZ-30) Install JDK 7 32bits on Bamboo server

Charts & Metrics

Erro ao processar a macro "gadget"

Error rendering gadget [http://3layer.com.br/jira/rest /gadgets/1.0/g/com.pyxis. greenhopper.jira:greenhopper- gadget-project-dashboard/gadgets /greenhopper-project-dashboard. xml]

Builds & Downloads

Não foi possível recuperar http://3layer.com.br/bamboo/rss/createAllBuildsRssFeed.action?feedType=rssAll&buildKey=MZ-TRUNK - Página não encontrada.

Effort & Costs

Macro desconhecida:'html'

Navigate