Exception Usage

- Examples
- Internal Exception Structure
- Exception usage
- Exception Hierarchy and way to the frontend

In translate5 to less different exception types were used till now. Mostly just a ZfExtended_Exception was used.

To be more flexible in filtering on logging exceptions but also on internal exception handling with try catch more exceptions are needed.

Examples

editor_Models_Import_FileParser_Sdlxliff_Exception extends editor_Models_Import_FileParser_Exception extends ZfExtended_ErrorCodeException

In the Sdlxliff fileparser editor_Models_Import_FileParser_Sdlxliff_Exception are thrown. In general fileparser code editor_Models_Import_FileParser_Exception should be used.

Since in FileParsing many errors can happen, a fine granulated exception structure is needed. In other code places this is not the case. At least one own Exception per Plugin / Package.

Internal Exception Structure

Each exception contains the mapping between the used event code and the related error message string. Since we are in an exception we can talk here about errors and not generally about events.

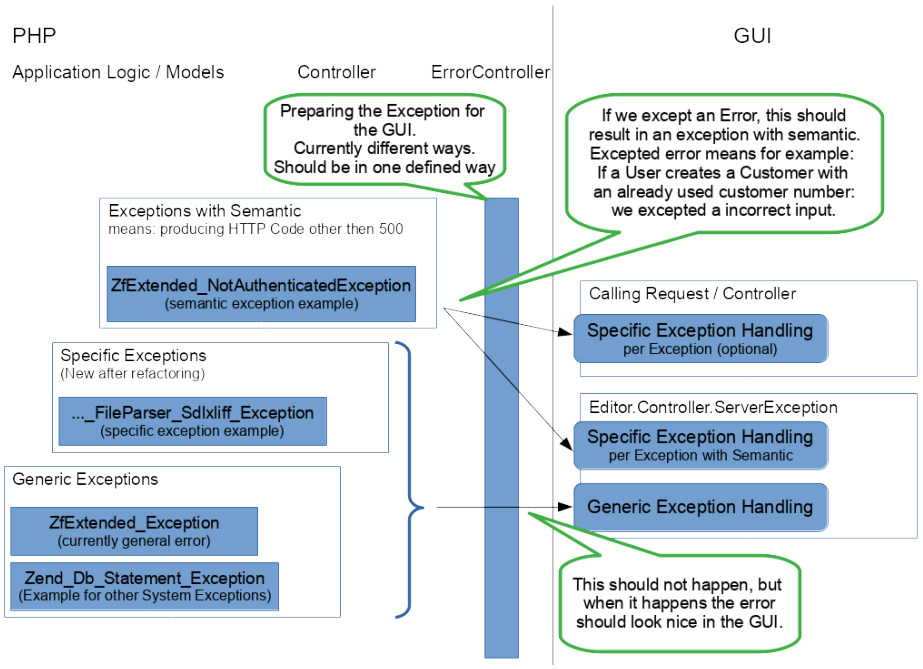
```
class editor_Models_Import_FileParser_Sdlxliff_Exception extends editor_Models_Import_FileParser_Exception {
    /**
     * @var string
     */
    protected $domain = 'editor.import.fileparser.sdlxliff';  //the domain (origin) string for hierarchical
filtering

    static protected $localErrorCodes = [              //the error (event) codes used by that exception
        'E1000' => 'The file "{filename}" contains SDL comments which are currently not supported!',
        'E1001' => 'The opening tag "{tagName}" contains the tagId "{tagId}" which is no valid SDLXLIFF!',
        'E1003' => 'There are change Markers in the sdlxliff-file "{filename}"! Please clear them first and
then try to check in the file again.',
                // [...] more mappings
    ];
}
```

Exception usage

```
if ($shitHappened && $itWasMyFault) {
        //There are change Markers in the sdlxliff-file which are not supported!         there should be a
brief comment to explain what is going wrong
    throw new editor_Models_Import_FileParser_Sdlxliff_Exception('E1003', [                //  The exception
receives just the EventCode and an array with extra data
        'task' => $this->task,
        'filename' => $this->_fileName,
    ]);
}
```

Exception Hierarchy and way to the frontend

In general	Language Resources



Basically the same:

Level	Exception Type
Semantic Level, only in the Wrapped Connector!	Only BadGateway, since this transports additional info into the GUI
Connector Implementation HttpApi Implementation	Specific Exceptions
Resources	LanguageResource Resources must not throw exceptions! If they use the systems API to get for example languages, no exception should be thrown, but a empty list should be returned.
Used Code	Generic Exceptions (DB / HTTP Connection)

About logging:

A BadGateway is not logged! So all caught exceptions transport info to the GUI via the BadGateway exception.

If the underlying error should be logged, this must be done at the place where the error happens.
This gives more control which error should be logged at all (and how often).

? Unknown Attachment

Generic Exceptions

For the handling / catching of such exceptions see the section "Examples for specific Exceptions" below.

Exception	HTTP Code	Description / Idea behind
Exception		PHP Basic Exception class should never be used directly. Can be thrown by underlying legacy code.
Zend_Exception		Zend Basic Exception class should never be used directly. This Exception an subclasses can be thrown by underlying legacy Zend code.
ZfExtended_Exception		ZfExtended Basic Exception class should not be used directly anymore. Should be replaced by specific exceptions.

