TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Study program:	Electrical Engineering and Informatics
Study branch:	Mechatronics

Release Management in Atlassian JIRA – Extending an Issue Tracking System

Master thesis

Author:	Bc. Viktor Dlouhý
Project leader:	Prof. Dr. rer. nat. Stefan Bischoff
Consultants:	Dipl.-Ing. (FH) Thomas Haak
	Dipl.-Ing. (FH) Christian Bork

Statement

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

22. 7. 2014

Abstract

The master thesis deals with a development of plugins for JIRA system. JIRA is an issue tracking tool allowing easy project and issue management. The thesis explores possibilities of system enhancements mostly by the release management features. Further the large amount of work is dedicated to general technologies, which are used for web application development. These technologies are then applied for designing and implementing the plugins.

Keywords

JIRA, Atlassian, Java, plugin, development, release management

Abstrakt

Tato diplomová práce se zabývá vývojem rozšíření pro systém JIRA, který umožňuje snadnou evidenci a správu projektů. Práce se snaží prozkoumat možnosti vylepšení systému především v oblasti release managementu. Dále je velký prostor věnován obecným technologiím, jenž se používají při vývoji webových aplikací. Tyto technologie jsou dále uplatněny při návrhu a implementaci samotných rozšírení pro systém JIRA.

Klíčová slova

JIRA, Atlassian, Java, plugin, programování, release management

Table of Contents

List of Illustrations

Glossary

ActiveObjects – Java based Object Relational Mapping for databases

AJAX – Asynchronous JavaScript, a technology for asynchronous communication

API – Application Programming Interface, list of developer usable entities (eg. functions)

Architecture – hight level structure of a software

AUI – Atlassian User Interface, set of advices and graphical components

Build – result of building procedure created from some source code

Client – software or hardware that access exposed services of server

Continuous delivery – process of automated software delivery

Continuous integration – process of automatically building and testing of software

Framework – software which provides API and other features for running applications

GUI – Graphical User Interface

HTML – Hypertext Markup language, language to define web pages

HTTP – Hypertext Transfer Protocol, application protocol used by web pages

Implementation – realization of an abstract plan

Instance – concrete software object in memory

Interface – set of abstract information specifying certain communication abilities

Jackson – data processing tool for JSON

Java – programming language

JavaScript – scripting language

Jersey – library for RESTful web services

jQuery – library for JavaScript

JSON – JavaScript Object Notation is an data format to exchange information

Library – collection of supporting software

Markup language – system for annotation and description of a document

Marshalling – process of transforming raw data into object representation

Maven – Apache Maven is a build tool

Mockup – quick illustration of desired GUI

MVC architecture – Model View Controller architecture, programming concept

Protocol – collection of digital rules for data exchange

Release – result of release procedure

Repository – stored data structure provided by server

Request – HTTP request, a message going from client to server

Response – HTTP response, a message answering request

REST - Representational state transfer, architectural style of communication

Revision – special incrementing identificator describing source in time

Server – software or hardware that allows clients to communicate via enabled services

Servlet – Java Servlet class provides capabilities to implement web server

SQL database – database using SQL language for query

Tomcat – web server and servlet container

VTL – Velocity Template Language

XML – Extensible Markup Language, general purpose description language

1. Introduction

1.1. Situation

Sphairon is a company developing its own hardware and software. The company makes number of routers and modems with advanced features of telephony. Sphairon has long time history which goes up to 1948, when it was founded as VEB Fernmeldewerk Bautzen a company devoted to network systems. Nowadays company enjoys stable growth and is mostly dedicated to research and development. The company was bought by a famous ZyXEL in the May 2013, which brought strong international partner from technology and business view.

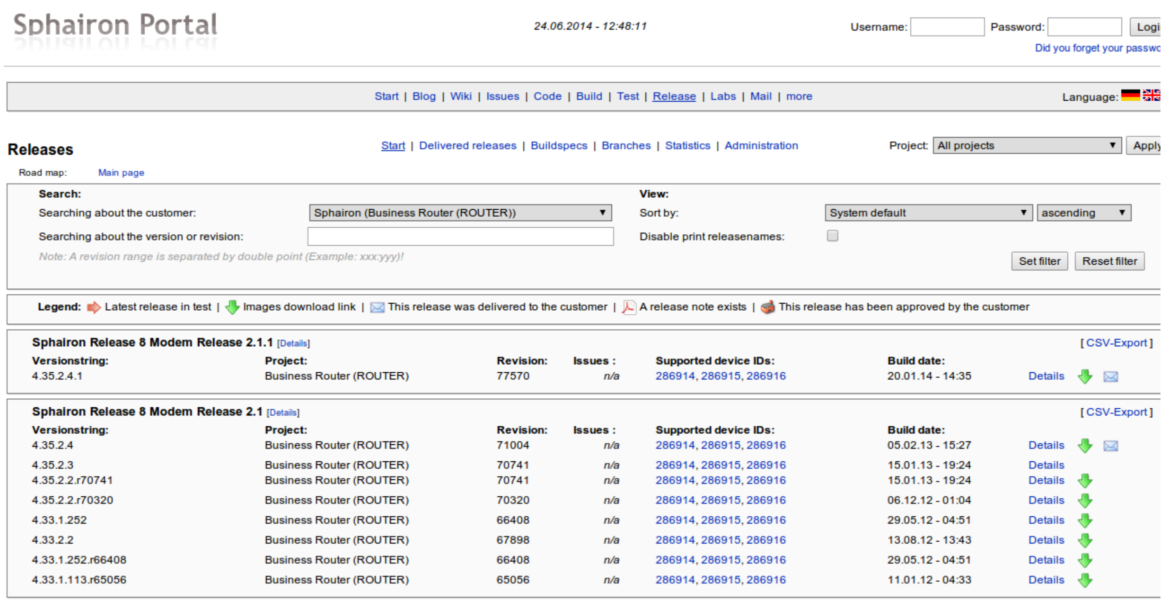
Illustration 1: Logo of Sphairon

Sphairon develops its own embedded Linux distribution. More than 80000 revisions were created during the time, it consists of 200 different software components and has 14 million lines of code in total. Sphairon Linux empowers very well known products as O2 HomeBox 1, Vodafone EasyBox 803S or NetCologne Deluxe. Tons of software releases were created delivering new features and updates to the customers.

The firm uses modern approaches in managing software releases, however the tools are kind of segmented. Some of the tools can be classified as a free software. Subversion is a **revision control** system maintained by Apache, it is as a free software. Jenkins is an open source continuous integration tool. It facilitates **test system** functionality in Sphairon. However many other functions are handled by proprietary server based system. It includes software automatic building, release management and release archivation. Atlassian JIRA is a great **issue tracking** tool, which can be also utilized as a project management tool.

Sphairon benefits on its own independent web portal and on mostly open-source continuous integration tools. This solution has its pros and cons. The biggest problem is context fragmentation and data multiplicity. If somebody uses commercial continuous integration tool, everything is handled inside. But if you want to be independent, it brings such issues.

To properly maintain software an issue tracking system is necessary. JIRA is a professional issue tracking tool. Its data structure is pretty simple, you have projects and issues. Issue can be a bug, task or any kind of user story. If you create an issue, you can assign a version to it. The problem is that the version does not contain a lot of information. Sphairon uses its own release management tool called ReleaseAdmin (main page on Illustration 2) to maintain versions and keep information. If somebody makes a release it needs to be listed in ReleaseAdmin. Release has a name, which is the same as the name of version in JIRA. If customer makes a complaint, it is necessary to create an issue (bug) in JIRA. If the bug is fixed, a new release must be created in ReleaseAdmin, but also the same version in JIRA must change its status to released. You cannot release a software until all bugs are fixed. So it is necessary to maintain two tools and keep switching between them to have complete data information. Unfortunately this can lead to wrong linking of different data information and possibly to create the crucial mistakes.


Sphairon Portal
24.06.2014 - 12:48:11
Username:
Password:
Logi
Did you forget your passwo
Start | Blog | Wiki | Issues | Code | Build | Test | Release | Labs | Mail | more
Language:
Releases
Start | Delivered releases | Buildspecs | Branches | Statistics | Administration
Project: All projects
Apply
Road map: Main page

Search:
Searching about the customer: Sphairon (Business Router (ROUTER))
Searching about the version or revision:
Note: A revision range is separated by double point (Example: xxx:yyy)!

View:
Sort by: System default ascending
Disable print releasenames: ☐

Legend: Latest release in test | Images download link | This release was delivered to the customer | A release note exists | This release has been approved by the customer

Sphairon Release 8 Modem Release 2.1.1 [Details] [CSV-Export]

Versionstring:	Project:	Revision:	Issues :	Supported device IDs:	Build date:			
4.35.2.4.1	Business Router (ROUTER)	77570	n/a	286914, 286915, 286916	20.01.14 - 14:35	Details	⬇	✉

Sphairon Release 8 Modem Release 2.1 [Details] [CSV-Export]

Versionstring:	Project:	Revision:	Issues :	Supported device IDs:	Build date:			
4.35.2.4	Business Router (ROUTER)	71004	n/a	286914, 286915, 286916	05.02.13 - 15:27	Details	⬇	✉
4.35.2.3	Business Router (ROUTER)	70741	n/a	286914, 286915, 286916	15.01.13 - 19:24	Details		
4.35.2.2.r70741	Business Router (ROUTER)	70741	n/a	286914, 286915, 286916	15.01.13 - 19:24	Details	⬇	
4.35.2.2.r70320	Business Router (ROUTER)	70320	n/a	286914, 286915, 286916	06.12.12 - 01:04	Details	⬇	
4.33.1.252	Business Router (ROUTER)	66408	n/a	286914, 286915, 286916	29.05.12 - 04:51	Details	⬇	
4.33.2.2	Business Router (ROUTER)	67898	n/a	286914, 286915, 286916	13.08.12 - 13:43	Details	⬇	
4.33.1.252.r66408	Business Router (ROUTER)	66408	n/a	286914, 286915, 286916	29.05.12 - 04:51	Details	⬇	
4.33.1.113.r65056	Business Router (ROUTER)	65056	n/a	286914, 286915, 286916	11.01.12 - 04:33	Details	⬇	

Illustration 2: ReleaseAdmin

1.2. Task

First Step

The first step is to bring data provided by standalone release management system (ReleaseAdmin) into JIRA. This creates new user context inside JIRA, which simplifies process of managing releases and brings synoptic and suitable number of views giving user convenient and desired information about releases. Mapping between ReleaseAdmin data entities and JIRA data entities is required.

The views can fulfill information role only and the editing feature must be still handled only by ReleaseAdmin itself.

The diagram in Illustration 3 shows the enhancements, which can ReleaseAdmin for JIRA bring into development process.

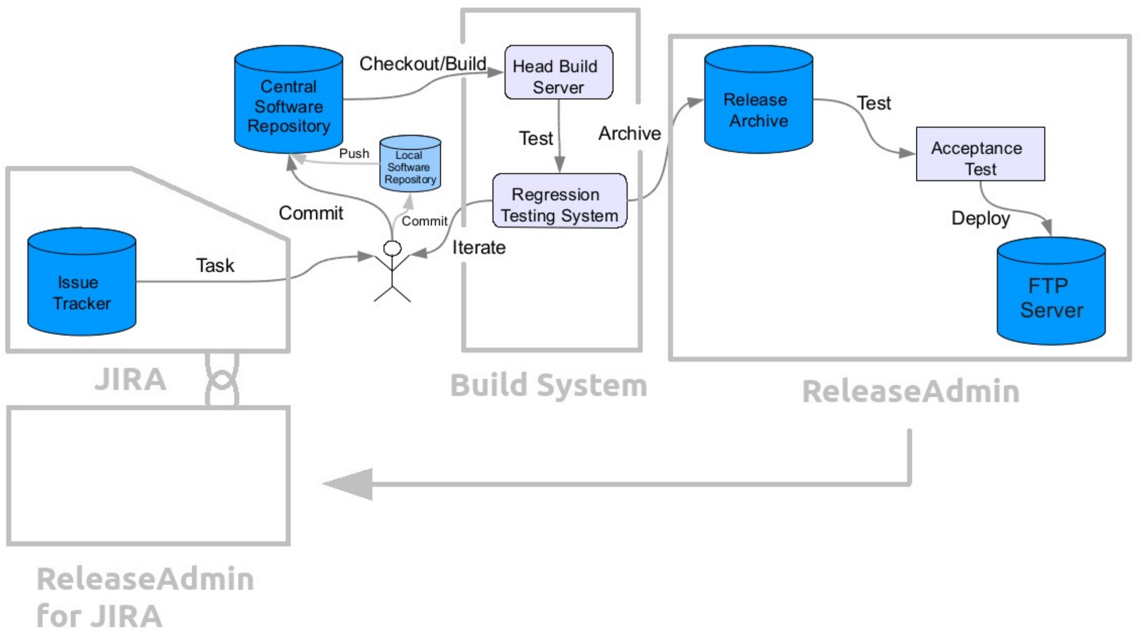


Illustration 3: ReleaseAdmin for JIRA

Second Step

The second step is to create extension plugin, which brings release management principles of continuous integration into JIRA issue tracking system. Continuous integration is composed from many steps and requires many tools, issue tracking is one them. The task is to implement release management into JIRA and create plugin (Illustration 4), which can stand on its own. The development procedure should keep in mind possibilities of other continuous integration principles to may be implemented in future. Many import functions are required for the plugin.

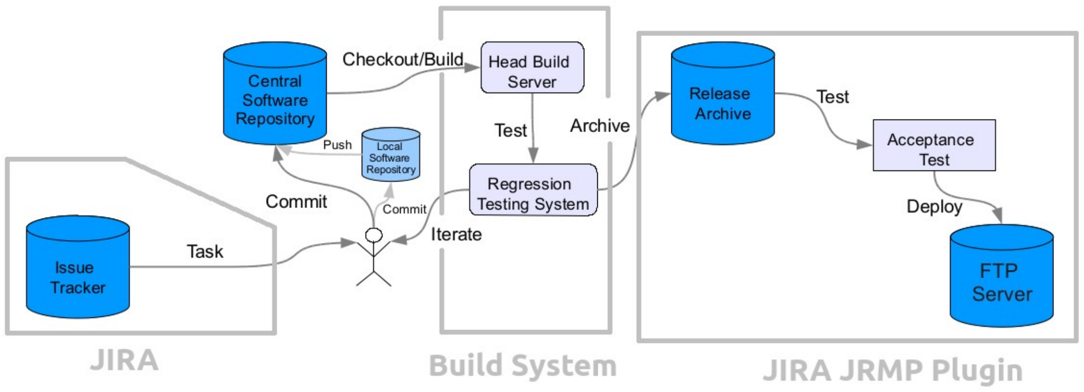


Illustration 4: Jira Release Management Plugin

2. Theoretical Foundations

2.1. Continuous Integration

Modern practice to develop software is to use continuous integration principles. Continuous integration (Illustration 5) is a collection of different development tools to provide faster development process, better collaboration in team and swift resolution of possible bugs. [5]

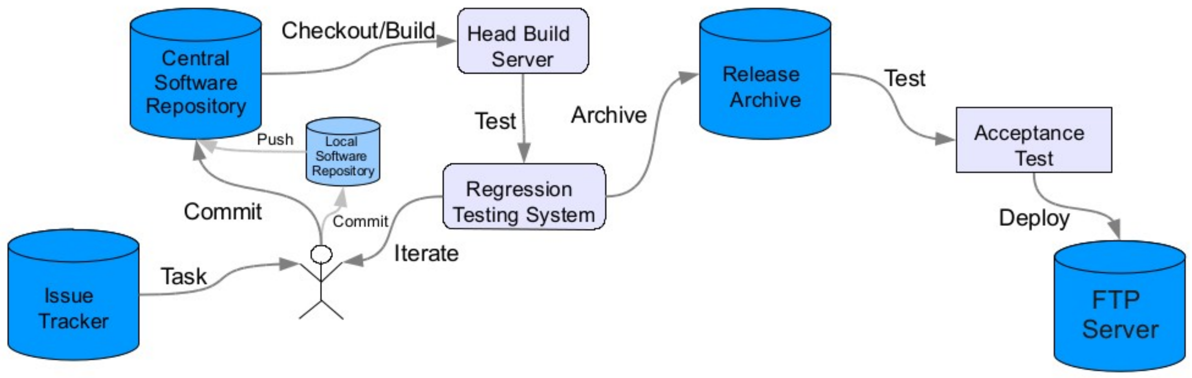


Illustration 5: Continuous Integration at Sphairon

Sometimes these practices are implemented separately as different tools, which can work together and user touch is required. These practices can be also managed in continuous integration systems, which handle all steps in one tool.

Issue Tracker

Issues are tasks which are necessary to be processed. It is not only bugs, it can be different milestones, functionalities and many others. This is the place where a demand is created.

Revision Control

The main principle is to store all code into software repository. Repository provides central storage of code for all members in a team. Most importantly code repository supports versioning of the code. Commitments should be done regularly.

Revision control system is a software providing control over changes of source code. It is designed mostly for team developers, but can be used for individuals. It stores changes in source code and attaches a time stamp information. It prevents concurrent access by locking files, so that only one developer has write rights. It provides some other important operations such as merging, branching, updating, messages checking and many others.

