Tutorial: Implement your own iTop AI extension

Minimal iTop version:	3.2.0
Code:	itomig-ai-explain-oql
Dependencies:	itomig-ai-base
Language:	en

Change language: de

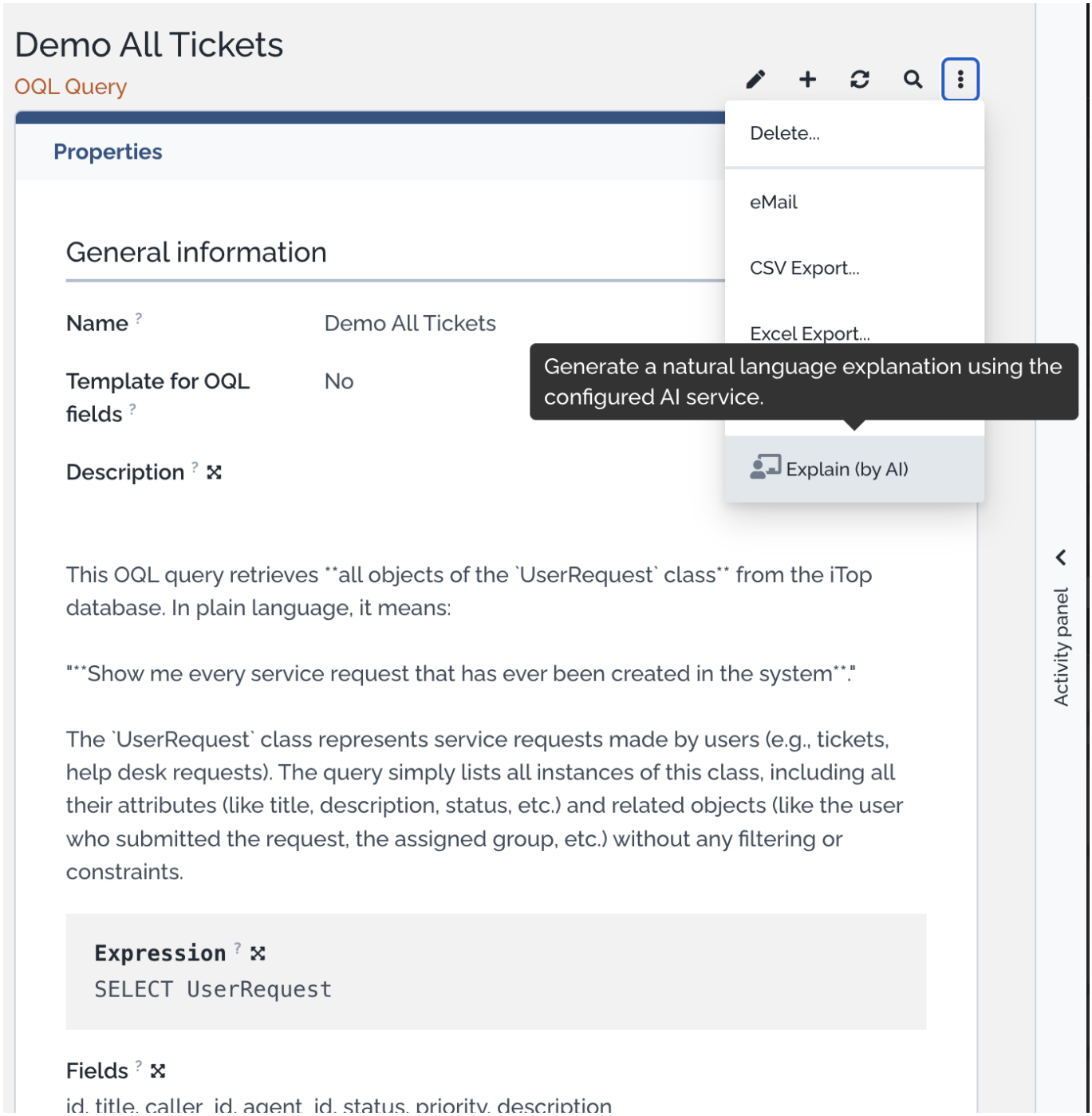
Introduction

This tutorial shows step by step how to develop a lean iTop extension based on the ITOMIG AI-Base to develop a lean iTop extension that has stored OQL queries automatically explained by an AI. The AI-Base provides all the basic functions (configuration, API client, prompt management) so that you can concentrate on the technical added value.

