Tutorial: Implementiere deine eigene iTop AI-Extension

Minimale iTop-Version:	3.2.0
Code:	itomig-ai-explain-oql
Abhängigkeiten:	itomig-ai-base
Sprache:	de

Sprache ändern: en

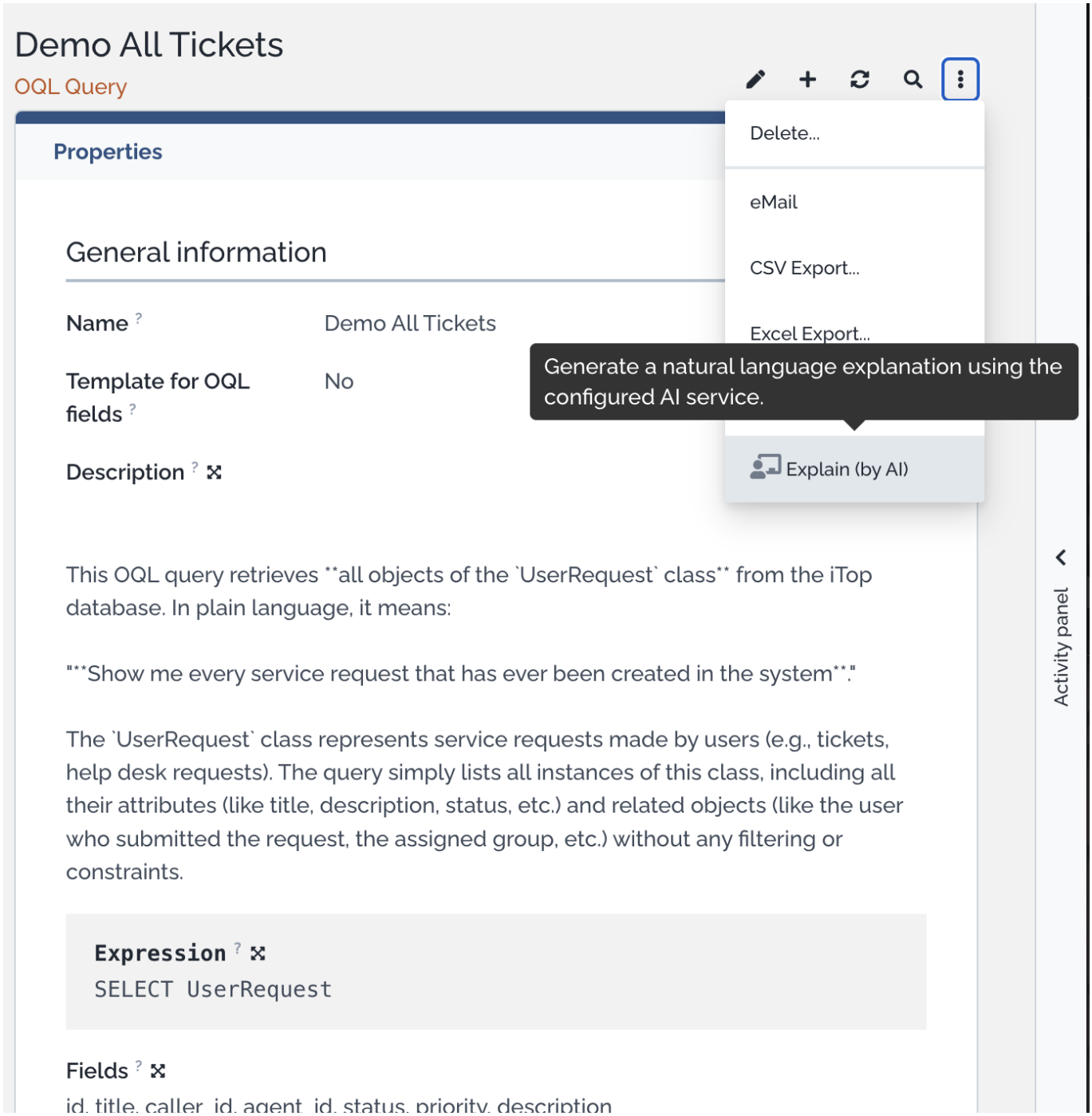
Einführung

Dieses Tutorial zeigt Schritt für Schritt, wie du auf Basis der ITOMIG AI-Base eine schlanke iTop-Erweiterung entwickelst, die gespeicherte OQL-Queries automatisch durch eine KI erklären lässt. Die AI-Base liefert dabei alle Grundfunktionen (Konfiguration, API-Client, Prompt-Verwaltung), sodass du dich auf den fachlichen Mehrwert konzentrieren kannst.

Ziel des Tutorials

Nach Abschluss dieser Anleitung steht in der Detailansicht einer Phrasebook-Abfrage (`QueryOQL`) eine neue Aktion **„Explain (by AI)“** zur Verfügung. Ein Klick darauf sendet die OQL-Anweisung an den AI-Server. Dieser liefert eine Erklärung in natürlicher Sprache zurück, die anschließend in ein konfigurierbares Attribut geschrieben wird (Standard: `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



Voraussetzungen

- Grundkenntnisse in der iTop-Extension-Entwicklung (siehe iTop Hub – Make your own Extension).
- Installierte und konfigurierte itomig-ai-base (einschließlich einer funktionierenden Verbindung zu deinem AI-Backend).
- Schreibzugriff auf das gewünschte Erweiterungsverzeichnis (`extensions/`) sowie die Möglichkeit, iTop neu zu kompilieren.

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

Zur Konfiguration der **ITOMIG-AI Base** ist mindestens die Konfiguration von URL, API-Key und Sprachmodell notwendig.