Branching and revision creation can be seen on the Illustration 6.

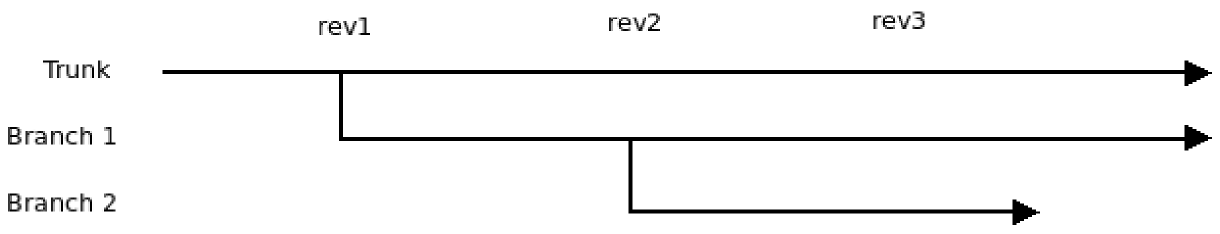


Illustration 6: Branching

Build Automation

Building is complicated process, that's why automation is necessary. Automation of a build often includes deployment, binaries compiling, documentation generation and many others. Once the code is built, testing should be added to confirm all behaves as expected.

Deployment

Making builds available to testers can reduce large amount of rework. Acceptance test can be necessary.

2.2. Release Management

Release management is a process of planning, testing and deploying releases. The whole procedure starts when the release is planned for the future. This situation is triggered by announcing a bug, or when the new features from the software are expected. These issues are grouped together for a new upcoming release. Then continues integration phase comes – the development and testing. The software needs to be properly tested. This is the testing phase of release. If the tests go wrong, everything is handled in frame of continuous integration circle. The testing can be done automatically or manually by exposing the software to testers in the field. The deployment phase is actually composed from the archiving of final version, testing and finally by releasing to the customers.

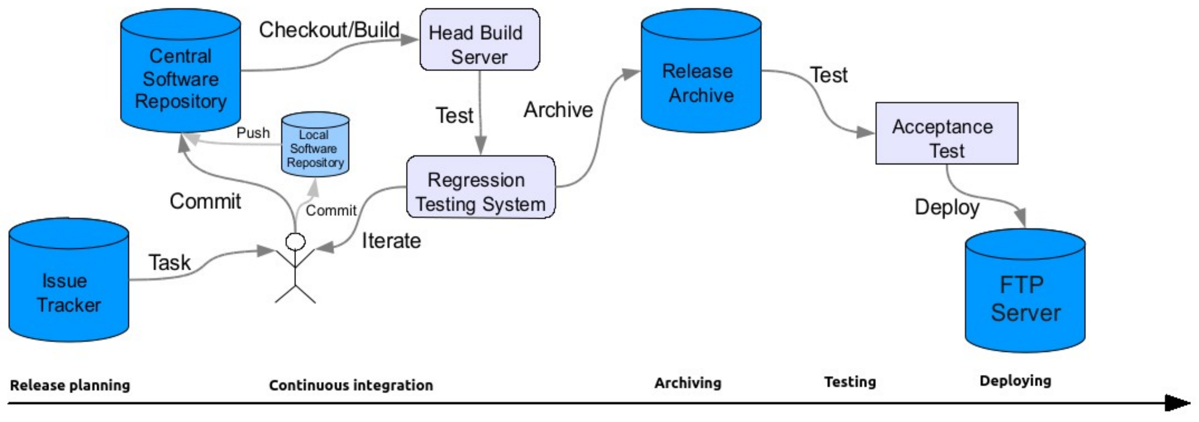


Illustration 7: Release management timeline

Above steps can be understood as a managing and storing the information about the procedure. The most important things are that all releases must be stored somewhere with all information about the steps created during the development, testing and deploying progress.

With storing the details about the release, the key information can be:

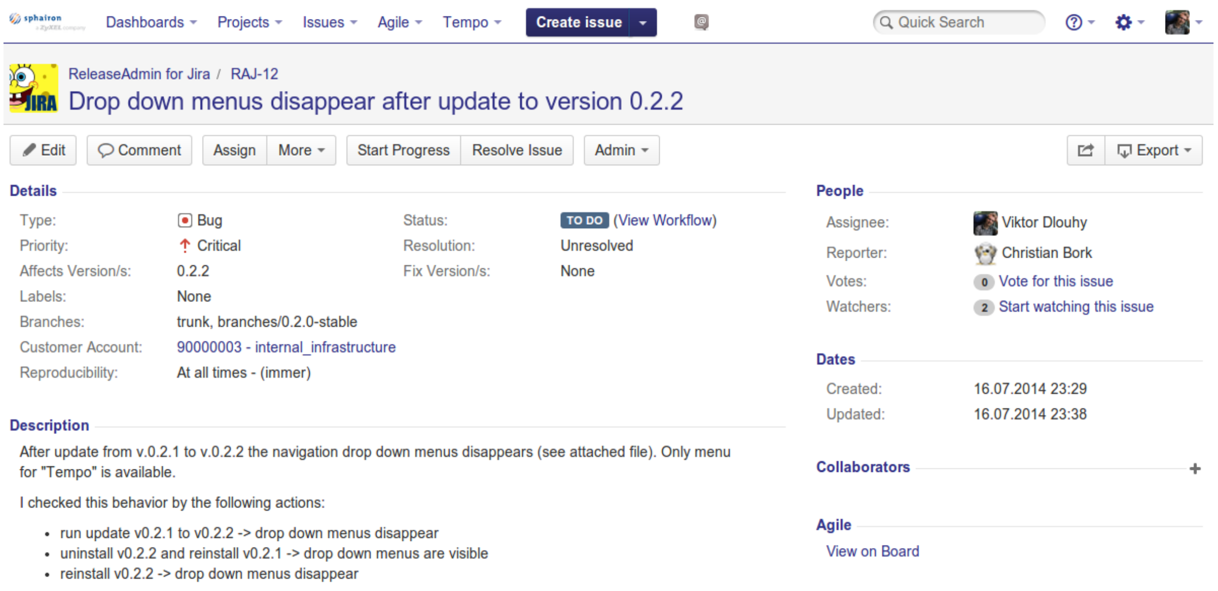
- name of the release – original version name
- targets
- source of the release
- build protocols with build results

- test protocol
- build date
- release date
- customer and others

Some of these information can be just a link to the other software eg. revision control system, build system and so on. Actually release management software can connect (or integrate) all tools from continuous integration process together to provide detailed information about a software in any step of its life-cycle.

2.3. JIRA

Atlassian JIRA is an issue tracking web based system, which helps to manage development processes with focus on continuous integration principles (2.1). The system is multi-user based and every user can have different permissions. Issues and projects are main entities provided by JIRA. Issues can be understood as a task, reminder, complaint or in case of software development a bug. Issue page (Illustration 8) is a view showing detail of certain issue. It allows to add comments, attach files or for example change status of the issue. Project is an entity enabling to group issues into big units based on eg. product.



Drop down menus disappear after update to version 0.2.2

Details

Type:	Bug	Status:	TO DO (View Workflow)
Priority:	Critical	Resolution:	Unresolved
Affects Version/s:	0.2.2	Fix Version/s:	None
Labels:	None		
Branches:	trunk, branches/0.2.0-stable		
Customer Account:	90000003 - internal_infrastructure		
Reproducibility:	At all times - (immer)		

Description

After update from v.0.2.1 to v.0.2.2 the navigation drop down menus disappears (see attached file). Only menu for "Tempo" is available.

I checked this behavior by the following actions:

- run update v0.2.1 to v0.2.2 -> drop down menus disappear
- uninstall v0.2.2 and reinstall v0.2.1 -> drop down menus are visible
- reinstall v0.2.2 -> drop down menus disappear

People

Assignee:	Viktor Dlouhy
Reporter:	Christian Bork
Votes:	0 Vote for this issue
Watchers:	2 Start watching this issue

Dates

Created:	16.07.2014 23:29
Updated:	16.07.2014 23:38

Collaborators

Agile

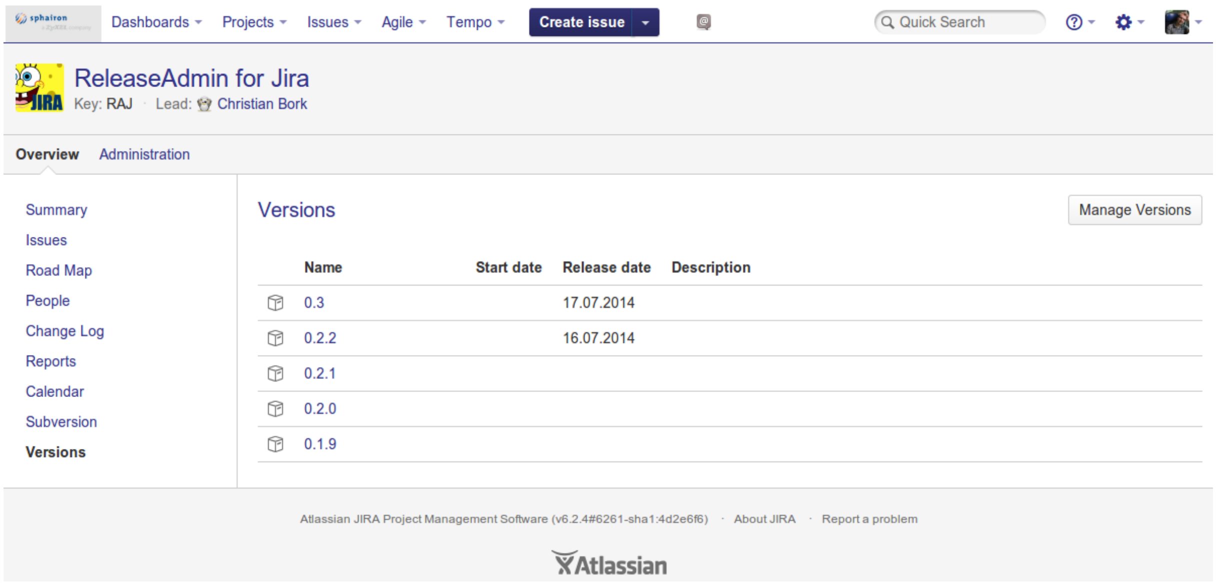
View on Board

Illustration 8: JIRA Issue page

JIRA also helps with planning and managing of versions. One can create a version and assign it to an issue. When the version is released (version becomes actually release), all issues should be resolved. Versions can be observed also as a road map. This view shows how many issues with same assigned version are done or undone. It allows simple overview of planned versions and work status.

The JIRA content can be divided into two classes from user perspective. The first class is view only (Illustration 9), this is usually called overview and the second class is administration. Administration views are visible only for those with edit rights for eg. certain project.

JIRA supports many other functionalities like statistics, reports, advanced search possibilities. The thing which is important, JIRA as other Atlassian products support plugin development. Atlassian has rich documentation how to create add-ons and enrich its JIRA product with other different functions and processes.



ReleaseAdmin for Jira

Key: RAJ · Lead: Christian Bork

Versions

Name	Start date	Release date	Description
0.3		17.07.2014	
0.2.2		16.07.2014	
0.2.1			
0.2.0			
0.1.9			

Atlassian JIRA Project Management Software (v6.2.4#6261-sha1:4d2e6f6) · About JIRA · Report a problem

Illustration 9: JIRA Project with Versions page

3. Preparation

3.1. General Technologies

First it is necessary to summarize possible technologies, which are used for web application. General technologies are platform independent and can be applied on different cases. The principles or approaches can be then used during development.

3.1.1. REST Communication

REST is an conceptual style designed for distributed environment, which describes communication interface [18]. REST proposes several methods based on HTTP protocol:

- GET (Retrieve)
- POST(Create)
- DELETE
- PUT (Update)

These methods can be implemented on a REST server on different paths and the client can use them to achieve desired result. REST is very often used to exchange data between two separate applications. However this approach can be also utilized to provide internal information exchange between client (browser) and server (web) to manage asynchronous communication.

A client using jQuery library in JavaScript is described in Code 1, where the the AJAX technology is used to generate request and receive response in text format. This snippet is actually concrete REST client written in JavaScript.

```
function releaseSync(id) {
      $.ajax({
            url : "path",
            type : "PUT",
            dataType : "text",
            contentType : "text/plain",
            success : function(data) {
                  alert(data);
            },
            error : function(response) {
                  alert(response.responseText);
            }
      });
}
```

Code 1: REST client - JavaScript

REST server can be implemented in Java using common Servlet class. REST client can be created in Java with raw HttpClient class from Jakarta Commons project. But since there exists open source Jersey library, it is convenient to use it. Jersey is a library based on Java API for RESTful Services (JAX-RS) and makes the job very well. [8]

An example how to create instance of REST client using Jersey in Java language, make a request and receive response in String representation can be observed from Code 2. This is actually concrete REST client in Java language.

```
public String releaseSync(int id) {
     ClientConfig config = new DefaultClientConfig();
     Client client = Client.create(config);
     WebResource  webResource  =  client.resource(contextPath  +"/"  +  id  +
"/sync");
     return webResource.get(String.class);
}
```

Code 2: REST client – Java – Jersey

3.1.2. XML Marshalling

XML is a general markup language, which is very often used to store or exchange information. If a random application provides public API, it usually offers JSON or XML format. Advantage of XML is clarity and flexibility. Modern approach to read external data consists of XML marshalling, this means a way to transform raw XML data into specific data object. [27]

XML files are very popular, because everyone can create own XML markups. If you want to show others which markups you use, you need to describe it. This is when the XML schema comes. So called XSD files are based on XML and contains information about user created XML files. It describes elements and attributes, simple and complex types, model groups and attribute relationships. [12]

Code 3 shows example of XSD schema file. Following XML object (Code 4) comes from the given schema.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema                                  elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element name="County" type="xs:string" minOccurs="0" />
        <xs:element name="PostCode" type="xs:string" />
        <xs:element name="Country" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Code 3: XSD schema file

```xml
<?xml version="1.0" encoding="utf-8"?>
<Address xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="SimpleAddress.xsd">
  <Recipient>Mr.WalterC.Brown</Recipient>
  <House>49</House>
  <Street>FeatherstoneStreet</Street>
  <Town>LONDON</Town>
  <PostCode>EC1Y8SY</PostCode>
  <Country>UK</Country>
</Address>
```

Code 4: XML output based on XSD schema

Schemes help very much, if you want to use data from external API. From provided XSD file you can create your own data model which corresponds to API and implement own serializer. Or there are already prepared tools, which can make the work for you. This approach is called XML data binding.

Java Architecture for XML Binding (JAXB) provides separate tools to generate Java classes from given XSD file. Mind that Java classes are generated externally, not in running application. Notice XML annotations, which are generated by JAXB. In Code 5 you can see how the XSD file (Code 3) was transformed into Java code by JAXB.

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "", propOrder = {
    "recipient",
    "house",
    "street",
    "town",
    "county",
    "postCode",
    "country"
})

@XmlRootElement(name = "Address")
public class Address {
    @XmlElement(name = "Recipient", required = true)
    protected String recipient;
    @XmlElement(name = "House", required = true)
    protected String house;
    @XmlElement(name = "Street", required = true)
    protected String street;
    @XmlElement(name = "Town", required = true)
    protected String town;
    @XmlElement(name = "County")
    protected String county;
    @XmlElement(name = "PostCode", required = true)
    protected String postCode;
    @XmlElement(name = "Country")
    protected String country;
}
```

Code 5: Java data class representation

Jersey is a library based on Java API for RESTful Services (JAX-RS). Jersey can bed used to make HTTP request from Java code to external API and marshall received data into instances of data classes (Code 6). Special annotations are required in marshaled data class (Code 5). [9]

```
public <T> T makeRequest(Class<T> newClass) {
      WebResource webResource = client.resource(path);
      webResource = client.resource(path);
      T response = webResource.get(newClass);
      return response;
}
```

Code 6: Client request and XML marshalling

3.1.3. JSON Marshalling

JSON is an open standard format to exchange data. It is often used as an alternative to XML. Advantages are simplicity and high efficiency. It is composed from attribute-value pairs. JSON is often used in AJAX techniques. If it is necessary to marshal JSON data, there are several approaches to do it. It can be transformed to primitives or mapped to prepared data classes. It provides also schema descriptions.

Jackson is a library for Java application to make marshalling of JSON. It provides annotations and of course object mapper as a marshaller. [6]

In Code 7 you can find and example of incoming data in JSON format. For the purposes of marshalling a general data class is necessary (Code 8).

```
{
	"release_id":"779",
	"project_id":"11",
	"versionstring":"4.37.2.4",
	"builddate":"1394623979",
	"releasename_id":"292",
	"sw_generation":"4",
	"released":"1",
}
```

Code 7: JSON object

```java
public class ReleaseModel {
	@JsonProperty(value = "release_id")
	protected short releaseId;
	@JsonProperty(value = "project_id")
	protected byte projectId;
	@JsonProperty(value = "versionstring")
	protected String versionstring;
	@JsonProperty(value = "builddate")
	protected int builddate;
	@JsonProperty(value = "releasename_id")
	protected short releasenameId;
	@JsonProperty(value = "sw_generation")
	protected byte swGeneration;
	protected byte released;
}
```

Code 8: Java data class representation

Combination of Jackson and Jersey can be used to manage HTTP request and marshalling itself (Code 9).

