Translateable Exceptions for ZF - Thomas Weidner

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Translateable Exceptions for ZF Component Proposal

Proposed Component Name	Translateable Exceptions for ZF
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Translateable Exceptions for ZF
Proposers	Thomas Weidner
Zend Liaison	Wil Sinclair
Revision	1.0 - 17 January 2009: Initial Draft 1.1 - 17 January 2009: getTranslatedMessage() -> getLocalizedMessage() (wiki revision: 10)

Table of Contents

1. Overview

Translateable Exceptions is not a component but a feature addition to the existing generic Zend_Exception class.
It allows to add behaviour to translate exceptions from ZF into any other language.

2. References

None

3. Component Requirements, Constraints, and Acceptance Criteria

- This feature **will** extend Zend_Exception
- This feature **will not** add translations themself for the components
- This feature **will not** need users to change anything on their existing code

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Locale
- Zend_Translate

5. Theory of Operation

This feature adds the possibility to translate any thrown exception into a previous given language.
Therefor some changes are needed which can be done component based. Also not all changes are needed by all components.

- First:
  Providing translations for component-exceptions on a generic directory.
  We will therefor use the existing translation teams.
  The directory where the translations from ZF are stored will be:
  library/Zend/Exception/Translation/en
  For each available language a new directory (locale) can be added.
  Each component adds it's own translation file. So we have a Zend_Currency.xxx
  As format I would propose to use array. It's the fastest adapter as it only ready and does no parsing.

- Second:
  When variables are used in the exception message the exception class has give these values explicit as parameter at contruction.
  Otherwise we would not be able to translate and provide the values.

- Third:
  Add a static property to be able to activate translated exceptions.

As the original proposal for changing the exception behaviour was not accepted we have no exception code nor do we have translation constants to know which exception was thrown.
Therefor we need to provide the complete message as key value which adds additionally performance overhead. It also adds the problem that one single false character leads to a non-translated exception message.
Or when the original exception was changed it could also no longer be translated.

- Forth:
  It will detect application wide locale. But translation will only be done when the static property is set.

- Sixth:
  I additionally propose that we also add exception codes for translated exceptions. This would:
  - simplify translation
  - add codes for those who need, even if most do not. Actually many people ask for it and seem to use this feature
  - allow to add help based on exception codes in wiki when requested
  - also allow to provide a list of exceptions with their code in the manual as this is the same work for the translators when they translate the exceptions themself

Background of this proposal
Exceptions are, normally, not displayed to users.
But this proposal allows two great features to be used which are useful for non-native english speakers.

Exception messages are normally logged. With this proposal you are able also to provide a localized log file even when exceptions are added to the log.

Sometimes it is necessary to rewrite the returned exception message.
When, for example, the exception message would say "database table xxx not found"... for security reasons you would like to rewrite the message to "Db problems, please ask your database admin".

With this proposal those two features are possible.

6. Milestones / Tasks

- Milestone 1: [done] Proposal written
- Milestone 2: Proposal accepted
- Milestone 3: Prototype
- Milestone 4: Unittests
- Milestone 5: Documentation
- Milestone 6: Add ability for each available component
- Milestone 7: Translate messages

7. Class Index

- Zend_Exception

8. Use Cases

UC-01

Default usage for translation, detects application locale

```
Zend_Exception::localizeMessages(true);

try {
    ... do something
} catch (Zend_Component_Exception $e) {
    print $e->getMessage();
}
```

UC-02

Manual usage of translating exceptions

```
try {
    ... do something
} catch (Zend_Component_Exception $e) {
    print $e->getLocalizedMessage();
}
```

UC-03

Manual usage of translating exceptions by giving a fixed locale

```
try {
    ... do something
} catch (Zend_Component_Exception $e) {
    print $e->getLocalizedMessage('de');
}
```

9. Class Skeletons

```
class Zend_Exception {

    /**
     * Internal storage for values
     */
    protected $_values = null;

    /**
     * Sets a flag that all exceptions are translated
     * @param boolean $flag
     */
    public static function localizeMessages($flag = true);

    /**
     * Sets a locale to be used for exception messages
     */
    public static function setLocale($locale);

    /**
     * Returns the actual set locale
     */
    public static function getLocale();

    /**
     * Translates messages manually
     * @param string|Zend_Locale $locale
     * @return string
     */
    public function getLocalizedMessage($locale = null);

    /**
     * Returns any set value for this exception
     */
    public function getValue();

    /**
     * Base constructor has to be rewritten to allow the following syntax
     * Zend_Component_Exception('message', $code, $value1, $value2);
     * or
     * Zend_Component_Exception('message', $code, array($value1, $value2));
     * or when no value is needed
     * Zend_Component_Exception('message', $code);
     */
    public function __construct($message, $code = null)
}
```

No change in the Exception class of the component

```
class Zend_Component_Exception extends Zend_Exception {
}
```

A thrown exception

```
new Zend_Component_Exception("My exception with value %1/$s and %2/$s", 4033,
array($value1, $value2));
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>