Zanata 1

Zanata Quickstart Guide

Quickstart guide for Zanata

Publican

BOOK PUBLISHING TOOL

Joshua Wulf

Zanata 1 Zanata Quickstart Guide
Quickstart guide for Zanata
Edition 0

Author Joshua Wulf jwulf@fedoraproject.org

This book contains quickstart guides for the Red Hat, Fedora, and JBoss Zanata instances.

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;
```

```
public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

Part I. Developer and Writer Quickstart

Introducing Zanata

Zanata is an online system for managing localization projects. *Localization* is the process of translating software interfaces or documentation into the various local languages used around the world.

Zanata allows a software developer or a documentation writer to provide their strings, have those strings localised into specific target languages by a community of translators, then retrieve the localized strings to build a localized version of the software or documentation.

The Zanata server implements a REST interface for pushing and pulling source and translated strings, an web-browser based translation editor, and a server-wide translation memory for translation reuse.

Cross-platform Zanata clients are available written in Python and Maven.

Prerequisites for using a Zanata instance

2.1. Prerequisites for using the Fedora Zanata instance

Procedure 2.1. Create a Fedora Account System (FAS) account

You need a FAS (Fedora Account System) account to use the Fedora Zanata instance. Follow this procedure to create a FAS account if you do not already have one.

1. Go to *https://admin.fedoraproject.org/accounts/*
2. Click "New Account"
3. Fill in your details and sign up

Result:

You have a FAS account for use with the Fedora Zanata instance.

Procedure 2.2. Validate your Fedora Zanata account

1. Log in to *https://fedora.zanata.org* using your FAS account.
2. The first time you sign in you are presented with an "Edit Profile" form pre-filled with your username and a suggested email address. Make sure that the email address is correct, modify it if necessary, and click "Save".
3. Check your email for the validation email. Click the link in the email to validate your FAS account for access to Zanata.

Procedure 2.3. Configure your local machine with credentials for fedora.zanata.org

1. Use your FAS credentials to login at *https://fedora.zanata.org*.
2. Click "My Profile"
3. If you do not have a current API key, click "Regenerate API key"
4. Copy the API key value
5. Create the file **`~/.config/zanata.ini`**
6. Paste the following into the file, replacing the username and key values with your FAS username and the API key, respectively:

```
[defaults]
debug = false
[servers]
fedora.url = https://fedora.zanata.org/zanata
fedora.username = <your FAS name>
fedora.key = <API key>
```

Result:

Your machine is now configured to communicate with the Fedora Zanata instance.

2.2. Install a Zanata client

A Zanata client is required to push or pull strings from the Zanata server. It is not required for doing translation work.

There are two Zanata clients. One is written in python, one is written in maven. Java software developers who use maven to build their software will be . The Python client is packaged as an rpm for Red Hat Enterprise Linux and Fedora, and recommended for managed environments. Windows and Mac OS X users are advised to use the maven client.

2.2.1. Python Client

Procedure 2.4. Install the Zanata python client

1. If you are using Fedora 15, issue the following command with root privileges:

```
yum install zanata-python-client
```

2. If you are using a version of Red Hat Enterprise Linux, you can obtain the client from *EPEL (Extra Packages for Enterprise Linux)*[1].

   a. Enable the EPEL repository following the instructions at *EPEL (Extra Packages for Enterprise Linux)*[2].

   b. Issue the following command with root privileges:

```
yum install zanata-python-client
```

2.2.2. Maven Client

Coming soon

[1] http://fedoraproject.org/wiki/EPEL
[2] http://fedoraproject.org/wiki/EPEL

Request a Zanata translation project

3.1. Request a translation project for the Fedora Zanata instance

Procedure 3.1. Create a request for a Zanata translation project for your book

1. Send an email to zanata-requests@redhat.com
2. Set the subject to "New Translation Project"
3. In the body give:
   - the name of your project
   - the name of the book or package
   - your FAS username
   - the FAS names of any additional maintainers

Result:

An administrator creates a translation project and makes you a maintainer. You receive an email when your translation project is ready. Log out of Zanata and log in again to ensure that all security permissions are updated.