```
public <T> T makeRequest(Class<T> newClass) {
      WebResource webResource = client.resource(path);
      String response = webResource.get(String.class);
      ObjectMapper mapper = new ObjectMapper();
      T rObject = mapper.readValue(response, newClass);
      return rObject;
 }
```

Code 9: Client request and JSON marshalling

3.1.4. Object Database

ActiveObjects is a Java based Object Relational Mapper developed from the ground as a fast and easy to use library. It supports number of SQL databases like Derby, HSQLDB, MSSQL, MySQL, Oracle or PostgreSQL. It is distributed under Apache free software license. ActiveObjects can be download as a standalone package or using Maven repository. [15]

ActiveObjects provides simple mapping between data classes in Java and database (DB) itself, so the Java object can be easily stored into database without using SQL queries. Most importantly data classes can be extended or changed and ActiveObjects handles changes, migrations or upgrades.

ActiveObjects API provides few crucial classes to work with. EntityManager manages database access from objective perspective, nevertheless it still supports SQL queries if necessary. Entity interface needs to be extended in own data interface, which describes data model. Entity is not recommended to be implemented and then instantiate. To create instance of desired data object EntityManager factory method should be used.

Code 10 shows an example of extending Entity interface into own data model. Setters and getters are used to map attributes into database columns. Code 11 explains how to create an object from given data model interface.

```
public interface Person extends Entity {
    public String getFirstName();
    public void setFirstName(String firstName);
    public String getLastName();
    public void setLastName(String lastName);
}
```

Code 10: Data model interface

```
EntityManager em = new EntityManager(jdbcURI, username, password);
Person p = em.create(Person.class);
p.setFirstName(“Pavel”);
p.setLastName(”Mokry”);
p.save();
```

Code 11: Using manager to create and store object [21]

ActiveObjects API supports also some demanding functionalities like relations, indexing or preloading. Annotations as @OneToMany and @ManyToMany can be used to describe relationship between two different data models connected in DB table by id. But in ActiveObjects you do not need to bother with joining the tables. It is done for you by special relation annotation tags (Code 12). [23]

```
public interface City extends Entity {
    public String getName();
    public void setName(String name);
    @OneToMany
    public Person[] getOccupants();
}

public interface Person extends Entity {
    public String getFirstName();
    public void setFirstName(String firstName);

    public String getLastName();
    public void setLastName(String lastName);

    public City getCity();
    public void setCity(City city);
}
```

Code 12: Data model relations [23]

As written before it is not recommended to implement Entity interface (only to extend). But how to achieve similar result? How to implement methods itself? Solution exists, however the implementation is little bit painful. Annotation @Implementation must be used and different class must be created, the name is specified by annotation. Implementation class needs to provide reference to Entity in constructor as it is described in Code 13.

```
@Implementation(PersonImpl.class)
public interface Person extends Entity {
    public String getFirstName();
    public void setFirstName(String firstName);

    public String getLastName();
    public void setLastName(String lastName);

     public City getCity();
     public void setCity(City city);
 }

 public class PersonImpl {

     private Person model;

     public PersonImpl(Person model) {
         this.model = model;
     }

     public String getFirstName(){
         return model.getCustomer().toUpperCase();
     }
 }
```

Code 13: Data model and implementation class [22]

3.2. JIRA Plugin Technologies

JIRA is a standalone application, that is capable of extension. Extensions are called plugins. Plugins should be designed in MVC architecture as well and Atlassian as a company standing behind JIRA provides many tools to achieve this. As a view layer you can use WebWorks or generic Servlets (3.2.6), as a data layer ActiveObjects mapper (3.1.4) is available. All these components are called plugin modules. Plugin modules expose certain kind of available functionality to plugin such as above written view, data layer or for example REST service.

3.2.1. Architecture

JIRA is a web based application using MVC architecture. JIRA is written in Java and is deployed as WAR file into Java Servlet Container usually Tomcat. [14]

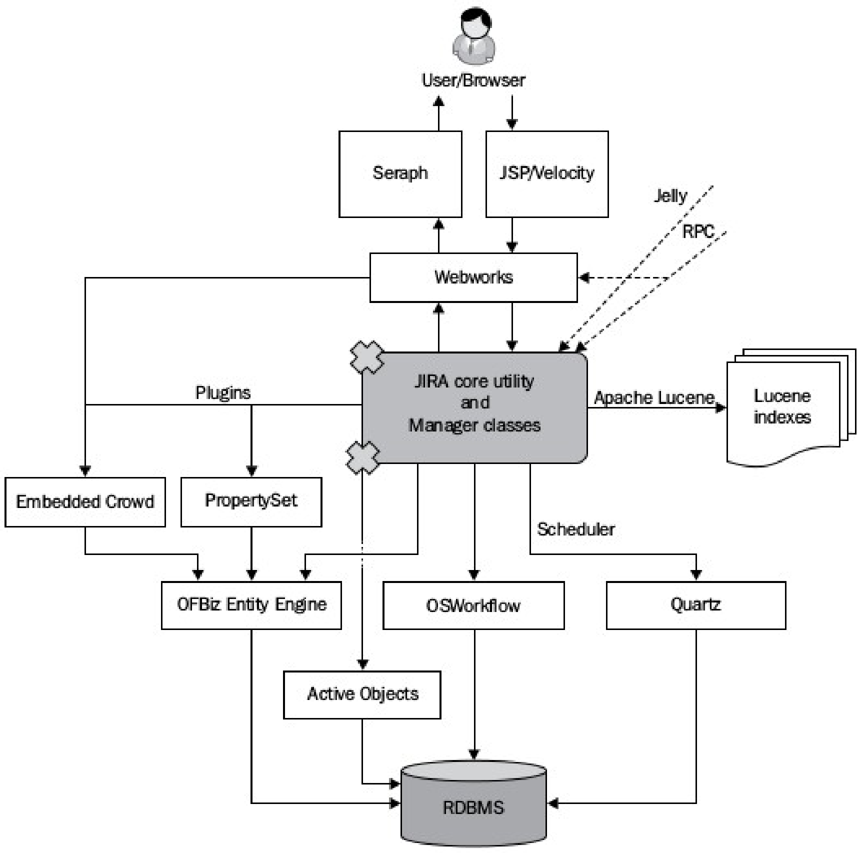
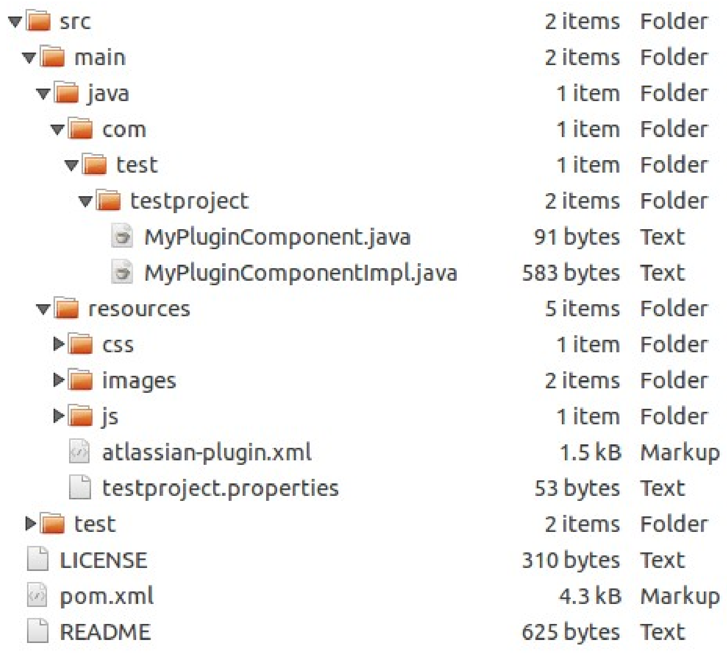


Illustration 10: Jira Architecture [14]

3.2.2. Plugin File Skeleton

Every plugin project must fulfill certain structure. Most important files are description files. File atlassian-plugin.xml contains information about used resources, plugin modules and other components. File pom.xml holds important information how to build plugin, which libraries (dependencies) to use, it also contains name, version and other specific fields about a plugin. The main folder contains the source code written in Java. The resources folder contains views, JavaScript codes, CSS styles. Files with extension .properties contains string values, which can be used for localization, if necessary.



```
▼ src                                      2 items   Folder
  ▼ main                                   2 items   Folder
    ▼ java                                 1 item    Folder
      ▼ com                                1 item    Folder
        ▼ test                             1 item    Folder
          ▼ testproject                    2 items   Folder
              MyPluginComponent.java      91 bytes   Text
              MyPluginComponentImpl.java 583 bytes   Text
    ▼ resources                            5 items   Folder
      ▶ css                                1 item    Folder
      ▶ images                             2 items   Folder
      ▶ js                                 1 item    Folder
        atlassian-plugin.xml              1.5 kB     Markup
        testproject.properties          53 bytes     Text
  ▶ test                                   2 items   Folder
  LICENSE                              310 bytes     Text
  pom.xml                                 4.3 kB     Markup
  README                               625 bytes     Text
```

Illustration 11: Plugin file structure

3.2.3. Important Modules

WebWork Plugin Module

WebWork plugin module is based on OpenSymphony WebWork1 framework. It provides view layer for JIRA plugins and interaction for users. WebWork can be invoked from links and return certain view. [25]

Servlet Plugin Module

General Servlet can be also used to provide any interaction between client and server. Servlet can return view or just a generic data. Servlet allows to use all of common HTTP methods. [24]

Project Tab Panel Plugin Module

This kind of module exposes possibility to add certain view into project page of JIRA. [19]

Version Tab Panel Plugin Module

This kind of module exposes possibility to add certain view into version page of JIRA. [13]

Component Plugin Module

From global perspective objects are created by JIRA itself when they are defined as modules (Code 14). Component plugin module makes possible to share own object between modules in plugin since you can not refer to the object manually, you need to use constructor injection method, example is in Code 15. [4]

```
<!-- COMPONENT - DATA PROVIDER -->
<component key="data-accessor" class="com.releaseadmin.DataAccessor"
name="Data Accessor" i18n-name-key="data-accessor.name">
      <description key="data-accessor.description">Data
Accessor</description>
</component>
```

Code 14: Component declaration

```
public class ReleasesRestService {

      private DataAccessor dataAccessor;

      public ReleasesRestService(DataAccessor dataAccessor) {
            this.dataAccessor = dataAccessor;
      }
...
}
```

Code 15: Example of injection

REST Plugin Module

REST module enables REST API service to be added into plugin. This can be used for communication between client and server. [16]

```
<rest name="Rest Service" i18n-name-key="rest-service.name"
           key="rest-service" path="/rest" version="1.0">
           <description key="rest-service.description">Rest Service
Plugin</description>
</rest>
```

Code 16: REST declaration

ActiveObjects Plugin Module

This module allows to store data model into object database. Data model must be declared as ActiveObject Plugin Module.

```
<!-- ACTIVE OBJECTS - DATA ITEM -->
<ao key="ao-module">
     <description>The module configuring the Active Objects used by this
plugin
     </description>
     <entity>com.example.data.ModelOne</entity>
     <entity>com.example.data.ModelTwo</entity>
</ao>
```

Code 17: Data model declaration

Component Import Plugin Module

This module allows to access Java components shared by other plugins. [7]

3.2.4. Velocity Templates

Velocity is a Java based template system, which separates Java code from views, but still it provides access to Java objects, which enables to get all public parameters or to call public methods. Simple and powerful scripting language Velocity Template Language (VTL) can be used to generate dynamic views. It gives options for example to create variables, make conditions and cycles. Since you can generate pages through Velocity, you can even generate JavaScript code, which brings almost endless possibilities. [2]

```
<html>
<body>
Hello $customer.Name!
<table>
#foreach($mud in $mudsOnSpecial )
   #if ($customer.hasPurchased($mud))
      <tr>
        <td>
          $flogger.getPromo($mud)
        </td>
      </tr>
   #end
#end
</table>
</body>
</html>
```

Code 18: Example of VTL [2]

Atlassian recommends to use Velocity Templates and mostly as a developer of plugins, it is the only way to create views. Simple way to create a view for Atlassian plugin is in Code 18. Using VTL it is easy to share objects between Java code and Velocity Template, most importantly Atlassian usually provides some basic objects itself, so it is not big problem to get for example base URL (Code 19) or to import some specific resources.

```
$webResourceManager.requireResource("com.atlassian.auiplugin:aui-
experimental-table-sortable")
...
<a href="${requestContext.baseUrl}/browse/$projectKey/fixforversion/
$version.getId()">$name</a>
...
```

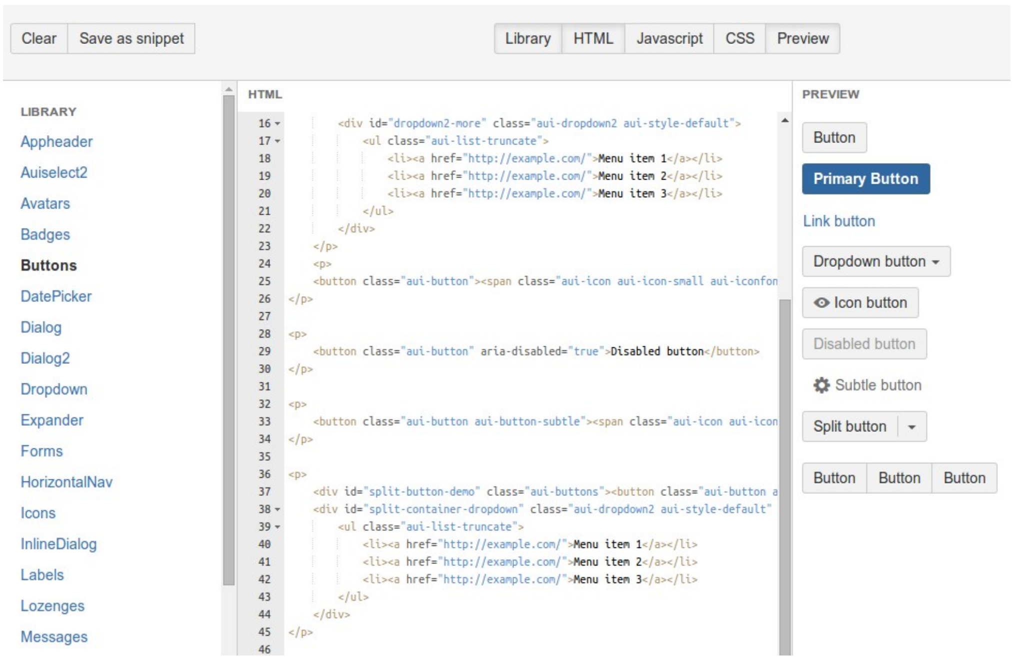
Code 19: Example of adding resources and using baseUrl

3.2.5. Atlassian User Interface

Atlassian User Interface (AUI) is a library of visual components, which Atlassian shows in its products including JIRA. These components are also available for plugin developers. A component is usually composed from HTML and can have JavaScript to provide some functionality. There are many components from the basic ones as buttons and dialogs or whole page decorators. [17]

There is also Atlassian Design Guidelines which is a library of articles and recommendations for both Atlassian developers and plugin developers. Every component from AUI is described from view of designer with suggestions how to use them and where to place them. [17]

Atlassian Sandbox (screenshot on Illustration 12) is a web tool for developers, which makes easier to create views. All available components are prepared to be copied including its JavaScript functions. Using Sandbox you can also run own JavaScript functions or test own HTML layouts.