Aim of the tutorial

After completing this tutorial, the detailed view of a phrasebook query (`QueryOQL`) contains a new action **„Explain (by AI)„** is available. Clicking on it sends the OQL statement to the AI server. This returns an explanation in natural language, which is then written to a configurable attribute (default: `description`).

ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein



Requirements

- Basic knowledge of iTop extension development (see </ignore>iTop Hub - Make your own Extension).
- Installed and configured itomig-ai-base (including a working connection to your AI backend).
- Write access to the desired extension directory (`extensions/`) and the ability to recompile iTop.

ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

To configure the **ITOMIG-AI Base** requires at least the configuration of the URL, API key and language model.

```
    'itomig-ai-base' => array (
        'ai_engine.configuration' => array (
          'api_key' => 'your-key',
          'url' => 'http://127.0.0.1/api/v1',
          'model' => 'qwen3:14b',
        ),
```

You can use a public AI API (e.g. Mistral) and store your corresponding API key (Anleitung Mistral: how do I get an API key) or you can set up your own LLM (e.g. Ollama) Anleitung Ollama: Set up your own LLM.

Implementation

Create the new AI extension "itomig-ai-explain-oql"

1. **Create new module framework** Create the folder `itomig-ai-explain-oql/` folder and create the standard files:

- `module.itomig-ai-explain-oql.php` - defines metadata, the dependency on the `itomig-ai-base/x.y.z` (e.g. 25.3.1) and default settings for prompt and target attribute.
- `datamodel.itomig-ai-explain-oql.xml` - can remain empty for this example, as no new classes are required.
- `composer.json` - registers the namespace prefix `<yournamespace>\\ItomigAiExplainOql\\` for the `src/`-folder and later allows a `composer dump-autoload -o`.

module.itomig-ai-explain-oql.php

```
<?php
// module.itomig-ai-explain-oql.php
// iTop module definition file
//
SetupWebPage::AddModule(
    __FILE__, // Path to the current file, all other file names are relative
to the directory containing this file
    'itomig-ai-explain-oql/1.0.0',
    array(
        // Identification
        'label' => 'OQL Explaining by AI (ITOMIG GmbH)',
```


ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

```
        'category' => 'business',

        // Setup
        'dependencies' => array(
            'itomig-ai-base/25.2.1',
        ),
        'mandatory' => false,
        'visible' => true,
        // Components
        'datamodel' => array(
            'model.itomig-ai-explain-oql.php',
            'vendor/autoload.php',
            'src/Hook/ExplainOqlPopupMenuExtension.php',
        ),
        'webservice' => array(
        ),
        'data.struct' => array(
        ),
        'data.sample' => array(
        ),

        // Documentation
        'doc.manual_setup' => '', // hyperlink to manual setup
documentation, if any
        'doc.more_information' => '', // hyperlink to more information, if
any

        // Default settings
            'settings' => array(
                'system_prompt' => <<<PROMPT
You are an experienced iTop consultant. Explain OQL queries to end users in
concise, clear language. Highlight the queried classes, key filters, joins,
and returned attributes. Mention parameter placeholders if present, but do
not output code.
PROMPT,
            ),
        )
    );
```

2. **Implement helper for AI communication** Place `src/Helper/AIOQLHelper.php` . The method `explainOQLQuery()` method takes the OQL text, target object and target field, loads module settings (`system_prompt`), calls `AIService::PerformSystemInstruction()` and writes the response back to the object.