Configure a Zanata translation project

Procedure 4.1. Select target languages for the project

1. When your translation project is created, log in to the Zanata instance.
2. Click Projects
3. Locate your translation project, and click it
4. On your project's page, click "Edit Project"
5. Move desired languages to "Selected Locales"
6. Click "Update"

Result:

You return to the translation project page

Procedure 4.2. Create a project version

1. When your translation project is created, log in to the Zanata instance.
2. Click Projects
3. Locate your translation project, and click it
4. On your project's page, click "Create Version"
5. Provide a Version ID in the ID field. The Version ID is an arbitrary value that is used to match a set of translations in Zanata with an svn tag, package version, or book revision.

Optionally:

Specify a customised list of target languages that overrides the project setting. This is used to restrict the target languages to a specific subset.

7. Click "Save"

Translate a Publican project in Zanata

Publican[1] is a popular open source Docbook publishing tool chain. It is used for writing documentation. Zanata and Publican - two great flavours that taste great together.

5.1. Publican project integration

Follow this procedure to integrate a Publican book with a Zanata instance for translation.

Procedure 5.1. Configure your Publican project for translation in Zanata

1. Complete the prerequisites in *Chapter 2, Prerequisites for using a Zanata instance* and *Chapter 4, Configure a Zanata translation project*.
2. Log in to the Zanata instance.
3. Click "Projects"
4. Locate your translation project, and click it
5. On your project's page, click "Configure file" to download the **`zanata.xml`** file
6. Copy the **`zanata.xml`** file to the directory containing your book's **`publican.cfg`** file

Procedure 5.2. Create publican translation files for Zanata

- In the directory containing publican.cfg, run the command:

```
publican update_pot
```

5.2. Push source strings and existing translations

Procedure 5.3. Push Publican .pot files to Zanata for translation

- In the directory containing **`zanata.xml`**, run the command:

For zanata-python-client 1.2.4:

```
zanata publican push
```

For zanata-python-client 1.2.6+:

```
zanata push
```

Result:

The strings are pushed to the Zanata instance and are available for translation.

[1] https://fedorahosted.org/publican/

Translate a software package in Zanata

Zanata can translate any software package that provides its source strings in GNU Gettext format as a **.pot** file, and can consume translated **.po** files.

6.1. Software project integration

Procedure 6.1. Configure your software project for translation in Zanata

1. Complete the prerequisites in *Chapter 2, Prerequisites for using a Zanata instance*, *Chapter 3, Request a Zanata translation project*, and *Chapter 4, Configure a Zanata translation project*.
2. Log in to Zanata.
3. Click "Projects"
4. Locate your translation project, and click it
5. On your project's page, click "Configure file" to download the **zanata.xml** file
6. Copy the **zanata.xml** file to the directory that contains your project's **.pot** file
7. Add the **zanata.xml** file to the project's source control.

6.2. Push source strings and existing translations

You can use Zanata to obtain translations for any software project that provides its source strings in GNU Gettext **.pot** file format and can consume translated **.po** files.

Procedure 6.2. Push software project .pot file(s) to Zanata for translation

Follow this procedure to push a project's source strings to Zanata for translation. If you have existing translations you wish to push to the server along with source strings, refer to *Procedure 6.3, "Push software project .pot file(s) and existing translations"*.

- In the directory containing **zanata.xml**, run the command:

For zanata-python-client 1.2.4:

```
zanata po push --srcdir .
```

For zanata-python-client 1.2.6+:

```
zanata push
```

Result:

All **.pot** files in the **srcdir** are pushed to the Zanata server and are available for translation.

Source string merging

If you push a newer version of source strings to an ongoing Zanata translation project with translation already underway, the server will merge any existing translations for that project to preserve translation work so far. Any source language strings that had translations on the server before the push will have them after the push.

Procedure 6.3. Push software project .pot file(s) and existing translations

Use this procedure to push a project's strings to Zanata along with existing translations.

- In the directory containing **`zanata.xml`**,

For zanata-python-client 1.2.4:

```
zanata po push --srcdir . --import-po --transdir ./po
```

run the command:

For zanata-python-client 1.2.6+:

```
zanata push --push-trans --transdir ./po
```

run the command:

Substitute the location of your existing **`.po`** files as the value for **`--transdir`**. Optionally, specify a merging schema if you are pushing **`.po`** file translations into an ongoing Zanata translation project with existing server-side translations. See *Target string merging*.