```
        <div id="dropdown2-more" class="aui-dropdown2 aui-style-default">
            <ul class="aui-list-truncate">
                <li><a href="http://example.com/">Menu item 1</a></li>
                <li><a href="http://example.com/">Menu item 2</a></li>
                <li><a href="http://example.com/">Menu item 3</a></li>
            </ul>
        </div>
    </p>
    <p>
    <button class="aui-button"><span class="aui-icon aui-icon-small aui-iconfon
</p>

<p>
    <button class="aui-button" aria-disabled="true">Disabled button</button>
</p>

<p>
    <button class="aui-button aui-button-subtle"><span class="aui-icon aui-icon
</p>

<p>
    <div id="split-button-demo" class="aui-buttons"><button class="aui-button a
    <div id="split-container-dropdown" class="aui-dropdown2 aui-style-default"
        <ul class="aui-list-truncate">
            <li><a href="http://example.com/">Menu item 1</a></li>
            <li><a href="http://example.com/">Menu item 2</a></li>
            <li><a href="http://example.com/">Menu item 3</a></li>
        </ul>
    </div>
</p>
```

Illustration 12: Atlassian Sandbox

3.2.6. Servlets and WebWorks

As it is written before (3.2.4) to create a view, you can use Velocity Templates. To create controllers for these views, you can use Java Servlets or OpenSymphony WebWork. Typical web application interacts with a server using web forms. On the server side, information are handled by Java Servlet. Servlet is like a controller, which access database and returns needed data back to the view.

Atlassian provides OpenSymphony WebWork version 1. WebWork are build on top of the servlet combining other frameworks like XWork and implementing dynamic parameter mapping to JavaBeans, validation of request and other features [26]. However with WebWork you cannot handle AJAX request, because response is always a view, response can not be in JSON format, this requires special object for asynchronous accessing eg. REST API server. So WebWorks have its positive and negative sides.

Table 1: Comparison of WebWork and own solution

feature	**own solution**	**WebWork**
Multiple view for action	yes	yes
Template layout system	yes	no (only velocity)
Security managed	no	yes
HTTP req validation	partly	partly
AJAX possible	yes	no
Proven solution	no	yes

The idea would be to create own controller, which supports everything on top of the Servlet, but since the WebWorks are quite well documented, proven and secure technology and on top of that it is generally advised by Atlassian to use WebWorks. From conceptual point of view, it is anyway better to divide controller for views and controller for background communication (AJAX) as it is possible to see on Illustration 13.

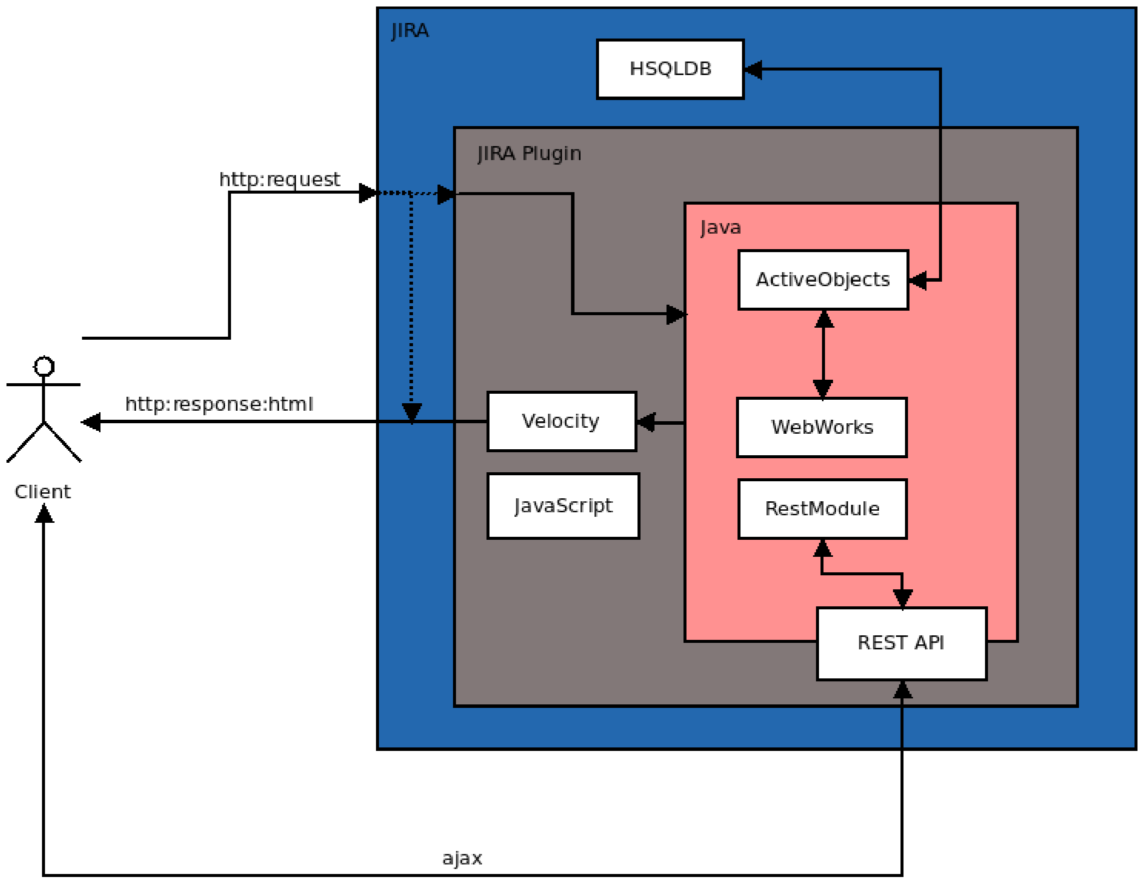


Illustration 13: Simplified communication flow

4. Development

4.1. ReleaseAdmin for JIRA

4.1.1. Motivation

The most important motivation is to help Sphairon company to have correct and full information about its releases. To remove switching of content in browser, which can easily cause mistakes and to have all necessary information about releases in one tool.

As JIRA has possibility to add versions to issue, it can be used to recognize release in ReleaseAdmin. When the correct release exists in ReleaseAdmin (same name as JIRA version) offer these information from ReleaseAdmin in JIRA. When the proper version is not found in ReleaseAdmin, it is probably not a software development issue. Or it is not correctly named.

This plugin does not have any ambition to be used by somebody else than Sphairon employees. It should be proprietary software.

4.1.2. Specifications

- Communication in JSON format
- Data model similar to ReleaseAdmin Rest API output
- Version in JIRA corresponds to release in ReleaseAdmin
    - Release extends version by adding new information (release is version)
- Ask ReleaseAdmin for releases presence
- Cache releases in JIRA plugin database
- Release in cache DB is always valid, allow user to refresh an item in the cache
- Provide simple mapping of projects
    - ReleaseAdmin project vs Jira project, eg.: ISDN-SIP-Gateway (SIPGW) vs ISDN-SIP-Gateway
    - Allow one-to-many relation in JiraProject to ReleaseAdmin project
- Show list of cached releases in JIRA
- Show panel on issue page of corresponding cached releases
- Show link to ReleaseAdmin if possible
- Show full information on version page and compact information on issue panel
- Support of devices in many-to-many relation to version
- Support of release notes
- Allow to download all information about all versions in JIRA project to cache

4.1.3. Communication

One of the most important thing is the communication between JIRA plugin and ReleaseAdmin. The plugin needs to retrieve data from ReleaseAdmin and show them. The technology which is proper for this situation is REST communication (3.1.1), because it uses usual HTTP protocol and fits the most for web application purposes.

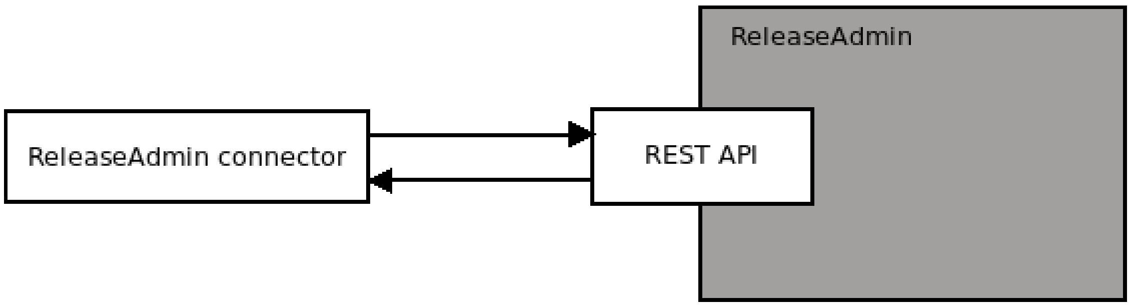


Illustration 14: ReleaseAdmin communication

REST communication is just a method how to exchange data. Another thing is the data format. Most often used standards JSON and XML are possible. For its efficiency JSON is used in this case.

Two REST commands are necessary on ReleaseAdmin side:

- Provides data about **release** based on release name.
    - URL eg.: api/rest/latest/version?version=2.4.5.1
    - returns empty JSON string or data about given release
- Provides data about **projects** in ReleaseAdmin
    - URL eg.: api/rest/latest/project
    - returns array of projects, its name and id

Using JSON Marshalling (3.1.3) it is pretty simple to transform JSON output into JAVA data objects.

4.1.4. Data

The main idea to create data structure is to separate data model for communication (ReleaseModelHelper) and data model for caching (ReleaseModel). Data objects which comes from ReleaseAdmin REST output must be transformable to Java objects from JSON. This process is called marshaling and for this purpose Jackson library can be used (3.1.3).

Model which goes to database must be separate in case it was necessary for example to be extended. Inheritance can be used, however another thing that tells the model must be separated is the limitation of ActiveObjects framework – we need to use interfaces to describe an item in database – not classes (3.1.4). And of course if the ReleaseAdmin REST API changes, we just change communication data class instead of whole application dependent data class.

The mapper must be implemented to convert communication objects into database objects. The opposite direction can be also useful, in case somebody wants to use AJAX for browser to server communication.

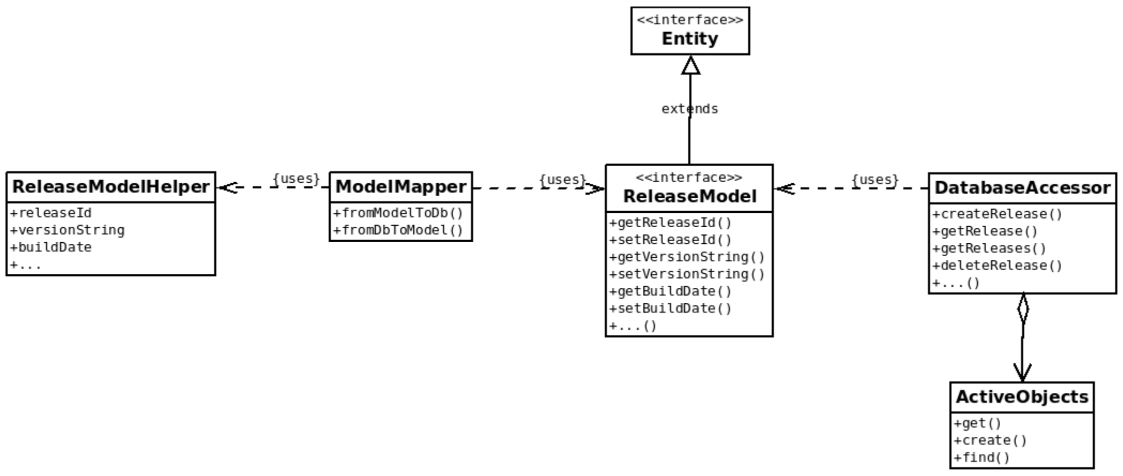


Illustration 15: Simplified data model

Another data classes (other than ReleaseModel) are necessary. For example to store devices and projects. However the idea with mapping stays.

In Code 20 there is a full implementation of ReleaseModelHelper with all annotations necessary for marshaling from JSON data to Java instances.

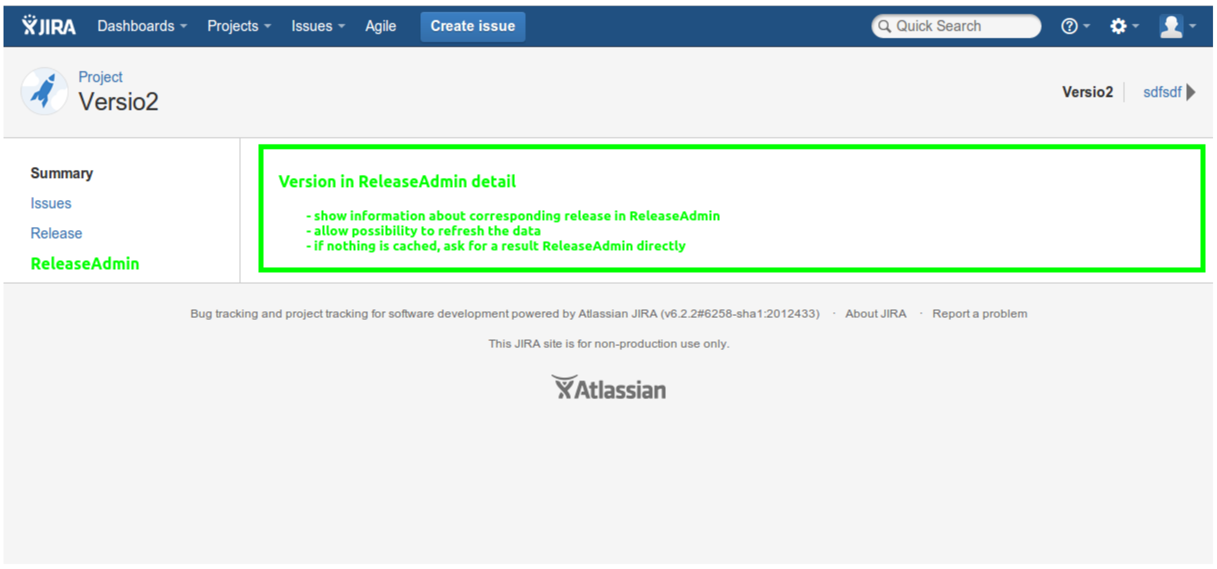
```java
@JsonIgnoreProperties(ignoreUnknown = true)
public class ReleaseModelHelper{

	@JsonProperty(value = "release_id")
	protected short releaseId;
	@JsonProperty(value = "project_id")
	protected byte projectId;
	@JsonProperty(value = "versionstring")
	protected String versionString;
	@JsonProperty(value = "builddate")
	protected int buildDate;
	@JsonProperty(value = "releasename_id")
	protected short releaseNameId;
	@JsonProperty(value = "sw_generation")
	protected byte swGeneration;
	@JsonProperty(value = "released")
	protected byte released;

	@JsonProperty(value = "releaseddate")
	protected int releasedDate;

	@JsonProperty(value = "url")
	protected String url;

	@JsonProperty(value = "project")
	protected String project;

	@JsonProperty(value = "releasename")
	protected String releaseName;

	@JsonProperty(value = "customer")
	protected String customer;

	@JsonProperty(value = "devices")
	protected List<ReleaseDeviceModelHelper> devices;

	@JsonProperty(value = "revision")
	protected int revision;

	@JsonProperty(value = "branch")
	protected String branch;

	@JsonProperty(value = "buildspec")
	protected ReleaseBuildSpecModelHelper buildSpec;