An instance of AIService (from the itomig-ai-base) is therefore used here to use the basic methods for communicating with an AI server. The prompt that is sent to the AI is prepared accordingly and sent

ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

via the method `PerformSystemInstruction()` method to the AIService. The response can then be persisted if required (as in this example as the attribute `description` attribute on the QueryOQL object).

src/Helper/AIOQLHelper.php

```php
<?php
/* src/Helper/AIOQLHelper.php
 * Copyright (C) 2025 ITOMIG GmbH
 */

namespace Itomig\ItomigAiExplainOql\Helper;

use Itomig\iTop\Extension\AIBase\Exception\AIResponseException;
use Itomig\iTop\Extension\AIBase\Service\AIService;
use MetaModel;

class AIOQLHelper
{
    public const MODULE_CODE = 'itomig-ai-explain-oql';
    private const INSTRUCTION_NAME = 'explain_oql';
    private const DEFAULT_PROMPT_PREFIX = 'Explain the following iTop OQL query in plain language.';

    /**
     * Generates an explanation for the given OQL statement via the configured AI service.
     *
     * @param string $sOql
     * @param \cmdbAbstractObject $oTargetObject Object that will receive the explanation
     * @param string $sTargetAttCode Attribute code on the target object that will be filled with the explanation
     *
     * @throws AIResponseException
     */
    public static function explainOQLQuery(string $sOql, \cmdbAbstractObject $oTargetObject, string $sTargetAttCode) : void
    {
        $oAIService = new AIService();

        // Register custom system prompt if configured for this module.
        $sSystemPrompt = MetaModel::GetModuleSetting(self::MODULE_CODE, 'system_prompt', '');
        if (is_string($sSystemPrompt) && trim($sSystemPrompt) !== '') {
```


ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

```
            $oAIService->addSystemInstruction(self::INSTRUCTION_NAME,
$sSystemPrompt);
        }

        // Determine the prefix that will precede every prompt sent to the
AI backend.
        $sPromptPrefix = MetaModel::GetModuleSetting(self::MODULE_CODE,
'prompt_prefix', self::DEFAULT_PROMPT_PREFIX);

        // Compose the final prompt: prefix first, then the raw OQL query as
required.
        $sPrompt = $sPromptPrefix . PHP_EOL . PHP_EOL . 'OQL:' . PHP_EOL .
$sOql;

        $sExplanation = $oAIService->PerformSystemInstruction($sPrompt,
self::INSTRUCTION_NAME);
        $oTargetObject->Set($sTargetAttCode, $sExplanation);
        $oTargetObject->DBWrite();
    }
}
```

The following steps describe a possible implementation for triggering the logic implemented above. In addition to the AJAX solution presented here, other implementations for a possible user interface are also conceivable. The implementation described here is quite complex and specially adapted for the QueryOQL class. Depending on the use case, other solutions may be preferable.

3. **Register pop-up menu hook** To make the function available in the iTop interface, define a suitable `iPopupMenuExtension`. This interface can be used to extend the context menu of a class. Here we include an AJAX call, which in turn calls the PHP method of our helper class `AIOQLHelper` helper class. Admittedly, this method is somewhat complex, as the iTop class `QueryOQL` is not defined as an XML class; for other use cases, this can be much easier (e.g. using the combodo-customer-hyperlinks extension, see </ignore>iTophub-Dokumentation).

Hooks must be explicitly integrated. Store `src/Hook/ExplainOqlPopupMenuExtension.php` and add the path in `module.itomig-ai-explain-oql.php` to the `datamodel`-list. The class implements `iPopupMenuExtension::EnumItems()` and returns for `QueryOQL`-objects an AJAX menu item including loading animation and error message dialogue.

ExplainOqlPopupMenuExtension

```
<?php
/*
```


ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

```
 * Copyright (C) 2025 ITOMIG GmbH
 */

namespace Itomig\ItomigAiExplainOql\Hook;

use Dict;
use iPopupMenuExtension;
use JSPopupMenuItem;
use QueryOQL;
use UserRights;
use utils;

class ExplainOqlPopupMenuExtension implements iPopupMenuExtension
{
    /**
     * @inheritDoc
     */
    public static function EnumItems($iMenuId, $param)
    {
        if ($iMenuId !== iPopupMenuExtension::MENU_OBJDETAILS_ACTIONS) {
            return array();
        }
        if (!($param instanceof QueryOQL)) {
            return array();
        }
        $oQuery = $param;
        if (!UserRights::IsActionAllowed(get_class($oQuery),
UR_ACTION_MODIFY, $oQuery)) {
            return array();
        }
        $sLabel = Dict::S('Class:QueryOQL:Action:ExplainByAI');
        $sTooltip = Dict::S('Class:QueryOQL:Action:ExplainByAI+');
        $sError =
addslashes(Dict::S('Class:QueryOQL:Action:ExplainByAI:Error'));

        $sAjaxUrl = utils::GetAbsoluteUrlModulesRoot().'itomig-ai-explain-
oql/ajax.explain-oql.php';
        $aPayload = array(
            'operation' => 'explain',
            'obj_class' => get_class($oQuery),
            'obj_key' => $oQuery->GetKey(),
        );
        $sPayloadJson = json_encode($aPayload);

        $sJsCode = <<<JS
var \$item = $(this);
```


ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

```
if (\$item.hasClass('ibo-is-loading')) {
    return;
}
\$item.addClass('ibo-is-loading');
$.ajax({
    url: '{$sAjaxUrl}',
    method: 'POST',
    dataType: 'json',
    data: {$sPayloadJson}
}).done(function(oData) {
    if (oData && oData.success === true) {
        window.location.reload();
        return;
    }
    var sMessage = (oData && oData.error) ? oData.error : '{$sError}';
    alert(sMessage);
}).fail(function() {
    alert('{$sError}');
}).always(function() {
    \$item.removeClass('ibo-is-loading');
});
JS;
        $oMenuItem = new JSPopupMenuItem('itomig-ai-explain-oql', $sLabel,
$sJsCode);
        $oMenuItem->SetTooltip($sTooltip);
        $oMenuItem->SetIconClass('fas fa-chalkboard-teacher');
        return array($oMenuItem);
    }
}
```

4. **Create AJAX endpoint** The file `ajax.explain-oql.php` is located directly in the module directory. It starts iTop, checks authorisations via `LoginWebPage::DoLoginEx()` loads the target object with `MetaModel::GetObject()` and calls the helper. The result is returned as JSON (`{'success': true}`) so that the UI can reload the page. When integrating in the hook, note the possible AppRoot depth (`../../` vs. `../../../`) if you want to use the module in different directories.

```
<?php
/**
 * @copyright (C) 2024 ITOMIG
 * @license   AGPL-3.0
 */

use Itomig\ItomigAiExplainOql\Helper\AIOQLHelper;

require_once('../../approot.inc.php');
```


ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein

```
require_once(APPROOT.'application/application.inc.php');

require_once APPROOT.'application/startup.inc.php';
require_once APPROOT.'application/loginwebpage.class.inc.php';
LoginWebPage::DoLoginEx(null, false);

$sClass = utils::ReadParam('obj_class', '', false, 'class');
$iObjectId = (int) utils::ReadParam('obj_key', 0);

/** @var QueryOQL $oObject */
$oObject = MetaModel::GetObject($sClass, $iObjectId, true, true);


$sTargetAttribute = MetaModel::GetModuleSetting(AIOQLHelper::MODULE_CODE,
'target_attribute', 'description');
$oObject->AllowWrite(true);
AIOQLHelper::explainOQLQuery((string) $oObject->Get('oql'), $oObject,
$sTargetAttribute);

header('Content-Type: application/json');
echo json_encode(array('success' => true));
```

5. **Add translations** Add `en.dict.itomig-ai-explain-oql.php` and `en.dict.itomig-ai-explain-oql.php` with the menu labelling, tooltips and error messages. The files are automatically recognised by iTop as long as they correspond to the naming scheme `<sprache>.dict.<modul>.php` naming scheme.

6. **Recompile and test iTop** Run the setup wizard or activate the symlink mode so that iTop recognises the new extension. Then open an existing phrasebook query, test the new menu item and check whether the AI explanation lands in the desired field. Add screenshots of the menu item, AJAX feedback and result to your wiki at this point.

Further customisations

- Adjust the module settings (`system_prompt`, `target_attribute`) either directly in `module.itomig-ai-explain-oql.php` or via the web interface and iTop configuration to vary the target group and output field.
- Use the same pattern to support other object types - for example change requests or tickets - by extending the hook or adding new hooks.

ITOMIG GmbH • Allensteiner Str. 20 • 77694 Kehl am Rhein • Tel. +49 7031 3892506 • Email: info@itomig.de • WWW: www.itomig.de
Geschäftsführer: David Gümbel Lars Kaltefleiter • HRB 716960 Amtsgericht Freiburg • Sitz der Gesellschaft: 77694 Kehl am Rhein