Result:

All **`.pot`** files in the **`srcdir`** are pushed to the Zanata server, along with all existing translations in **`.po`** files in the **`transdir`**.

Target string merging

If you push **.po** translations to an ongoing translation project that has translations on the server, you have two different merging schemas.

Merge Auto schema

The Merge Auto is the default schema, and is used when no schema is specified, or when the **--merge auto** command switch is specified. The Merge Auto schema will add all new translations from the **.po** file (translations added to the **.po** file since the last import). So, if translations have been done offline in **.po** files, you can contribute these translations to your Zanata translation project. Where a string has been updated on the server, but remains unchanged in the **.po** file since the last import, the server-side translation is preserved. When a translation has been updated in both the **.po** file and the server since the last import, the **.po** file translation will overwrite the server-side translation.

Merge Import schema

The merge import schema is used when the **--merge import** command switch is supplied. The Merge Import schema will overwrite all server-side translations with the **.po** files contents. Where a server-side translation exists and no translation exists in the **.po** file, the server-side translation is lost.

View translation project status in Zanata

Procedure 7.1. Viewing translation status

1. Log in to the Zanata instance.
2. Click Projects
3. Locate your translation project, and click it
4. On your project's page, click "View Status" next to the project version you want to check.

Result:

You are taking to the statistics page for that version of your project. Here you can see how many words have been uploaded, and view the percentage completion for each of the target languages.

Pull translations from Zanata

You can pull translations from the Zanata server at any stage of translation completion above 0%.

The Zanata client makes the translations available on the local file system as GNU Gettext **.po** files. You may opt to keep these files in the source control system you use for your for your project source, or you may use them as transient files only by adding the pull as a step in your build process.

8.1. Pull translations for a Publican project

Procedure 8.1. Pull translations for a Publican project

- In the directory of your Publican project that contains the **publican.cfg** and **zanata.xml** files, run the following command:

For zanata-python-client 1.2.4:

```
zanata publican pull
```

For zanata-python-client 1.2.6+:

```
zanata pull
```

Result:

Translations above 0% are pulled from the Zanata and written as **.po** files in the Publican project.

8.2. Pull translations for a software project

Procedure 8.2. Pull translations for a software project

- In the directory containing **zanata.xml**, run the following command:

For zanata-python-client 1.2.4:

```
zanata po pull --dstdir ./po
```

Substitute the directory where you would like the .**po** files to be written as the value for **dstdir**.

For zanata-python-client 1.2.6+:

```
zanata pull --transdir ./po
```

Substitute the directory where you would like the .**po** files to be written as the value for **transdir**.

Part II. Translator Quickstart

Contributing translations

9.1. Pre-requisites

All you need to contribute translation to Zanata are:

1. A web browser
2. A validated Fedora Account System (FAS) account
3. A second language

Most modern web browsers support the Zanata online translation editor functionality.

To obtain and validate a FAS account refer to *Procedure 2.1, “Create a Fedora Account System (FAS) account”* and *Procedure 2.2, “Validate your Fedora Zanata account ”*.

9.2. Join a language team

A Language Team groups translators together through a common language. Language Teams can work simultaneously on translating a single document into their respective languages. All translations and all revisions are kept with each respective Language Team without affecting any other Team.

To get into the action you need to join a Language Team. Zanata will only show you the translation projects that correspond to your Language Teams. No Language Team, no visible projects.

Procedure 9.1. Join a Language Team

1. Sign in to Zanata
2. Click the "Language" tab
3. Click the Language Team you want to join
4. Click "Join Language Team"

Result:

You join that Language Team, and the Language Team is added to "My Languages" under the "My Profile" tab.

Procedure 9.2. Leave a Language Team

1. Sign in to Zanata
2. Click "Language" or "My Profile"
3. Click the Language Team you want to leave
4. Click "Leave Language Team"

Result:

You are removed from that Language Team.

Appendix A. Revision History

Revision 0-2 **Wed Aug 10 2011** **Joshua Wulf** *jwulf@redhat.com*

Added zanata-python-client 1.2.6 info and Translator Quickstart

Revision 0-1 **Wed Aug 10 2011** **Joshua Wulf** *jwulf@redhat.com*

First publish to web

Revision 0-0 **Tue Jul 5 2011** **Joshua Wulf** *jwulf@redhat.com*

Initial creation of book by publican

Index

F