

Generic internationalization and localization strategy for FLOW3

Google Summer of Code Proposal

by Karol Gusak

Table of Contents

Abstract.....	3
Project proposal.....	3
Locale object and automatic locale detection.....	3
Locale-aware resources and message formatting.....	4
Input parsers and validators.....	5
Other i18n / l10n topics.....	5
Deliverables.....	6
Timeline.....	6
Prior May 24, 2010.....	6
May 24 - 30, 2010.....	6
May 31 - June 6, 2010.....	6
June 7 - 13, 2010.....	6
June 14 - 20, 2010.....	6
June 21 - 27, 2010.....	6
June 28 - July 4, 2010.....	6
July 5 - 11, 2010.....	7
July 12-18, 2010.....	7
July 19-25, 2010.....	7
July 26 - August 1, 2010.....	7
August 2 - 8, 2010.....	7
August 9 - 15, 2010.....	7
August 16 - 20, 2010.....	7
About me.....	8

Abstract

Good internationalization and localization support is very important for a web application framework. However, most PHP frameworks addresses only basic problems due to the extensiveness of the topic. Because of very loose-coupled architecture and high quality standards, FLOW3 is the best framework for implementing advanced, clean i18n and l10n solution. I propose to create this kind of solution as my Google Summer of Code 2010 project.

Project proposal

Locale object and automatic locale detection

Locale class provides uniform representation of chosen locale. Default instance of this class exists and can be provided by framework in any time, describing framework-wide locale currently chosen as active. However, there is possibility to create more objects if needed (e.g., when multiple translations are needed for one request).

Object of Locale class can be provided to any method which depends on locale (e.g., formatter classes). However, this is optional parameter, and if not provided, default Locale object will be used.

As a part of i18n / l10n subsystem utils, methods for automatic locale detection are available. Best matching locale is being selected taking into account HTTP headers, user's preferences (session scope), manually provided locale name, and list of locales available in corresponding FLOW3 installation.

Locales' names (tags) are represented in well known format: `ll_CC.@variant`, where:

- `ll` is a two-letter language code, as in ISO 639-2
- `CC` is a two-letter country code, as in ISO 3166
- `@variant` is a sub-tag (or sub-tags) indicating additional variations of locale
- `@variant` and `CC` can be omitted, so that `ll`, and `ll_CC` locale tags are correct

Locales' tags are used to organize locale-aware resources in filesystem. Depending on the type of the resource, they are placed in package's private resource directory (e.g. translated messages) or public resource directory (e.g. image files).

Locales are hierarchical, with root locale being most generic one, and always present. When locale resource is being retrieved, first the most specific locale is being searched, then less specific one, and so on up to the root locale.

When new locale object is being requested, framework searches for most suitable locale in a list of locales available in the filesystem. It means that locale object doesn't have to represent exactly same locale which was requested.

Locale-aware resources and message formatting

Classes extending `ResourceManager` and `Resource` provide locale-aware version of resource management. They can be used to load locale-specific images, or other items.

On top of this API more specific classes exist, for example a class representing message translation in different singular and plural variants, which provides suitable translation variant depending on an integer parameter provided. A very simple example:

- Message variant 1 [one]: *We have {0} orange.*
- Message variant 2 [other]: *We have {0} oranges.*

Exact form of one message depends on parameter provided. In English, the rule is very simple (variant1 is correct when parameter equals 1, variant2 is correct otherwise), but in other languages these rules can get very complicated. In translation resource files', each message can be translated up to 6 different variants (which are tagged as defined in CLDR: zero, one, two, few, many, other, the last one being default). Plural rules defined in CLDR are used to choose correct variant.

However, translation messages are just strings, optionally with placeholders which should be substituted with real values. This can be achieved with `MessageFormat` class, which supports simple but useful substitution rules. Examples of placeholders' formats, which can be used in messages:

- You have {0} items in cart, total value: {1}
- You have {0,number} items in cart, total value: {1,number}
- You have {0,number,integer} items in cart, total value: {1,number,currency}

Basically each placeholder has a format type (number, date, time), and format style (dependent on type). Some assumptions are being made when not all of these elements are provided (for example, for number type, default format is integer and when even type is not defined, value is not being formatted at all). Custom format types and styles can be developed, using interface provided.

Formatters make extensive use of CLDR data. Messages are stored in XLIFF file format. ViewHelpers for Fluid are available, so message formatting can be done directly in template files.