	@JsonProperty(value = "buildsystem")
	protected ReleaseBuildSystemModelHelper buildSystem;
...
}
```

Code 20: ReleaseModelHelper data class

4.1.5. Mockup

Mockups are simplified illustrations of views to help design an application structure. For the mockups, there are many tools using different approaches to generate illustrations. Mockups can be very simple, the purpose is to find out where to place GUI components and how they should behave. Usual screenshots of pure JIRA instance can be used together with simple paint tool.

ReleaseAdmin for JIRA needs to have separate page (Illustration 16) with details of release information contained in cache. For the administrators, the button to reload data from remote host can be visible.



Version in ReleaseAdmin detail

- show information about corresponding release in ReleaseAdmin
- allow possibility to refresh the data
- if nothing is cached, ask for a result ReleaseAdmin directly

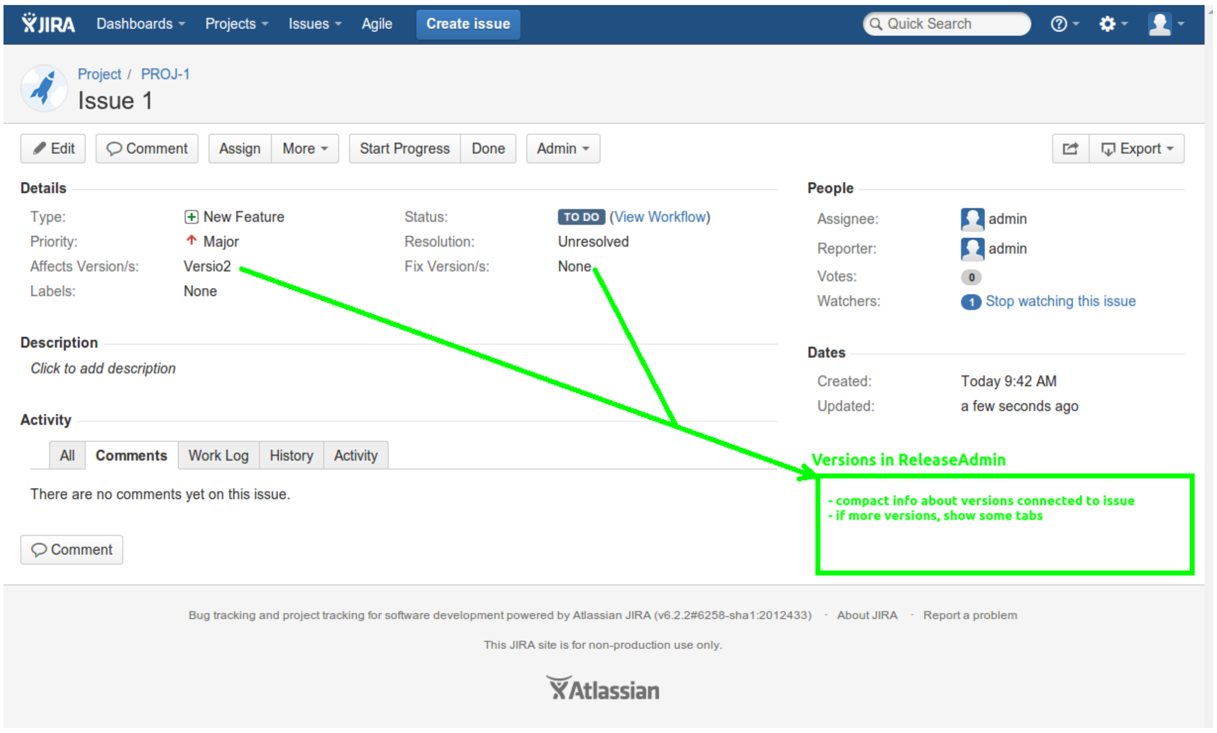
Bug tracking and project tracking for software development powered by Atlassian JIRA (v6.2.2#6258-sha1:2012433) · About JIRA · Report a problem

This JIRA site is for non-production use only.

Atlassian

Illustration 16: New tab on version page, shows detail of release in ReleaseAdmin

The issue page is crucial for JIRA. So that is why it so important to introduce panels showing brief information of corresponding releases. An issue can have affects versions and fixed versions, so it is necessary to implement two panels.



Project / PROJ-1

Issue 1

Details

Type:	New Feature	Status:	TO DO (View Workflow)
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	Versio2	Fix Version/s:	None
Labels:	None		

People

Assignee:	admin
Reporter:	admin
Votes:	0
Watchers:	1 Stop watching this issue

Description

Click to add description

Dates

Created:	Today 9:42 AM
Updated:	a few seconds ago

Activity

There are no comments yet on this issue.

Versions in ReleaseAdmin

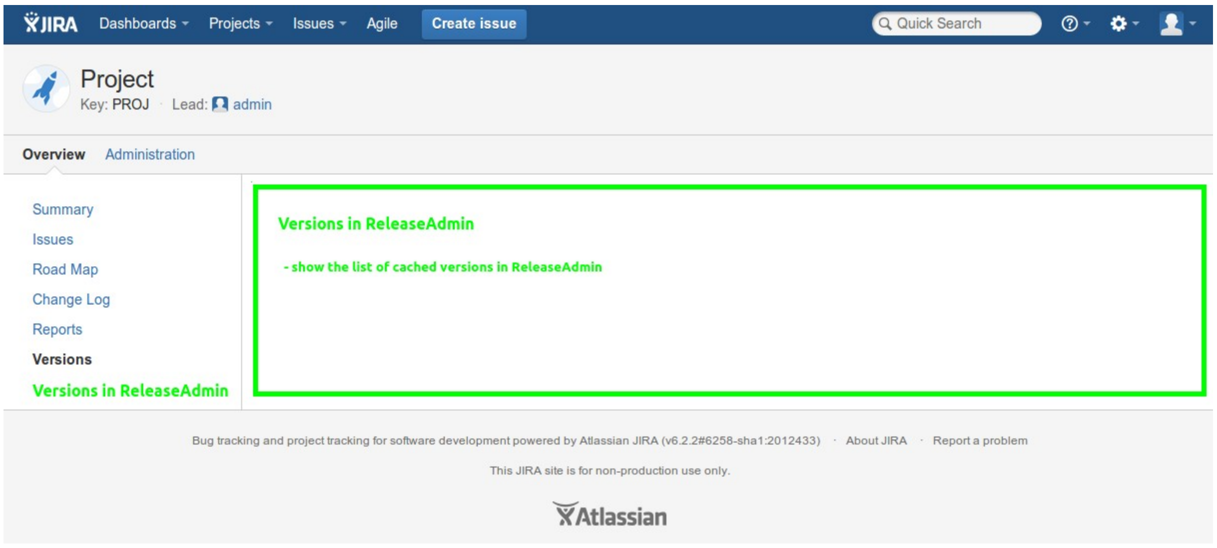
- compact info about versions connected to issue
- if more versions, show some tabs

Bug tracking and project tracking for software development powered by Atlassian JIRA (v6.2.2#6258-sha1:2012433) · About JIRA · Report a problem

This JIRA site is for non-production use only.

Illustration 17: New panels on issue page with basic info about version

List of cached releases needs to be created to provide complex view for all versions in project. This list can show small amount of information about releases stored in JIRA.



Versions in ReleaseAdmin

- show the list of cached versions in ReleaseAdmin

Bug tracking and project tracking for software development powered by Atlassian JIRA (v6.2.2#6258-sha1:2012433) · About JIRA · Report a problem

This JIRA site is for non-production use only.

Atlassian

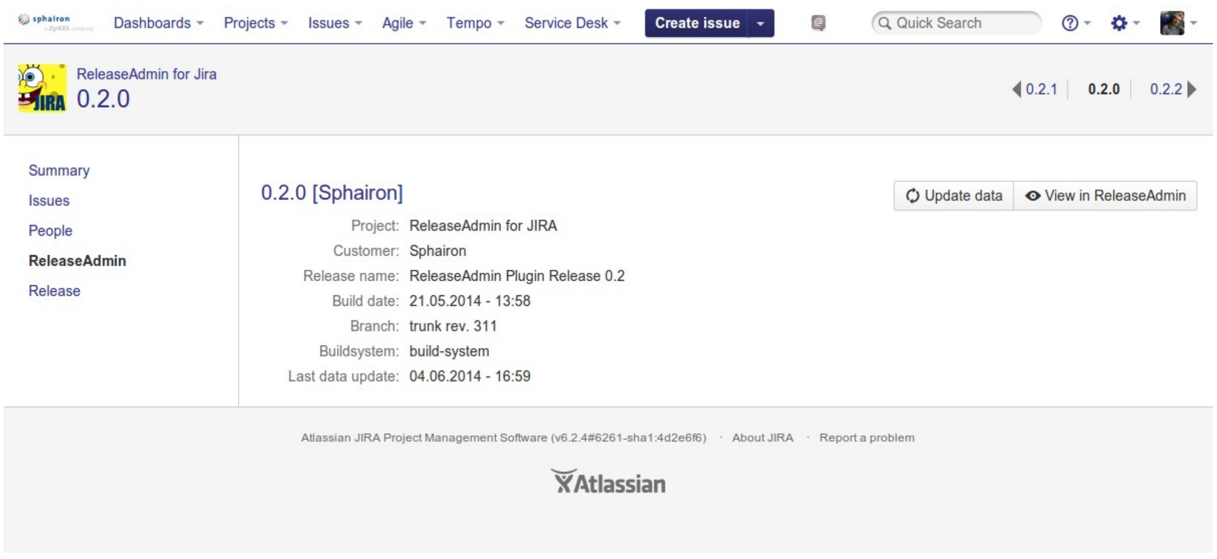
Illustration 18: New tab on project page, shows the version list

4.1.6. Views

All views are composed from Java classes, Velocity templates and usually some JavaScript functions. Java class needs to extend some Atlassian view class (3.2.3). The proper definition in atlassian-plugin.xml is necessary too.

Version detail page

Version detail page (Illustration 19) contains detailed information about release in ReleaseAdmin connected to version in JIRA. It is a tab on version page of JIRA. It offers also a few buttons. Update data allows project administrator to get new data from ReleaseAdmin. There are buttons available, if release contains release notes or download links. If more corresponding releases to currently viewing version are found, all of them are visible.



0.2.0 [Sphairon]

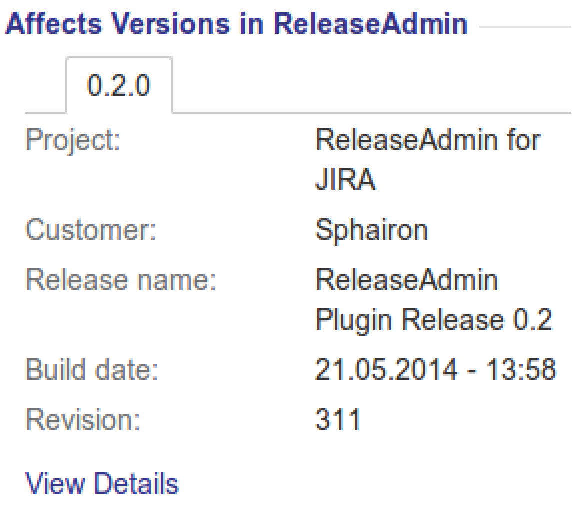
Project: ReleaseAdmin for JIRA
Customer: Sphairon
Release name: ReleaseAdmin Plugin Release 0.2
Build date: 21.05.2014 - 13:58
Branch: trunk rev. 311
Buildsystem: build-system
Last data update: 04.06.2014 - 16:59

Atlassian JIRA Project Management Software (v6.2.4#6261-sha1:4d2e6f6) · About JIRA · Report a problem

Illustration 19: ReleaseAdmin full detail

Issue page panel

Issue page panel (Illustration 20) provides information on issue page. This is a crucial view, that brings necessary data from ReleaseAdmin into JIRA. There are two panels, one contains details about affect version and one about fixed version.



Affects Versions in ReleaseAdmin

0.2.0

Project:	ReleaseAdmin for JIRA
Customer:	Sphairon
Release name:	ReleaseAdmin Plugin Release 0.2
Build date:	21.05.2014 - 13:58
Revision:	311

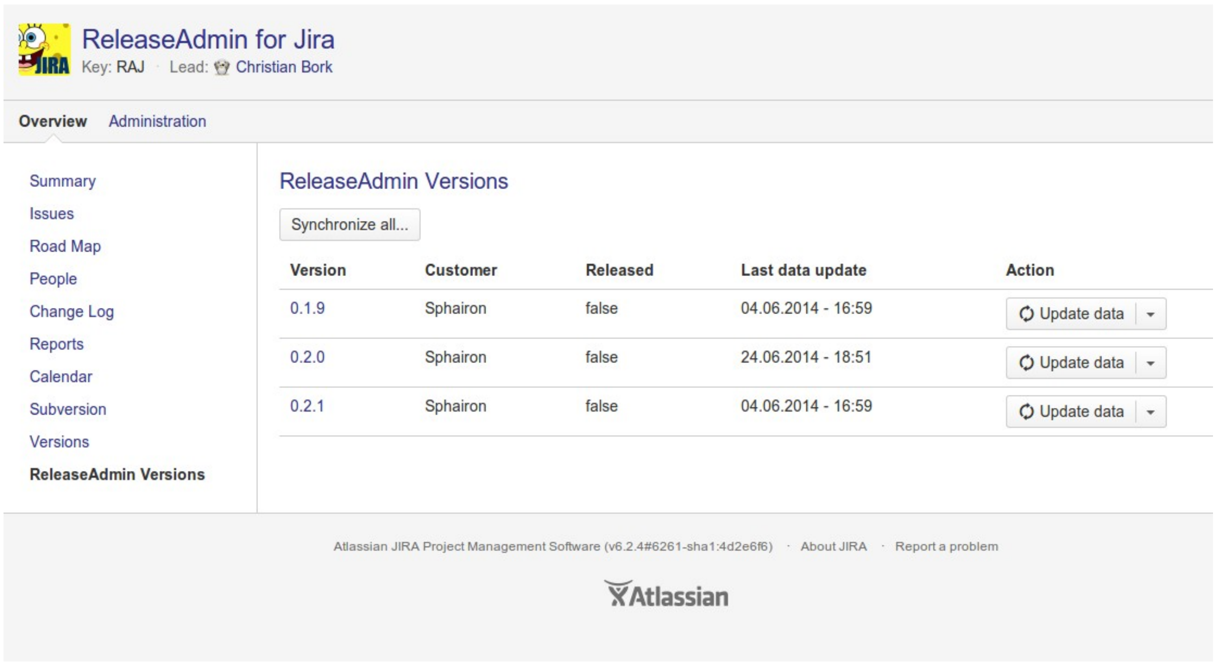
View Details

Illustration 20: Affects Versions panel

Version list page

Project page has many tabs concerning to a project. It usually shows all issues included, people assigned, associated versions and others. One could say, why not to extend current versions tab with new information regarding to ReleaseAdmin. To add new columns into table and new links. Sadly this is not possible and Atlassian does not provide any correct way to change existing Versions tab. This means that own tab must be created, which contains all necessary information about cached versions (releases in ReleaseAdmin).

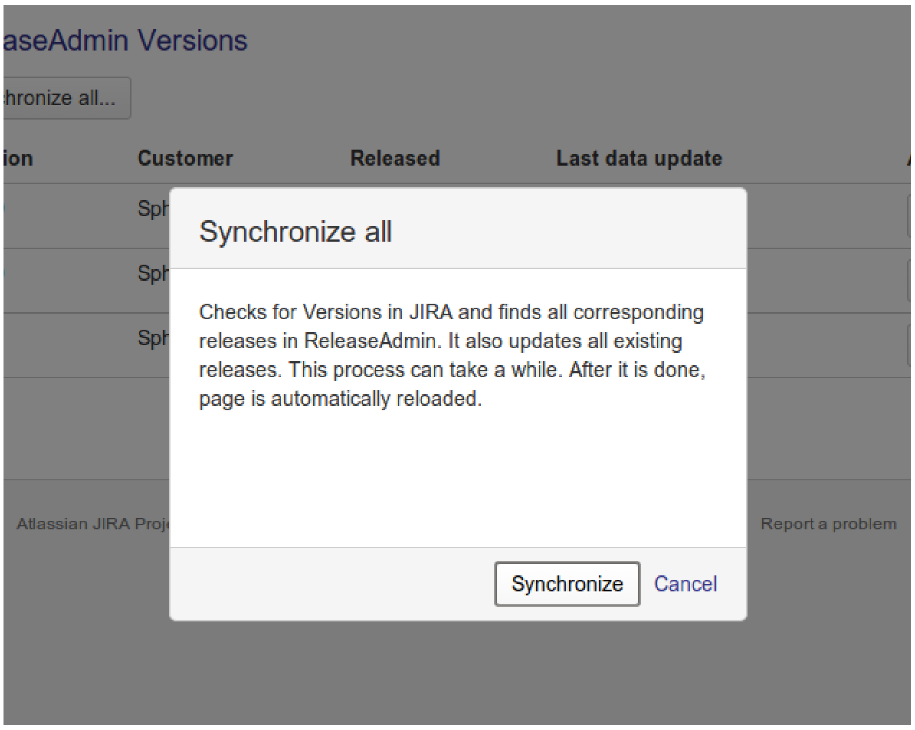
The tab allows user to see all cached versions in plugin. For the project administrators there are advanced features as synchronizing of all versions in Jira with ReleaseAdmin (Illustration 22) and off course a function that updates one particular release. The screenshot is visible on Illustration 21.



ReleaseAdmin Versions

Version	Customer	Released	Last data update	Action
0.1.9	Sphairon	false	04.06.2014 - 16:59	Update data
0.2.0	Sphairon	false	24.06.2014 - 18:51	Update data
0.2.1	Sphairon	false	04.06.2014 - 16:59	Update data

Illustration 21: Project tab with ReleaseAdmin releases in database



Synchronize all

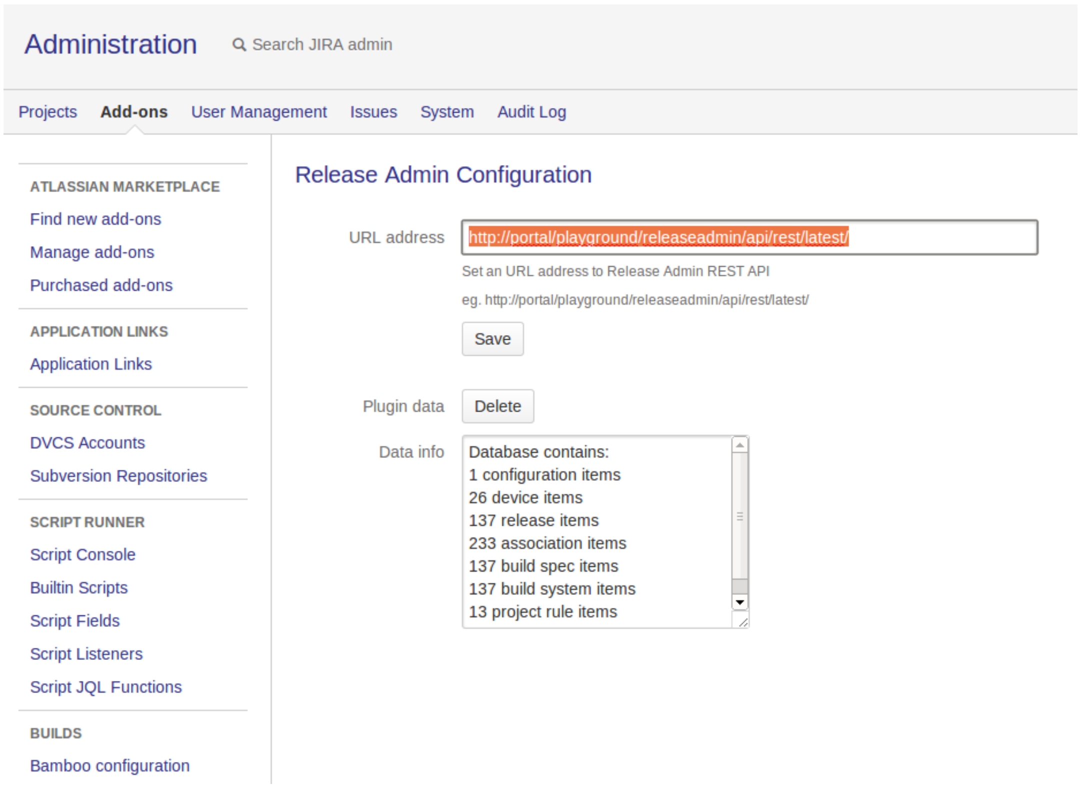
Checks for Versions in JIRA and finds all corresponding releases in ReleaseAdmin. It also updates all existing releases. This process can take a while. After it is done, page is automatically reloaded.

Illustration 22: Synchronize all dialog

Configuration

Configuration page is available only for JIRA administrators and allows them to configure plugin:

- ReleaseAdmin REST API path
- Cached items statistics
- Delete items in database



Release Admin Configuration

URL address: http://portal/playground/releaseadmin/api/rest/latest/

Set an URL address to Release Admin REST API

eg. http://portal/playground/releaseadmin/api/rest/latest/

Plugin data

Data info:

```
Database contains:
1 configuration items
26 device items
137 release items
233 association items
137 build spec items
137 build system items
13 project rule items
```

Illustration 23: Configuration page

Project mapping

Project mapping is very important to be done after plugin installation. Plugin needs to know which projects in ReleaseAdmin corresponds to JIRA projects. With versions it is simple, the name must be the same. But with projects it is more difficult, because project names are not always the same and sometimes one project in JIRA have to be connected with more projects in ReleaseAdmin (eg. general library development). Using project mapping view it is possible to connect all projects and it also offers automatic mapping functionality (4.1.7).

4.1.7. Features

Acquiring releases from ReleaseAdmin

Very important question is when to acquire data from ReleaseAdmin? The best answer is “When it is needed”. The idea is to provide user as much information as possible and in the time it is necessary. The approach is simple, if there is a version in JIRA browsing context which can have corresponding item in ReleaseAdmin, ask for it. If it is not in cache (local database), ask remote host (ReleaseAdmin) for a result. If the result is successful, show it to the user. And more if the user is administrator, allow possibility of refresh.

Version listener

Version listener is added functionality to JIRA, listener is called whenever new version is created. If release with corresponding name is found in ReleaseAdmin it is also added into database. This approach makes the plugin more automatic and comfortable.

String comparison

To map project between JIRA and ReleaseAdmin, the plugin uses special String comparison algorithm which comes out from Levenshtein distance function. This function calculates how many substitutions are necessary to reach conformity between two strings. String comparison algorithm uses this function but in little bit adjusted way. The algorithm is possible to see in Code 21.

Possible positive scenarios:

- returns true, if one string is a part of the second one (eg. Google Inc vs Google)
- returns true, if the number of changes necessary to get conformity are less than half size of longer string (eg. Googla Inc vs Google)

