Getting started with Gemstracker Events

Introduction

Events triggered in Gemstracker There are a number of predefined event scripts that can already be selected when configuring your tracks, rounds and surveys. If these do not suffice, specific event scripts will have to be made in PHP and added to your project. This document will give a good head start to create these event scripts.

Type of Events

7 type of events: * Survey before answering * Survey completed * Survey Display * Round changed * Track Calculation * Track completed * Track Field update

Survey before answering:

To add something to the specific survey before answering it. For instance to insert the answers of a previous round of the survey.

Survey Completed:

To change something after the survey is complete. For instance score calculations.

Survey Display:

To change the way the answers of a survey are shown. For instance to show only specific fields like score fields.

Round changed event:

To change something after a specific round is done. Similar to Survey completed but then specific to that round.

Track Field update:

To change something in the track when a track field updates. E.G. a treatment has been selected, and needs to activate or deactive certain rounds.

The different events will also be triggered when recalculating your tracks.

Location

To make Gemstracker see your event script you need to place it In your Project application folder in the respective event code folder.

Your event script can now be selected in the track, round or survey edit screen.

General

Inheritance

Each event script has to implement the interface of their respective event type. These interfaces are the blueprint of your event script, naming the methods needed for that specific event. * Gems_Event_SurveyBeforeAnsweringEventInterface * Gems_Event_SurveyCompletedEventInterface * Gems_Event_SurveyDisplayEventInterface & Gems_Tracker_Snippets_AnswerNameFilterInterface * Gems_Event_RoundChangedEventInterface * Gems_Event_TrackCalculationEventInterface * Gems_Event_TrackCompletedEventInterface * Gems_Event_TrackFieldUpdateEventInterface

To get access to registry objects like Zend_Translate, Gems_loader or Zend_Db_Adapter_Abstract your script can also extend MUtil_Registry_TargetAbstract or a class that extends this.

Example:

```
class NewProject_Event_Survey_BeforeAnswering_MyBeforeAnsweringEvent extends
\MUtil_Registry_TargetAbstract
    implements \Gems_Event_SurveyBeforeAnsweringEventInterface
{
}
```

Event name

Each event script needs the method getEventName which returns the name of the event as displayed in the track screen.

e.g.

```
public function getEventName()
{
    return 'My event name';
}
```

Optionally you can add the class property translate to your event code if you want your event name to translate correctly across the different languages in use in your gemstracker project.

```
protected translate;
```

```
public function getEventName()
{
    return $this->translate->_('My event name');
}
```

Common used methods

token getRawAnswers

As most events will process tokens and their values, the $token→getRawAnswers(); is used a lot in scripts. It will return an array containing the surveys values with their Limesurvey question code as Key

```
array(
    'weight' => 75,
    'height' => 180
);
```

Survey events

Survey Before answering

Interface

A survey before answering event script implements Gems_Event_SurveyBeforeAnsweringEventInterface

ProcessTokenInsertion

Survey before answering events have the processTokenInsertion method. This method receives the current Token as parameter and needs to return an array with all the values that have changed.

For instance, if you'd like to set the default length in an BMI calculation survey, you'd do:

```
public function processTokenInsertion(\Gems_Tracker_Token $token)
{
    $answers = array();
    $answers['length'] = 180;

    return $answers;
}
```

The often used 'get previous answers' event script shows a good example of how to look at previous tokens in this track, and pre-fill the answers.

```
public function processTokenInsertion(\Gems_Tracker_Token $token)
{
    if ($token->hasSuccesCode() && (! $token->isCompleted())) {
        $surveyId   = $token->getSurveyId();

        $prev = $token;
        while ($prev = $prev->getPreviousToken()) {

            if ($prev->hasSuccesCode() && $prev->isCompleted()) {
                if ($prev->getSurveyId() == $surveyId) {
                    return $prev->getRawAnswers();
                }
            }
        }
    }
}
```

```
if ($token->hasSuccesCode() && (!$token->isCompleted()) {
```

Checks if the token can be filled in (has an OK as reception Code) and if it hasn't been filled in already.

```
$surveyId   = $token->getSurveyId();
```

gets the survey ID of the token, so we can compare it with the previous tokens, to make sure we have the same survey but in an earlier round.

```
$prev = $token;
while ($prev = $prev->getPreviousToken()) {
}
```

Loops through all the previous tokens in this track, until the right match has been found and the loop is broken.

```
if ($prev->hasSuccesCode() && $prev->isCompleted()) {
}
```

Checks if the previous survey has an OK reception code and if this one IS finished.

```
if ($prev->getSurveyId() == $surveyId) {
    return $prev->getRawAnswers();
}
```

Checks if the surveyID is the same on both tokens, if that's the case, it returns all the raw answers to be filled into the current Survey.