ZfExtended_ErrorCodeException		Base class for specific exceptions, should not be thrown directly.

Exceptions with Semantic

This exceptions are used to answer errors to the API in REST full way, by transporting semantic as HTTP response code.

This Exceptions are intended not to be caught, since they transport information to the Caller.
But of course they can be caught if needed and handled differently.

Exception	HTTP Code	Description / Idea behind
ZfExtended_BadMethodCallException	405 Method Not Allowed	Means that the used HTTP Method is not allowed / not implemented. So a usage makes only sense on the controller level.
ZfExtended_NotAuthenticatedException	401 Unauthorized	Means the the user is not authorized, so in the request there was no information to identify the user. This error should redirect the user in the GUI to the login page.
ZfExtended_NotFoundException	404 Not Found	Is used in application routing, is thrown when the whole requested route is not found (invalid URL).
ZfExtended_Models_Entity_NotFoundException	404 Not Found	Is used if an entity can not be found (if it is loaded via an id or guid)
ZfExtended_NoAccessException ZfExtended_Models_Entity_NoAccessException	403 Forbidden	The user (authenticated or not) is not allowed to see / manipulate the requested resource. This can be either by missing ACLs, or missing other preconditions disallowing the request. In difference to 422 and 409: the authenticated user is the reason why the request can not be processed. TODO: clear the difference between both exceptions.
TODO new Unprocesseable Entity Exception (or reuse ValidateException?) ZfExtended_UnprocessableEntityException::createResponse	422 Unprocessable Entity	Should be used PUT/POSTed content does prevent normal processing: Wrong parameters, missing parameters. In other words: the requested call and its data is reasonable that the request can not be processed.
ZfExtended_Models_Entity_Conflict	409 Conflict	Should be used if the status of the entity does prevent normal processing: The entity is locked, the entity is used/referenced in other places. In other words: the entity it self is reasonable that the request can not be processed.
ZfExtended_VersionConflictException	409 Conflict	Must only be used if entity can not be saved due a changed entity version. The classical usage of the 409 HTTP error: the entity was changed in the meantime.
ZfExtended_BadGateway TODO implement 504 Gateway Timeout	502 Bad Gateway	Should be used if our request calls internally a third party service, and the third party service or the communication with it does not work.
ZfExtended_ValidateException	422 Unprocessable Entity	Use this exception if the given data in the request can not be validated or contains non processable data.
ZfExtended_FileUploadException	400 Bad Request	TODO, currently not used. Should be used in the context of errors with uploaded data
ZfExtended_Models_MaintenanceException	503 Service Unavailable	Only usable in the context of the maintenance mode
ZfExtended_Models_Entity_Exceptions_IntegrityDuplicateKey	TODO specific exception? No direct HTTP code	Is thrown on saving entities and the underlying DB call returns a "integrity constraint violation 1062 Duplicate entry" error: That means an entity with such key does already exist (entities with additional unique keys, customer number or user login).
ZfExtended_Models_Entity_Exceptions_IntegrityConstraint	TODO specific exception? No direct HTTP code	Is thrown on saving/deleting entities and the underlying DB call returns one of the following "integrity constraint violation" error: • 1451 Cannot delete or update a parent row: a foreign key constraint fails the to be deleted entity is used via foreign key, and may not deleted therefore. • 1452 Cannot add or update a child row: a foreign key constraint fails the to be saved foreign entity does not exist (anymore)

Creation of exceptions static method createResponse

One fundamental problem for development in translate5 is that the default GUI language of translate5 is german, but most of the error messages are in english.
At some places the error message is intended to be used directly as user feedback, therefore the error message must be also in german at that place and the message must go through the internal translation mechanism before send to the user.

That is indeed not the case for errors which should unlikely not happen, there a default english message is enough.

Therefore on exception creation we have to decide if:

1. the error message is shown to the user in a usual dialog (for example if a new user is created where the loginname is already in use) and therefore the user can change something to retry the request
2. the error can not be changed by the user (or it is a very seldom situation), there the message is left in english.

This results in two different ways how semantic exceptions are produced:

- the default constructor, all data must be given as usual.
- the static method getResponse which creates a new exception instance prefilled with the given error messages in the users GUI language

Examples for specific Exceptions

This exceptions are used in the case of errors in the application.

Handling of such exceptions:

1. They are caught and the application deals with the error.
   No impact to the API result.
2. They are caught and the exceptions is translated into an exception with semantic
   This happens mostly on the Controller level. The main error message remains untranslated.
   In an additional Container multiple errors, with extra data and translated messages can be stored into the semantic exception.
3. Nothing is done and the exception bubbles to the caller:
   In direct API usage this should happen only if we really don't except that such exceptions happens.
   In worker context it is Ok to bubble such exceptions, since they are called via REST,
   but are then mostly running asynchronous without answering in a RESTfull way to the caller.
   This exceptions are transported untranslated into the GUI.

Exception	Description / Idea behind
editor_Models_Import_FileParser_Sdlxliff_Exception	Exception which is thrown on the usage of the SdlXliff fileparser.
ZfExtended_Logger_Exception	Exception which is thrown in the context of error logging, so if in the logging is happening an error
ZfExtended_ErrorCodeException	Base class for all ErrorCode based Exceptions