```
public static boolean checkSimilarity(String str1, String str2) {

     if (str1.contains(str2) || str2.contains(str1)) {
          return true;
     } else {
          float limit = 0.5f;
          int biggestSize = str1.length();
          if (str2.length() > biggestSize)
               biggestSize = str2.length();
          int changesNeeded = computeLevenshteinDistance(str1, str2);
          if(((float) changesNeeded / (float) biggestSize) <= limit){
               return true;
          }
     }
     return false;
}
```

Code 21: Check similarity method

4.1.8. Result

The plugin was tested during development on clean JIRA test instance. Introducing new features the plugin was everytime uploaded to JIRA staging instance. JIRA staging instance is a copy of live instance including all data inside. JIRA staging instance is for Sphairon employees available, so that is how the testing went.

Currently the plugin is deployed to the live instance of JIRA in Sphairon company. ReleaseAdmin for JIRA has 40 revisions created during 19 days of development, 4513 lines of codes (without empty lines) were written. Pure Java code is presented by 2410 lines, this is server side application. ReleaseAdmin for JIRA was first released in version 0.1.9, than version 0.2.0 was deployed to public live instance of JIRA, version 0.2.1 was immediately added repairing some crucial bugs announced by users. Version 0.2.2 is in future expected release to have some new features.

4.1.9. Future

For the next release, there are some ideas to extend the plugin. For example enhance the communication between ReleaseAdmin and ReleaseAdmin for JIRA. It would be nice, when a release is created in ReleaseAdmin, it would announce that to JIRA REST API by creating a version in JIRA, then the plugin would get detailed information about the release from ReleaseAdmin. The proposal of communication is described on diagram in Illustration 24.

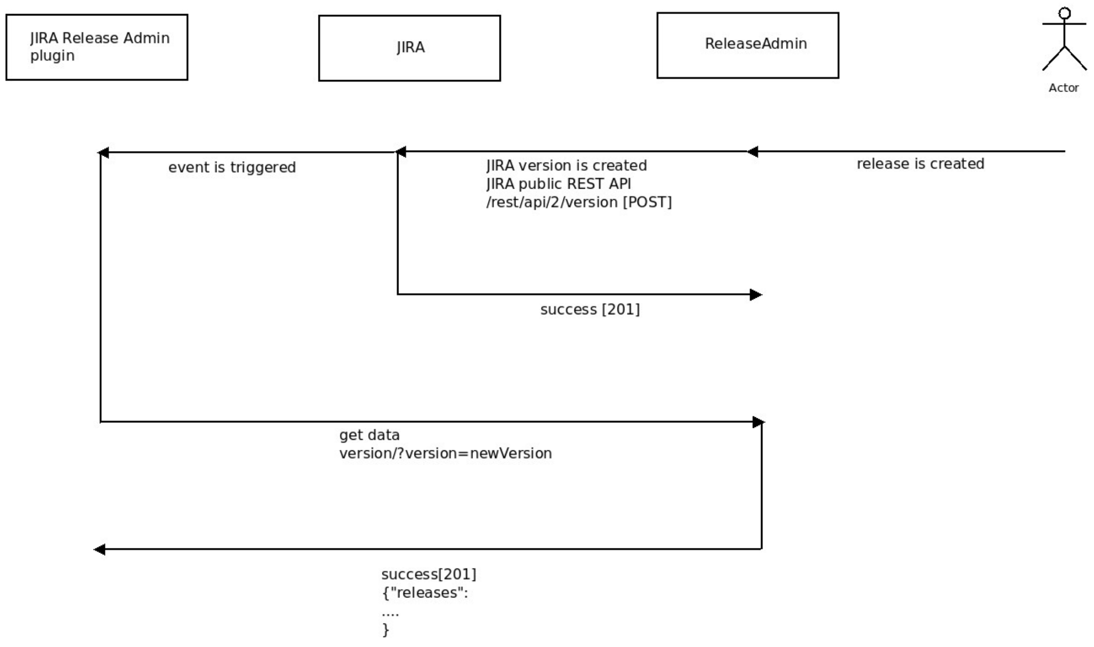


Illustration 24: Extension of communication

4.2. JIRA Release Management Plugin

4.2.1. Motivation

JIRA is one of the most often used bug tracking systems and still it doesn't have any plugin, which adopts release management functionalities. The JIRA itself supports version entities as a first step to introduce these abilities. However it is not enough. Competition is pretty rare, if you don't think about big commercial continuous integration systems, where the issue tracking is one small part of big colossus.

As a competition some standalone open source projects can be listed, for example:

- Apache Continuum
- Jenkins (build system with release management plugin)
- GitHub (host system on top of the GIT with bug tracking and release features)
- BitBucket

These applications are usually created on top of certain tool and don't support connection of multiple tools.

The idea is to create a plugin for users of Atlassian JIRA which combines most of the release management abilities, with possibilities in future to adopt other continuous integration features, so that the plugin can be a competition to projects like Apache Continuum or BitBucket, but running inside of JIRA.

The plugin should be general in its data model, so everyone can use it. For the Sphairon company the motivation is to replace own proprietary release management system (ReleaseAdmin) in future, and replace it with Jira Release Management Plugin (JRMP), so that the data are not presented in two systems (JIRA and ReleaseAdmin), but only in one.

4.2.2. Release Management Software

Release management software is a software, which takes care of managing the life time of release. With usage of continuous integration tools, it fulfills the role of the main information source for managers of software development. Main idea which is hidden behind the software is to cover and to store all steps and information, that are used during continues integration cycle, with a release procedure and with planning.

JIRA takes care of planning itself, it supports issues and versions to be created for upcoming releases. But the necessary information incoming from continues integration tools are missing. Release management software should gather these information to keep complex overview of software inside.

The first step to create successful release management software is to create general data model, which fits most of the use-cases. Since JIRA is a great project management tool for software development, it needs to be also a great platform to create successful release management software.

The second step is to connect all tools together, which are usually used in development and create general software which link existing development applications. The task is not to implement these tools from the scratch, but use them to be more connected, since they are indirectly connected anyway.

4.2.3. Specifications

- JRMP version extends version in JIRA
    - adds new information like revision control system, build system, milestone or customers
- Plugin must use most of JIRA common tools eg. Versions Administrator page
- Milestone is as a group for versions
- Milestone is connected to JIRA project
- Version can have only one milestone
- Version can have many customers
- Version has many targets and one revisions
- Since Version Overview page can not be extended, new Milestone page should do the trick with expandable number of versions inside
- Revision control systems, customers, build systems are global entities, they are not connected to project
- New menu should be added to provide overview and administration pages for new entities
- Panels Affect Versions and Fixed Versions on Issue page should contain compact JRMP version information
- Version Detail page should be added to have all information about JRMP version

4.2.4. Data Structure

Since Jira knows the data entity, which is called version, it is necessary that plugin works with it. Version in JIRA has few attributes, these are name, description, release date and start date. Version also knows about itself, if it was released or archived. It is not possible to inherit from version, because it is forbidden by Atlassian API. So another solution is to keep reference on version in our data model.

The comparison what attributes usual JIRA version offers is on Illustration 25. The proposal enhancements are quite hight, and JrmpVersion brings many new parameters to extend release management features.

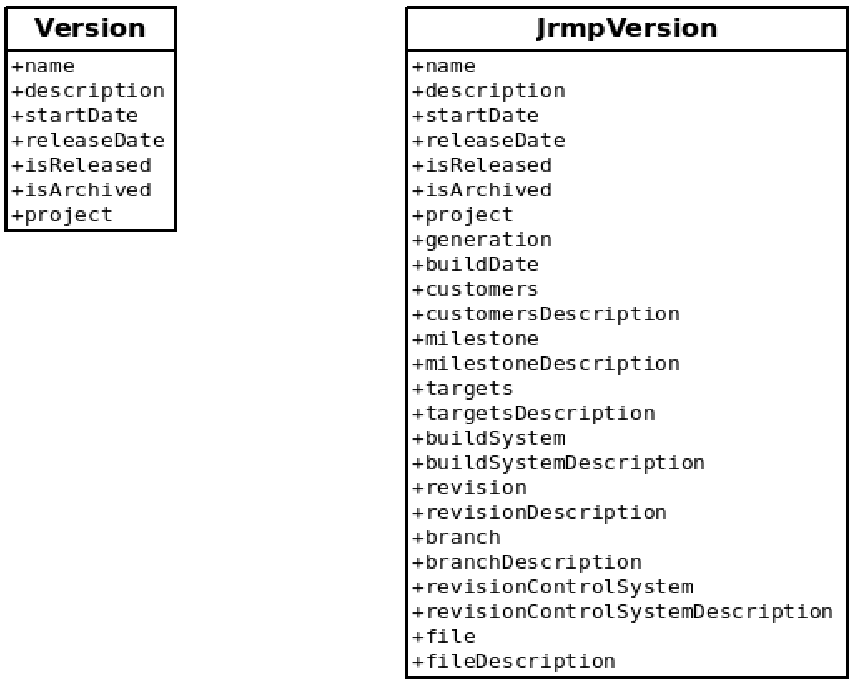


Illustration 25: Accessible fields JIRA version vs JRMP version

The overall data model of plugin can be seen on Illustration 26. JrmpVersion keeps reference on Jira version (by jiraVersionId field) in relation one-to-one. JrmpVersion owns reference for Customer data object. This relation is many-to-many. JrmpVersion can have many customers, and customer can have many versions. JrmpVersion has also reference to Milestone. Milestone knows into what Jira project (by jiraProjectId field) it belongs. JrmpVersion can have only one Milestone assigned, but Milestone can have many JrmpVersion. JrmpVersion has many targets as Target can reference to many JrmpVersions. This is a difference between software version information and deployment. There could be many deployments with the same software information. Target can have many stored files so as the JrmpVersion. But a single File has only on Target and one JrmpVersion.

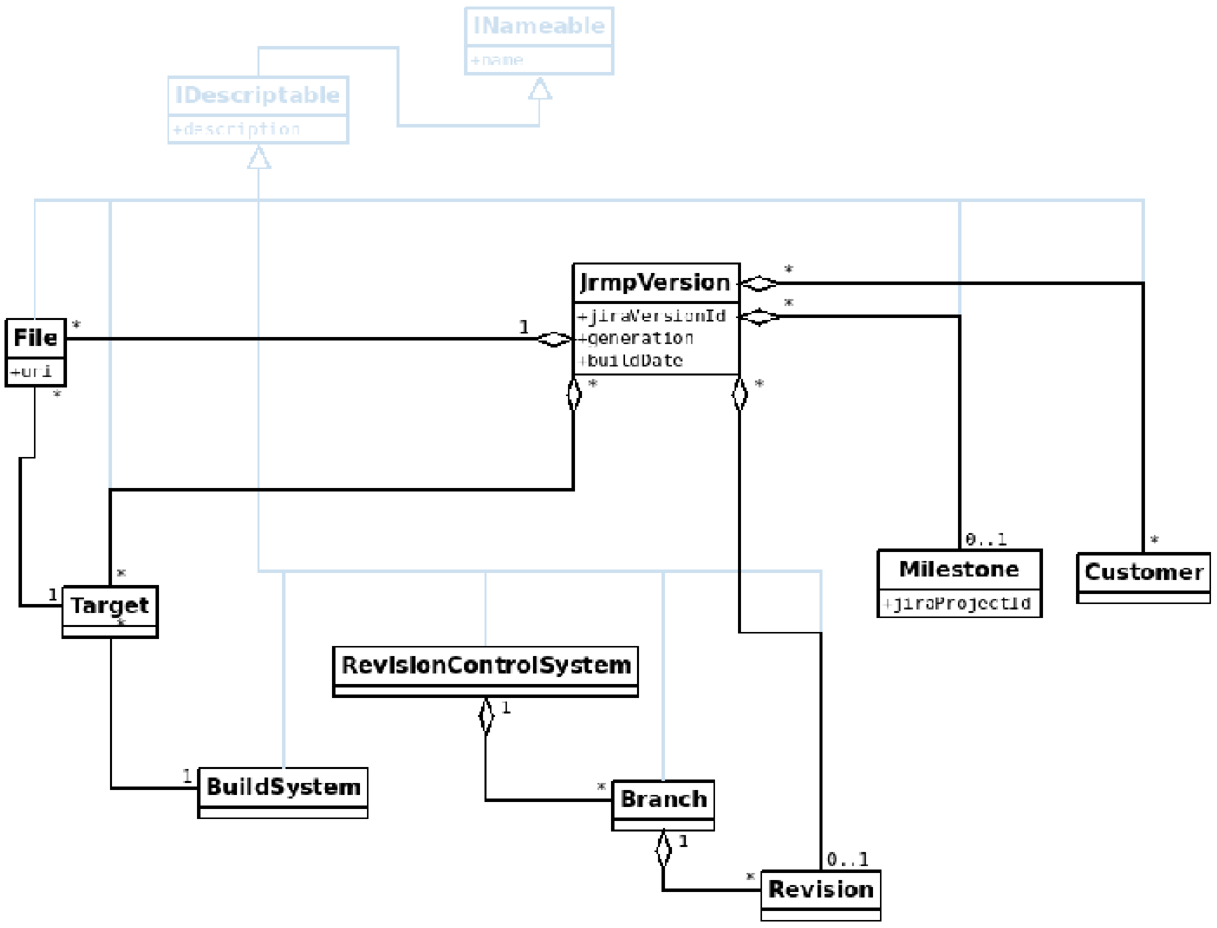


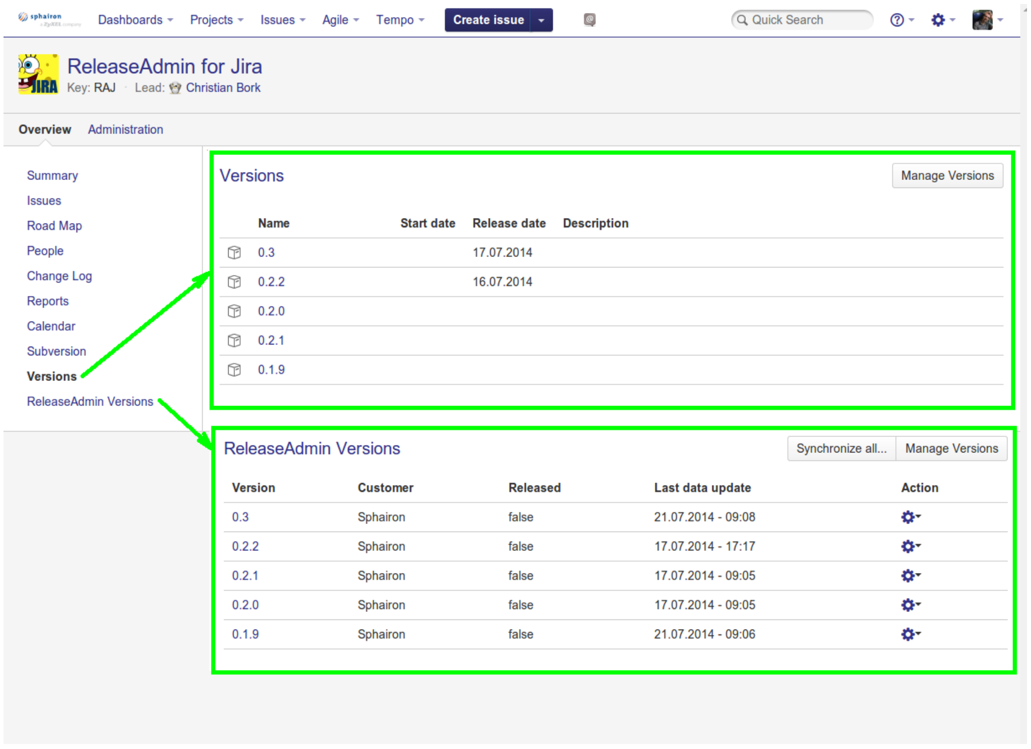
Illustration 26: Data model

4.2.5. Application Structure

Application structure should have some common elements as ReleaseAdmin for JIRA plugin (4.1.5). The panels (Illustration 17) which brings brief information about versions should stay, only the data contained need to be changed a bit. The version tab (Illustration 16) with detailed information about release also stays, but the content needs to change a lot, since the data are bit different.

New features need to be introduced, since the data have to be maintained and edited by users, it is important to add some kind of central overview and administrator pages, where all entities can be edited. These pages are global, they are not connected with issue or project in JIRA. In these pages you could edit for example customer, target, build system and others.

The duality from ReleaseAdmin for JIRA plugin (Illustration 27) is not very convenient, where there are two pages for versions. One contains usual JIRA versions and the other contains only these versions, which are cached in the database (correspond to ReleaseAdmin). This approach is the only solution for ReleaseAdmin for JIRA, however for JRMP plugin the solution can be better.



Versions

Name	Start date	Release date	Description
0.3		17.07.2014	
0.2.2		16.07.2014	
0.2.0			
0.2.1			
0.1.9			

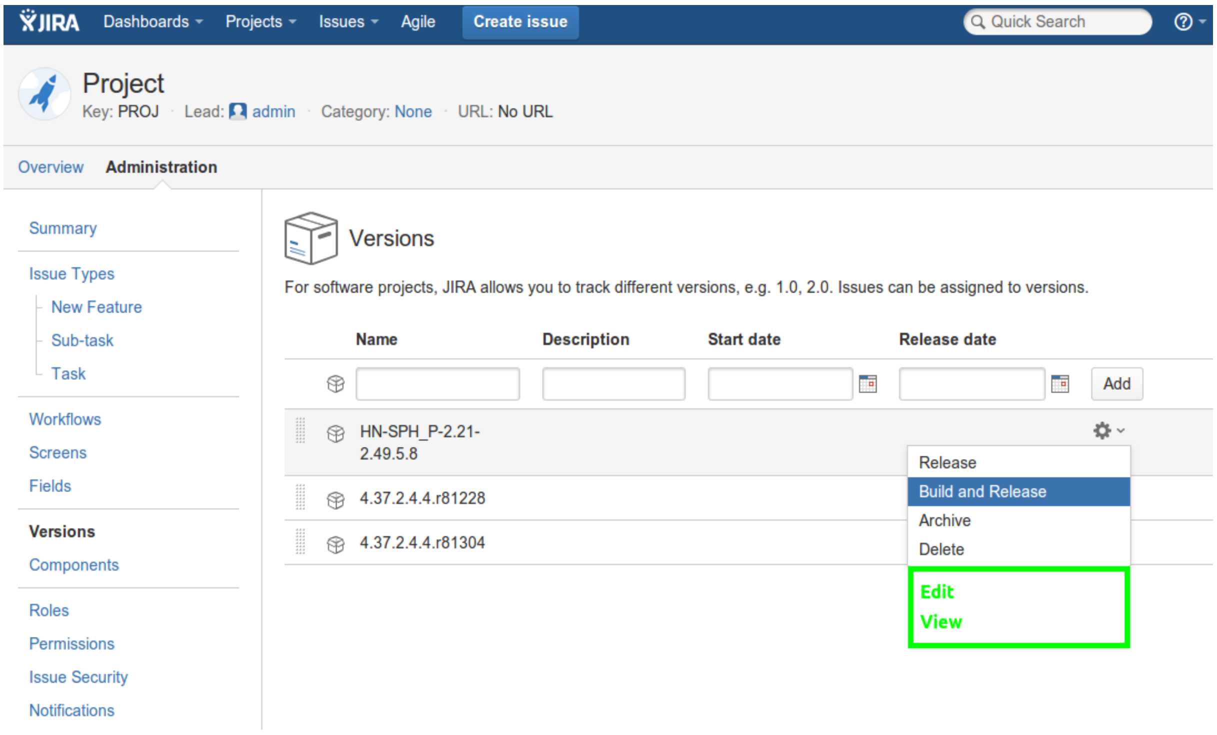
ReleaseAdmin Versions

Version	Customer	Released	Last data update	Action
0.3	Sphairon	false	21.07.2014 - 09:08	
0.2.2	Sphairon	false	17.07.2014 - 17:17	
0.2.1	Sphairon	false	17.07.2014 - 09:05	
0.2.0	Sphairon	false	17.07.2014 - 09:05	
0.1.9	Sphairon	false	21.07.2014 - 09:06	

Illustration 27: ReleaseAdmin for JIRA - Versions pages duality

An entity called milestones can be used for that. Milestone is something like a group for versions, usually it divides these versions which introduce new features. For the purpose of milestones the separate page needs to be created in context of JIRA project page. Since the milestone contains versions, it is easy to design the page as expandable milestone list with table of versions inside.

To bring new features to JIRA version, new data model needs to be implemented (4.2.4). With new data model, new editing features are necessary. Editation Luckily in compare of usual Versions overview page, this Version administrator page can be extended. And new items to context menu of each version can be added as it is possible to see on Illustration 28.



Key: PROJ · Lead: admin · Category: None · URL: No URL

For software projects, JIRA allows you to track different versions, e.g. 1.0, 2.0. Issues can be assigned to versions.

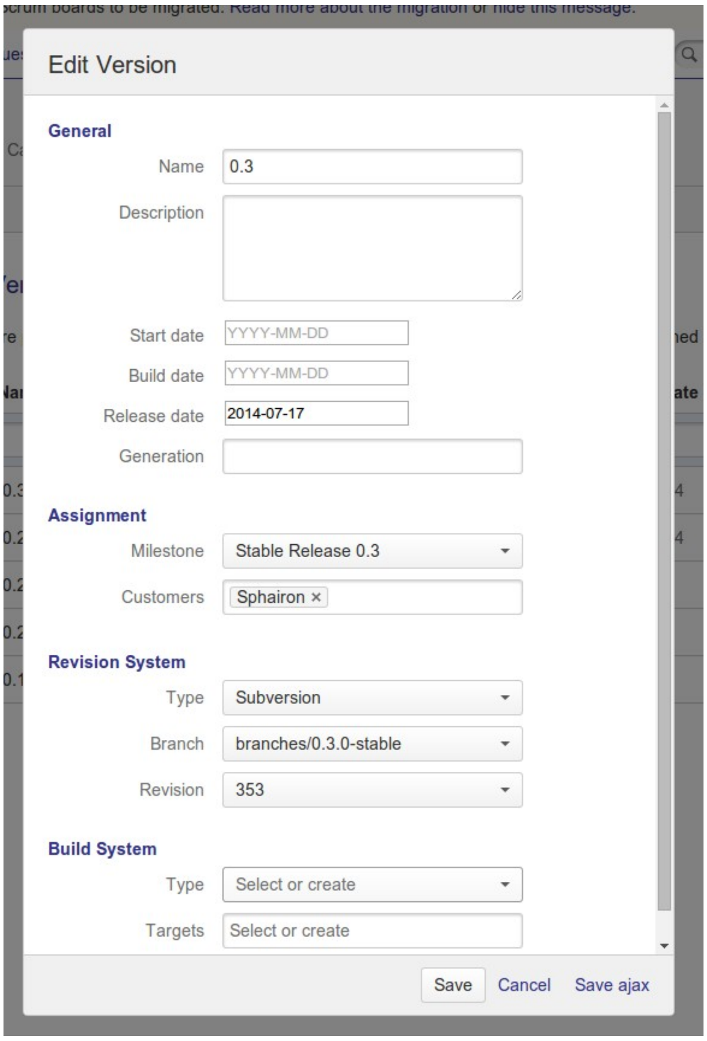
Name	Description	Start date	Release date
HN-SPH_P-2.21-2.49.5.8			
4.37.2.4.4.r81228			
4.37.2.4.4.r81304			

Illustration 28: Extended context menu

4.2.6. Views

Version edit page

Version edit page can be accessed from standard version list in administration section of project. Edit function allows to add desired values in comfortable way. Selection box uses Select2 library for compact view and easy modification. Select2 box enables searching, adding, removing of elements in given category. Version edit page (Illustration 29) brings the most important fields to usual JIRA version with nonviolent way. The entities are acquired by AJAX background communication.



Edit Version

General

Name	0.3
Description	
Start date	YYYY-MM-DD
Build date	YYYY-MM-DD
Release date	2014-07-17
Generation	

Assignment

Milestone	Stable Release 0.3
Customers	Sphairon ×

Revision System

Type	Subversion
Branch	branches/0.3.0-stable
Revision	353

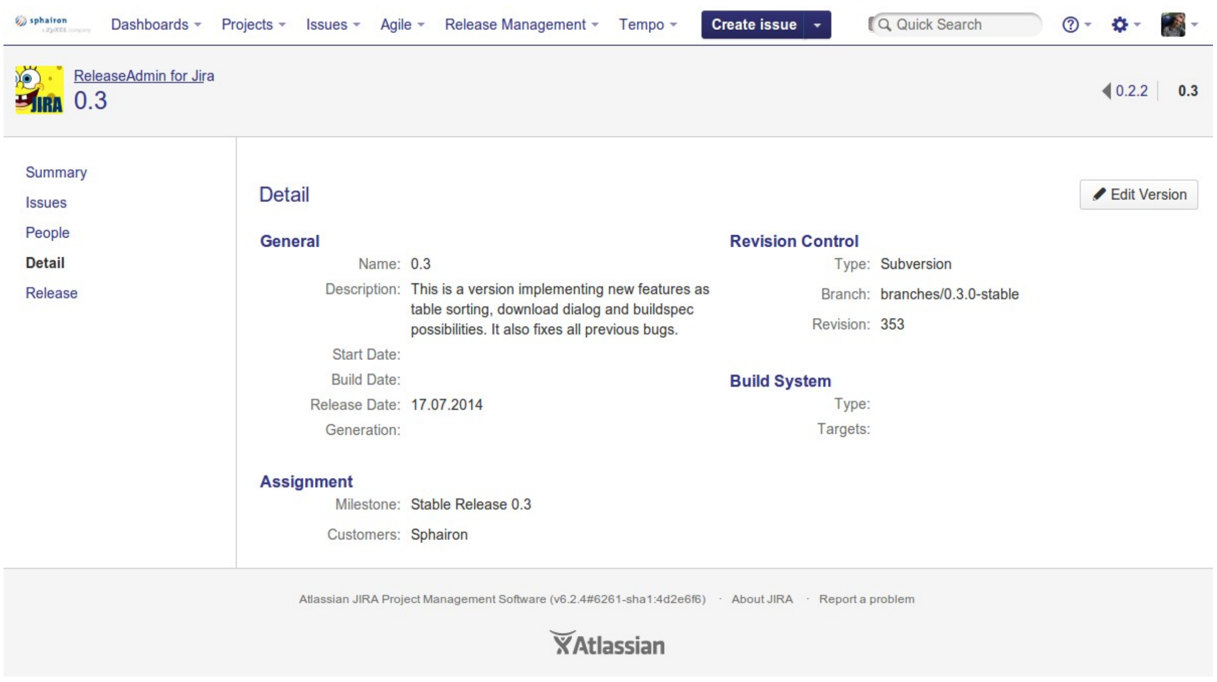
Build System

Type	Select or create
Targets	Select or create

Illustration 29: Version edit page