This code can for instance be adjusted to only take the Length answer of a previous BMI survey, as the length might not change during a track, or maybe even take answers from a completely different survey.

Survey completed

Interface

A survey before answering event script implements Gems_Event_SurveyCompletedEventInterface

ProcessTokenData

Survey completed events have the processTokenData method. This method receives the current Token as parameter and needs to return an array with all the values that have changed.

Here you usually get the raw token answers and transform, or supplement them. For instance the BMI calculation

```php
public function processTokenData(\Gems_Tracker_Token $token)
{
    $tokenAnswers = $token->getRawAnswers();

    if (isset($tokenAnswers['LENGTH'], $tokenAnswers['WEIGHT']) &&
$tokenAnswers['LENGTH'] && $tokenAnswers['WEIGHT']) {
        $length = $tokenAnswers['LENGTH'] / 100;
        $newValue = round($tokenAnswers['WEIGHT'] / ($length * $length),
2);

        if ($newValue !== $tokenAnswers['BMI']) {
            return array('BMI' => $newValue);
        }
    }

    return false;
}
```

Survey Display

Interface

A survey display event script implements Gems_Event_SurveyDisplayEventInterface

If you need to filter specific answers as well, you can also implement Gems_Tracker_Snippets_AnswerNameFilterInterface For extra filter functions you can also extend Gems_Event_SurveyAnswerFilterAbstract

Gems_Event_SurveyDisplayEventInterface -> getAnswerDisplaySnippets

Survey display events have the getAnswerDisplaySnippets method. In this method you return the name of the snippet that shows the answers.

```
public function getAnswerDisplaySnippets(\Gems_Tracker_Token $token)
{
    return 'Tracker_Answers_TrackAllAnswersModelSnippet';
}
```

Will refer to a snippet found in application\classes\<yourProject>\Snippets\Tracker\Answers

an answer snippet can extend Gems_Tracker_Snippets_AnswerModelSnippetGeneric to display the answers.

An example snippet that also shows all answers without a label is:

```
class Gems_Snippets_Tracker_Answers_TrackAllAnswersModelSnippet extends
Gems_Tracker_Snippets_AnswerModelSnippetGeneric
{
    /**
     * Use compact view and show all tokens of the same surveyId in
     * one view. Property used by respondent export
     *
     * @var boolean
     */
    public $grouped = true;

    /**
     * Creates the model
     *
     * @return MUtil_Model_ModelAbstract
     */
    protected function createModel()
    {
        $model = parent::createModel();

        foreach ($model->getItemNames() as $name) {
            if (! $model->has($name, 'label')) {
                $model->set($name, 'label', ' ');
            }
        }

        return $model;
    }

    /**
     * Overrule to implement snippet specific filtering and sorting.
     *
     * @param MUtil_Model_ModelAbstract $model
     */
    protected function processFilterAndSort(MUtil_Model_ModelAbstract
$model)
    {
        if ($this->request) {
            $this->processSortOnly($model);

            if ($this->grouped) {
                $filter['gto_id_respondent_track'] =
$this->token->getRespondentTrackId();
                $filter['gto_id_survey']           =
$this->token->getSurveyId();
            } else {
                $filter['gto_id_token']            =
$this->token->getTokenId();
            }

            $model->setFilter($filter);
        }
    }
}
```

Gems_Tracker_Snippets_AnswerNameFilterInterface -> filterAnswers

To filter which answers get shown in the answer screen, you can filter the answers with the filterAnswers method. As attributes it gets the table bridge, the model and an array of the current answer names that need to be displayed. It should return an array with the answer names it should display.

```
public function filterAnswers(MUtil_Model_Bridge_TableBridge $bridge,
MUtil_Model_ModelAbstract $model, array $currentNames)
{
// If the token is not finished, it will not display the answers anyway, so
no need to go through the answers
if (! $this->token->isCompleted()) {
return $currentNames;
}

$hideQuestions = array(
            'weight'
        );

        $filteredNames = $currentNames;

        foreach($hideQuestions as $hideQuestion) {
            if ($key = array_search($hideQuestion, $filteredNames)) {
                unset($filteredNames[$key]);
            }
        }
        return $this->restoreHeaderPositions($model, $filteredNames);
    }
```

The restoreHeaderPositions is a helper function in Gems_Event_SurveyAnswerFilterAbstract.