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

Dabei kann eine öffentliche KI-API genutzt werden (z. B. Mistral) und man hinterlegt seinen entsprechende API-Key (Anleitung Mistral: wie bekomme ich einen API-Key) oder man richtet sein eigenen LLM ein (z. B. Ollama) Anleitung Ollama: Setup eigenes LLM.

Implementierung

Erstellen der neuen AI-Extension "itomig-ai-explain-oql"

1. **Neues Modulgerüst anlegen** Lege den Ordner `itomig-ai-explain-oql/` an und erstelle die Standarddateien:

- `module.itomig-ai-explain-oql.php` – definiert Metadaten, die Abhängigkeit zur `itomig-ai-base/x.y.z` (z. B. 25.3.1) sowie Standard-Einstellungen für Prompt und Zielattribut.
- `datamodel.itomig-ai-explain-oql.xml` – kann für dieses Beispiel leer bleiben, da keine neuen Klassen benötigt werden.
- `composer.json` – registriert das Namespace-Präfix `<yournamespace>\\ItomigAiExplainOql\\` für den `src/`-Ordner und erlaubt später einen `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. **Helper für die AI-Kommunikation implementieren** Lege `src/Helper/AIOQLHelper.php` an. Die Methode `explainOQLQuery()` übernimmt OQL-Text, Zielobjekt und Zielfeld, lädt Modul-Einstellungen (`system_prompt`), ruft `AIService::PerformSystemInstruction()` auf und schreibt die Antwort zurück auf das Objekt.

Man nutzt hier also eine Instanz von AIService (aus der itomig-ai-base), um die grundlegenden Methoden zur Kommunikation mit einem AI-Server zu verwenden. Der Prompt, der an die AI gesendet

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

wird, wird entsprechend vorbereitet und über die Methode `PerformSystemInstruction()` an den AIService übergeben. Die Response kann anschließend bei Bedarf persistiert werden (wie in diesem Beispiel als Attribut `description` auf dem QueryOQL-Objekt).

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();
    }
}
```

Nachfolgende Schritte beschreiben eine mögliche Implementierung zum Trigger der oben umgesetzten Logik. Neben der hier vorgestellten AJAX-Lösung, sind auch andere Umsetzungen für eine mögliches User-Interface denkbar. Die hier beschriebene Umsetzung ist ziemlich komplex und speziell für die Klasse QueryOQL angepasst. Je nach Use-Case sind andere Lösungen zu präferieren.

3. **Popup-Menü-Hook registrieren** Um die Funktion in der iTop-Oberfläche verfügbar zu machen, definierst du eine passende `iPopupMenuExtension`. Über dieses Interface lässt sich das Kontextmenü einer Klasse erweitern. Hier binden wir einen AJAX-Aufruf ein, der wiederum die PHP-Methode unserer Helper-Klasse `AIOQLHelper` startet. Zugegeben, dieser Weg ist etwas komplex, da die iTop-Klasse `QueryOQL` nicht als XML-Klasse definiert ist; für andere Use-Cases kann das deutlich einfacher sein (z. B. mithilfe der Extension combodo-customer-hyperlinks, siehe iTophub-Dokumentation).

Hooks müssen explizit eingebunden werden. Hinterlege `src/Hook/ExplainOqlPopupMenuExtension.php` und füge den Pfad in `module.itomig-ai-explain-oql.php` zur `datamodel`-Liste hinzu. Die Klasse implementiert `iPopupMenuExtension::EnumItems()` und liefert für `QueryOQL`-Objekte einen AJAX-Menüpunkt inklusive Ladeanimation und Fehlermeldungsdialog.

ExplainOqlPopupMenuExtension

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

```php
<?php
/*
 * 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);
```


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

```
        $sJsCode = <<<JS
var \$item = $(this);
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. **AJAX-Endpunkt erstellen** Die Datei `ajax.explain-oql.php` liegt direkt im Modulverzeichnis. Sie startet iTop, prüft Berechtigungen via `LoginWebPage::DoLoginEx()`, lädt das Zielobjekt mit `MetaModel::GetObject()` und ruft den Helper. Das Ergebnis wird als JSON (`{'success': true}`) zurückgegeben, damit die UI die Seite neu laden kann. Beachte beim Einbinden im Hook die mögliche AppRoot-Tiefe (`../../` vs. `../../../`), falls du das Modul in verschiedenen Verzeichnissen einsetzen möchtest.

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

use Itomig\ItomigAiExplainOql\Helper\AIOQLHelper;
```


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.inc.php');
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. **Übersetzungen hinzufügen** Ergänze `en.dict.itomig-ai-explain-oql.php` und `de.dict.itomig-ai-explain-oql.php` um die Menübeschriftung, Tooltips und Fehlermeldungen. Die Dateien werden von iTop automatisch erkannt, solange sie dem Namensschema `<sprache>.dict.<modul>.php` entsprechen.

6. **iTop neu kompilieren und testen** Führe den Setup-Assistenten aus oder aktiviere den Symlink-Modus, damit iTop die neue Erweiterung erkennt. Öffne anschließend eine bestehende Phrasebook-Abfrage, teste den neuen Menüpunkt und überprüfe, ob die KI-Erklärung im gewünschten Feld landet. Ergänze in deinem Wiki an dieser Stelle Screenshots von Menüpunkt, AJAX-Feedback und Ergebnis.

Weiterführende Anpassungen

- Passe die Modul-Einstellungen (`system_prompt`, `target_attribute`) entweder direkt in `module.itomig-ai-explain-oql.php` oder über die Weboberfläche und die iTop-Konfiguration an, um Zielgruppe und Ausgabefeld zu variieren.
- Nutze das gleiche Muster, um andere Objekttypen zu unterstützen – zum Beispiel Change Requests oder Tickets – indem du den Hook erweiterst oder neue Hooks ergänzt.

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