Version detail page

Version detail page can be observed on the Illustration 30. This page brings detailed view with all items assigned to the certain version. Usually every item can have also a description, the description field is hidden, if no text is inside. This ensures clear approach to show information. For the users with administration permissions, the edit button is visible. It shows Version edit page in dialog.



General

Name:	0.3
Description:	This is a version implementing new features as table sorting, download dialog and buildspec possibilities. It also fixes all previous bugs.
Start Date:	
Build Date:	
Release Date:	17.07.2014
Generation:	

Revision Control

Type:	Subversion
Branch:	branches/0.3.0-stable
Revision:	353

Build System

Type:	
Targets:	

Assignment

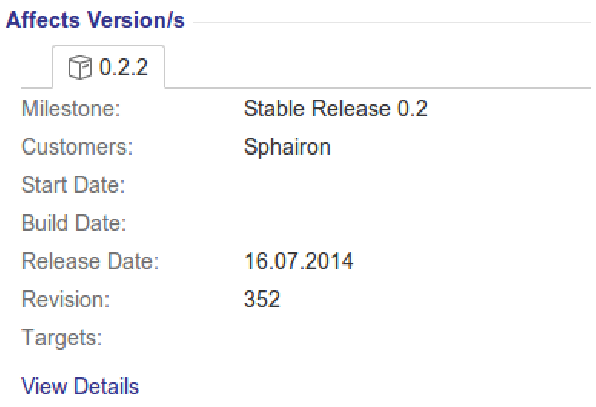
Milestone:	Stable Release 0.3
Customers:	Sphairon

Atlassian JIRA Project Management Software (v6.2.4#6261-sha1:4d2e6f6) · About JIRA · Report a problem

Illustration 30: Version detail page

Issue page panel

Issue page panel (Illustration 31) is very similar to that one in ReleaseAdmin for JIRA plugin. It shows little bit different information. It also brings new feature, that the icon is shown next to the version name. Icon marks if the version is released or not.



Affects Version/s

0.2.2

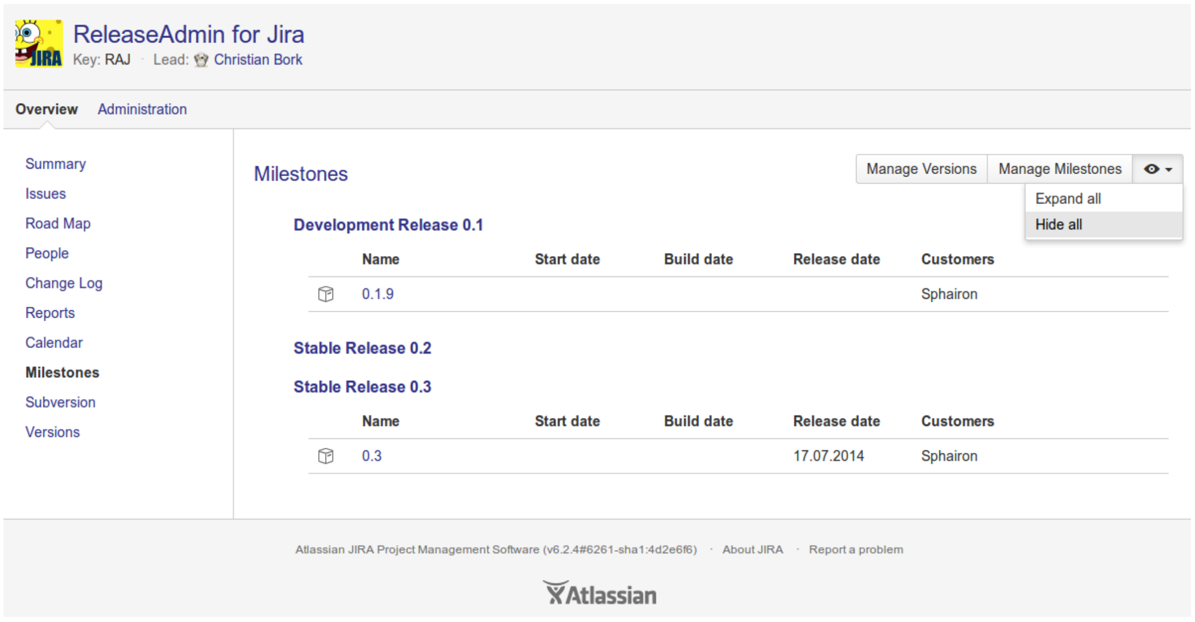
Milestone:	Stable Release 0.2
Customers:	Sphairon
Start Date:	
Build Date:	
Release Date:	16.07.2014
Revision:	352
Targets:	

View Details

Illustration 31: Affects Versions panel

Milestone overview page

Milestone overview page is visible to all users, it shows versions in JIRA with focus of milestone assignment. Archived versions are hidden. The example of page is shown on Illustration 32.



ReleaseAdmin for Jira
Key: RAJ · Lead: Christian Bork

Overview Administration

Summary
Issues
Road Map
People
Change Log
Reports
Calendar
Milestones
Subversion
Versions

Milestones

Manage Versions | Manage Milestones | Expand all | Hide all

Development Release 0.1

Name	Start date	Build date	Release date	Customers
0.1.9				Sphairon

Stable Release 0.2

Stable Release 0.3

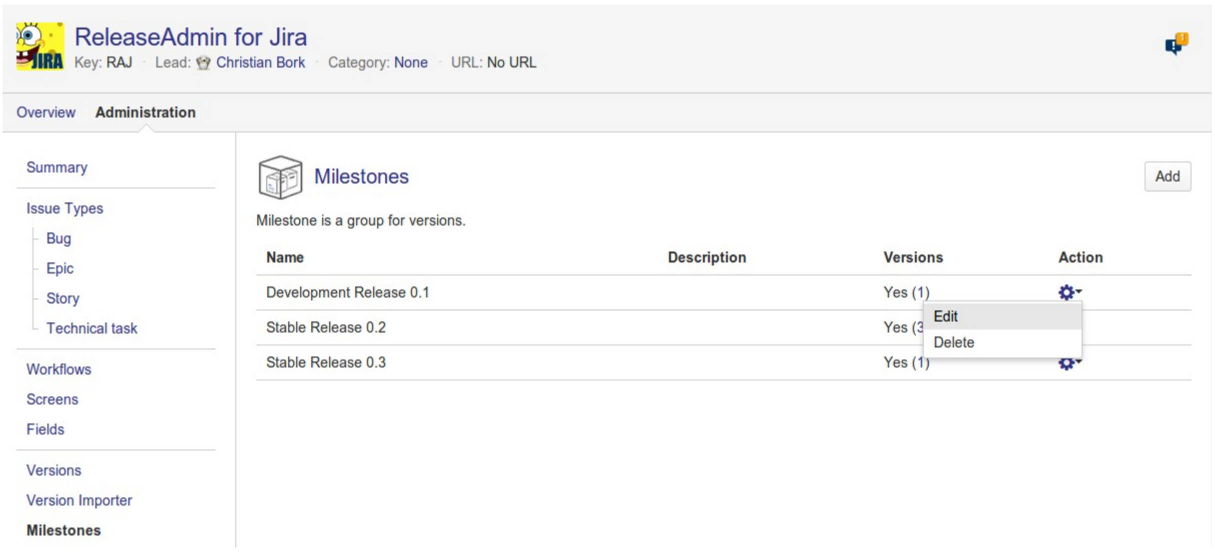
Name	Start date	Build date	Release date	Customers
0.3			17.07.2014	Sphairon

Atlassian JIRA Project Management Software (v6.2.4#6261-sha1:4d2e6f6) · About JIRA · Report a problem

Illustration 32: Milestone list page

Milestone administration page

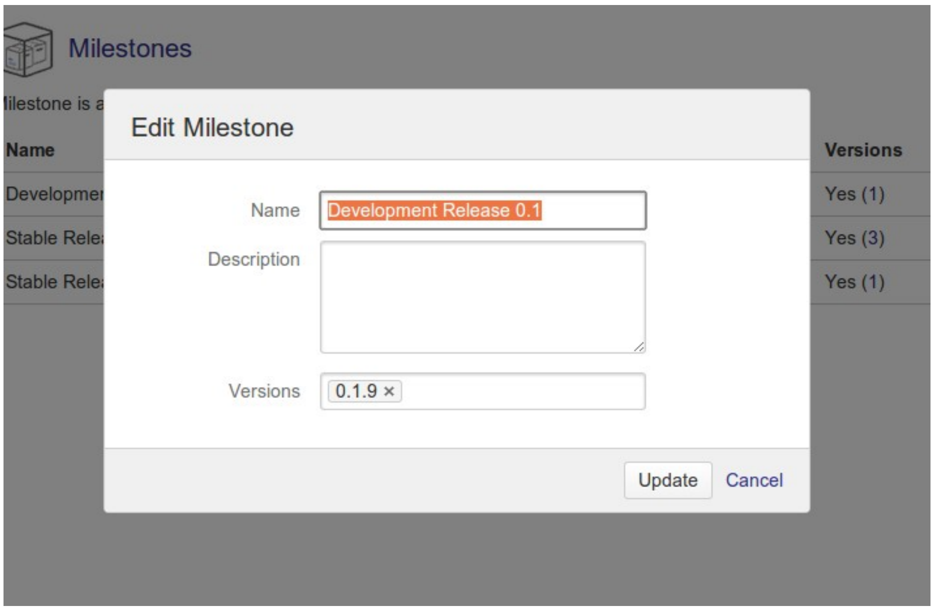
Milestone administration page (Illustration 33) shows a list of milestones with possibility of modification. The edit dialog (Illustration 34) shows also the versions assigned to the selected milestone. It is easy to add them or remove them. If the version has already another milestone assigned, it is properly displayed to the user. When it is necessary to delete a milestone, proper unassigned of all entities is done before, the versions inside are not deleted.



Milestone is a group for versions.

Name	Description	Versions	Action
Development Release 0.1		Yes (1)	
Stable Release 0.2		Yes (3	
Stable Release 0.3		Yes (1)	

Illustration 33: Milestone administrator page

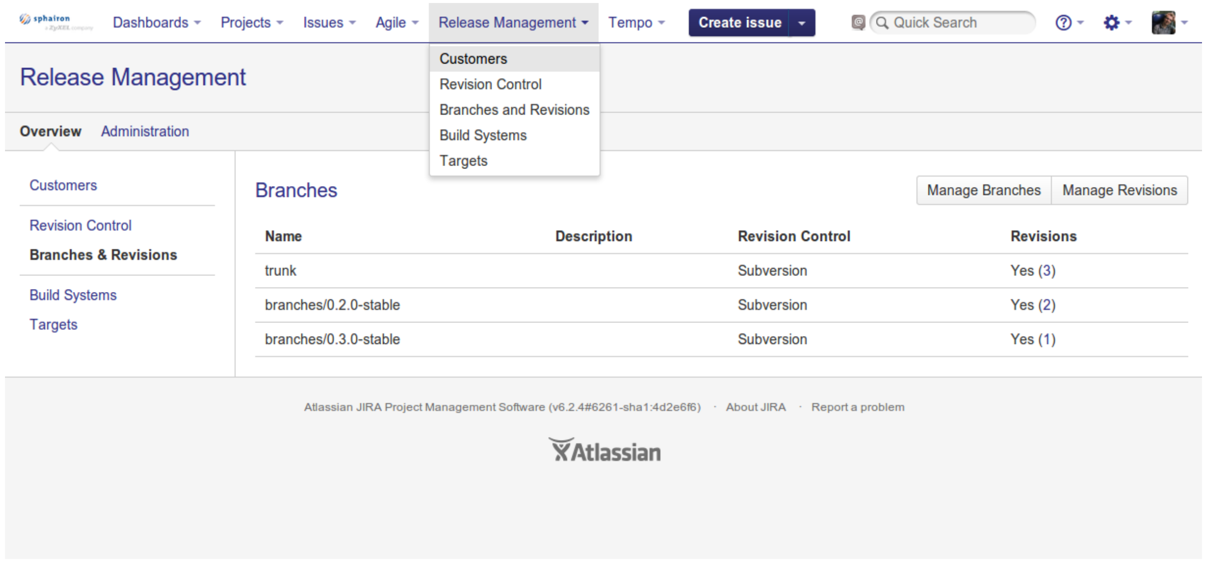


Name: Development Release 0.1
Description:
Versions: 0.1.9 ×

Illustration 34: Milestone modification dialog

Overview pages

There are many overview pages sharing the same same controller class. However the view is defined for each page separately. Overview pages show global entities in plugin for example customers, build systems, branches and revisions (Illustration 35) and others.



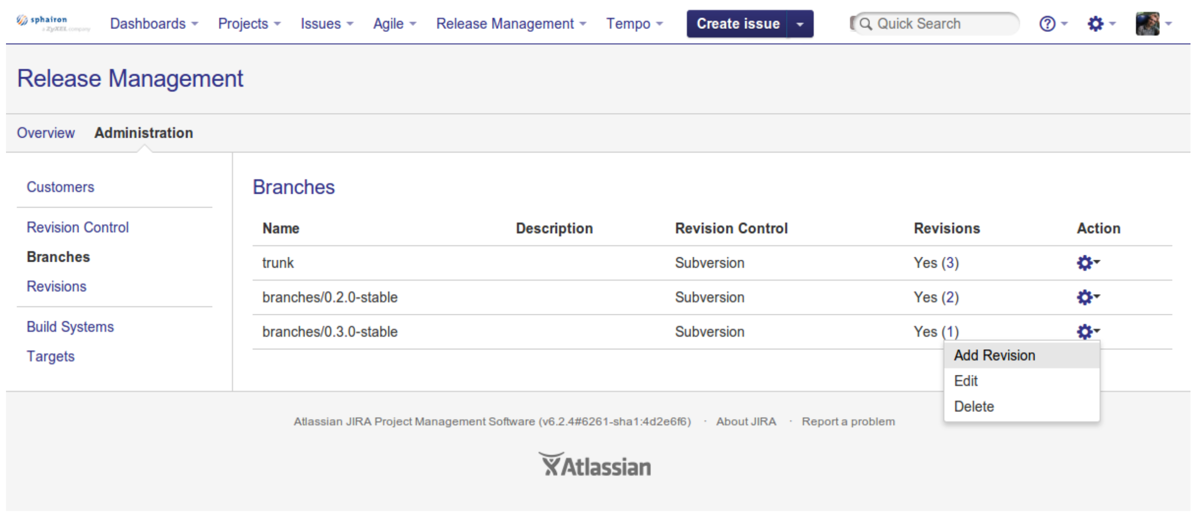
Branches

Name	Description	Revision Control	Revisions
trunk		Subversion	Yes (3)
branches/0.2.0-stable		Subversion	Yes (2)
branches/0.3.0-stable		Subversion	Yes (1)

Illustration 35: Branches and Revisions page

Administration pages

Administration are only visible for the users with certain permissions. They allow to edit existing global entities in plugin with possibility to add a new entity.



Branches

Name	Description	Revision Control	Revisions	Action
trunk		Subversion	Yes (3)	⚙
branches/0.2.0-stable		Subversion	Yes (2)	⚙
branches/0.3.0-stable		Subversion	Yes (1)	⚙

Illustration 36: Branches administration page

4.2.7. Result

JRMP plugin is successfully deployed to a JIRA staging instance in Sphairon for the testing of behavior. The plugin has extensive editing features to achieve user specific desires. It is as general as possible. It is a standalone release management plugin running in JIRA. It successfully extends current JIRA functionalities of release management and brings new desired attributes to JIRA versions.

However the plugin needs many features to be implemented in future to be successful as a release management software. The links with build systems and revision control systems are important. Current version allows only manually entering of data from these systems. Other helping functions would be convenient. For example automatic versioning or automatic deployment to reach continues delivery principles.

Jira Release Management Plugin has 10 revisions created during 23 days of development, 9373 lines of codes (without empty lines) were written. It shares with ReleaseAdmin for JIRA 859 lines of code. 3552 lines of code are in Java, the rest are views using Velocity Templates and JavaScripts. Java code contains 28,5% abstract and 71,4 % concrete entities.

5. Conclusion

Within the master thesis two plugins were created. One of them ReleaseAdmin for JIRA is regularly used by employees of the Sphairon company. It helps to solve daily situations with release management issues. The plugin is still in development mostly bringing new features in GUI and reflecting the comments within the company. In the ReleaseAdmin for JIRA number of new technologies I had to learn to provide the best functionality and the best user experience.

Using the skills from developing ReleaseAdmin for JIRA another plugin was designed and developed. The Jira Release Management Plugin is planned to replace proprietary ReleaseAdmin application in future. The plugin should be also introduced to Atlassian Marketplace to offer another companies to enjoy enhanced release management features inside the JIRA system. The plugin still needs development to be used on live systems as a proper release management software. It requires to implement some other suitable functions in future.

During the development of newer versions of ReleaseAdmin for JIRA I could use the plugin itself to manage its versions. This way I was introduced to release management process. I found the approach so practical, I was pleased it happened and I am going to use these experiences in future. I was also introduced to continuous principles of developing software, which I partly used on case of my plugins. I met new tools like Subversion, Jenkins and other which helped me with proper development techniques. I also enjoyed excellent knowledges of my consultants and leaders in Sphairon, who always brought appropriate criticism into my work.

Literature

[1] Algorithm Implementation/Strings/Levenshtein distance. In: Wikibooks: open books for an open world [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-07-21]. Available on: http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance

[2] Apache Velocity: Velocity User Guide. THE APACHE SOFTWARE FOUNDATION. The Apache Velocity Project [online]. 2007 [cit. 2014-07-21]. Available on: https://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html

[3] Caching Guidance. MICROSOFT. Developer Network [online]. 2014 [cit. 2014-07-21]. Available on: http://msdn.microsoft.com/en-us/library/dn589802.aspx

[4] Component Plugin Module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/JIRADEV/Component+Plugin+Module

[5] FOWLER, Martin. Continuous Integration. [online]. 2006 [cit. 2014-07-21]. Available on: http://martinfowler.com/articles/continuousIntegration.html

[6] JacksonInFiveMinutes. FASTERXML, LLC. Jackson [online]. [cit. 2014-07-21]. Available on: http://wiki.fasterxml.com/JacksonInFiveMinutes

[7] JAMESON, Rosie. Component Import Plugin Module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Dostupné z: https://developer.atlassian.com/display/JIRADEV/Component+Import+Plugin+Module

[8] Java API for RESTful Services. Java.net [online]. [cit. 2014-07-21]. Available on: https://jax-rs-spec.java.net/

[9] Jersey 2.10.1 User Guide. ORACLE CORPORATION. [online]. [cit. 2014-07-21]. Available on: https://jersey.java.net/documentation/latest/index.html

[11] J Tricks: Little JIRA Trics [online]. 2014 [cit. 2014-07-21]. Available on: http://www.j-tricks.com/

[12] KOŠEK, Jiří. XML schémata. [online]. 2013 [cit. 2014-07-21]. Available on: http://www.kosek.cz/xml/schema/

[13] NECHT, Andreas. Version Tab Panel Plugin Module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/JIRADEV/Version+Tab+Panel+Plugin+Module

[14] KURUVILLA, Jobin. JIRA 5.x development cookbook: this book is your one-stop resource for mastering JIRA extensions and customizations. Birmingham, U.K.: Packt Publishing, 2013, 1 online source (v, 491 p.). ISBN 978-1-78216-908-6.

[15] LE BERRIGAUD, Samuel. Pure-Java Object Relational Mapping. Java.net [online]. 2011 [cit. 2014-07-21]. Available on: https://java.net/projects/activeobjects/pages/Home

[16] MADDOX, Sarah. REST Plugin Module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/DOCS/REST+Plugin+Module

[17] MADDOX, Sarah. Atlassian User Interface (AUI) Developer Documentation. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/AUI/Atlassian+User+Interface+(AUI)+Developer+Documentation

[18] MALÝ, Martin. REST: architektura pro webové API. Zdroják [online]. 2009 [cit. 2014-07-21]. Available on: http://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[19] Project Tab Panel Plugin Module. JIRADEV [online]. [cit. 2014-07-21]. Available on: http://jiradev.com/project-tab-panel.html

[20] RAGOZIN, Alexey. Data Grid Pattern - Proactive caching. [online]. 2011 [cit. 2014-07-21]. Available on: http://blog.ragozin.info/2011/10/grid-pattern-proactive-caching.html

[21] SPIEWAK, Daniel. ActiveObjects: An Easier Java ORM. Javlobby [online]. [cit. 2014-07-21]. Available on: http://www.javalobby.org/articles/activeobjects/

[22] SPIEWAK, Daniel. An Easier Java ORM Part 2. Code Commit [online]. [cit. 2014-07-21]. Available on: http://www.codecommit.com/blog/java/an-easier-java-orm-part-2

[23] SPIEWAK, Daniel. An Easier Java ORM: Relations. Code Commit [online]. [cit. 2014-07-21]. Available on: http://www.codecommit.com/blog/java/an-easier-java-orm-relations

[24] Servlet Plugin Module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/JIRADEV/Servlet+Plugin+Module

[25] TURNER, Jeff. Webwork plugin module. ATLASSIAN. Atlassian Developers [online]. [cit. 2014-07-21]. Available on: https://developer.atlassian.com/display/JIRADEV/Webwork+plugin+module

[26] WebWork. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-07-21]. Available on: http://en.wikipedia.org/wiki/WebWork

[27] XML Schema (W3C). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2003, 2014 [cit. 2014-07-21]. Available on: http://en.wikipedia.org/wiki/XML_Schema_(W3C)