Input parsers and validators

Generally parsers converts a string provided by user, to uniform inner representation depending on value type (number, date, currency) and Locale object (singleton or provided). Some additional problems need to be taken into account: for example, what timezone is used for inner date representation (and how much it differs from user's timezone), what currency is used in business logic, etc.

Validators check whether format of value provided by user conforms constrains. Validators are meant to be used with FLOW3 validation subsystem. These validators need to be Unicode-aware (for example equivalent of StringValidator), or even locale-aware (e.g. NumberValidator).

Parsers and validators also use CLDR data extensively.

Other i18n / l10n topics

During my research about internationalization and localization topics I came across few more ideas / problems, which are less important than ones described above, but could be useful in some cases. I mention them for the record, as implementing ideas described earlier would most likely take the whole available time.

- Script for initial internationalization of Fluid templates (extracting messages to a resource file, substituting plain messages with i18n tag in templates, etc)
- Number spell-out - converting integer number to translated text (depending on locale)
- Transcription to English - converting Unicode text to ASCII characters
- On-line translation system - Pootle¹ web portal for coordination of FLOW3 translation process

¹ Pootle homepage: <http://translate.sourceforge.net/wiki/pootle/index>

Deliverables

- FLOW3 support for locale-aware resources
- Auto locale detection (matching best locale for particular visitor)
- Extendable locale-aware message formatting subsystem
- Extendable locale-aware input parsing subsystem
- Documentation for implemented functionality (i18n / l10n chapter in the manual)

Timeline

Prior May 24, 2010

- Define how exactly developers should use i18n / l10n subsystem
- Identify problems that may occur during development

May 24 - 30, 2010

- Implement Locale class

May 31 - June 6, 2010

- Implement automatic locale detection

June 7 - 13, 2010

- Develop locale-aware resources management subsystem

June 14 - 20, 2010

- Implement CLDR data reader - plural forms

June 21 - 27, 2010

- Implement CLDR data reader - numbers, currencies, date and time

June 28 - July 4, 2010

- Develop message resources support - XLIFF, plural forms

July 5 - 11, 2010

- Implement Message Formatter subsystem - general (base) implementation

July 12-18, 2010

- Implement Message Formatter subsystem - number, date, time formatters

July 19-25, 2010

- Create Fluid ViewHelpers for message formatting

July 26 - August 1, 2010

- Develop Input Parser subsystem - general (base) implementation

August 2 - 8, 2010

- Develop Input Parser subsystem - number, date, time parsers

August 9 - 15, 2010

- Implement validators for FLOW3 Validation subsystem

August 16 - 20, 2010

- Write the i18n / l10n documentation chapter for the FLOW3 Manual

About me

I am a 21 years old Computer Science student and software engineering passionate, currently living in Poland and attending the Białystok University of Technology, on the third year of my degree.

I know many programming languages more or less, having most experience in PHP and Java. I use PHP for few years, at the beginning learning it for fun - developing websites involved me in. For a year from now, I was developing various web applications using customized version of KohanaPHP framework. I read a lot, and being interested in web applications I got to know (also, more or less) many different PHP related frameworks and tools, like CodeIgniter, Yii, Zend Framework, Doctrine, PHPUnit, to name a few. Also I have some knowledge about Linux systems, especially when it comes to LAMP, networking, or administration.

When time allows I like to learn new software tools and solutions, developed for various languages. For example, I have some knowledge about JSF, Django, or Rails. Spring, Grails, and Lift being frameworks I plan to learn next.

This is the first year I try to participate in the Google Summer of Code, so I have no experience with it but I was preparing myself for a quite long time. From all the projects available I chose FLOW3, because it brings techniques and software patterns not seen in PHP world before. I also read all the documentation available, and made first steps to know and understand the source code. I'm impressed by FLOW3 code quality and I'm sure that by participating in this project I would learn a lot in the field of software engineering.

I think I am good candidate for FLOW3 project because I am hard working, fast learning and very interested in future shape of the framework. I'm the type of person who always tries to plan and organize a work ahead and gets deadlines seriously. I hope I would be allowed to make positive difference for FLOW3 framework. Also, I would be interested in further development of i18n / l10n subsystem after Summer of Code, in my spare time, as this topic is extensive and a lot more can be done.

Please visit my homepage² for additional information about me, contact details, etc.

² My homepage address: <http://gusak.eu>