```php
<?php
namespace Plugin\HogePlugin;

use Eccube\Plugin\AbstractPluginManager;
use Symfony\Component\DependencyInjection\ContainerInterface;
use Eccube\Entity\PageLayout;
use Eccube\Repository\LayoutRepository;
use Eccube\Repository\PageLayoutRepository;
use Eccube\Repository\PageRepository;
use Eccube\Entity\MailTemplate;
use Eccube\Repository\MailTemplateRepository;
use Plugin\HogePlugin\Entity\Config;
use Plugin\HogePlugin\Repository\ConfigRepository;

class PluginManager extends AbstractPluginManager
{
    // dtb_layout テーブルのid=2のレコードのレイアウトを使用する。それをnew PageLayouutの際のセットデータとして使用。
    const ADD_PAGE_LAYOUT_ID   = 2;

    // dtb_page newPage でレコードを追加する際の各カラム毎のセットデータとして使用
    const ADD_PAGE_NAME        = "問合せフォーム（ほげ）";
    const ADD_PAGE_URL         = "my_contact";
    const ADD_PAGE_FILE_NAME   = "HogePlugin/Resource/template/my_index";
    const ADD_PAGE_META_ROBOTS = "noindex";

    const ADD_PAGE_EDIT_TYPE   = 2;

    // dtb_mail_template  new MailTemplate  でレコードを追加する際の各カラム毎のセットデータとして使用
    const ADD_MAIL_TEMPLATE_NAME        = "ほげプラグイン用メール";
    const ADD_MAIL_TEMPLATE_FILE_NAME   = "HogePlugin/Resource/template/mail/my_contact_mail.twig";
    const ADD_MAIL_TEMPLATE_MAIL_SUBJECT = "お問い合わせの受付が完了しました。";

    // plg_question_plugin_config レコードを追加する際のセットデータとして使用
    const ADD_CONFIG_KENSAKU = 'search_mail_template';
    const ADD_CONFIG_NAME = "ほげほげ";

    // 順番指定
    /**
     * @param array $meta
     * @param ContainerInterface $container
```

```
     */
    public function enable(array $meta, ContainerInterface $container) {
        $this->createPage($container);
        $this->createMailTemplate($container);
        $this->createConfig($container);
    }

    // 順番指定
    /**
     * @param array $meta
     * @param ContainerInterface $container
     */
    public function disable(array $meta, ContainerInterface $container) {
        $this->deleteConfig($container);
        $this->deleteMailTemplate($container);
        $this->deletePage($container);
    }

    //後で修正

    /**
     * dtb_page & dtb_page_layout レコード作成
     *
     * @param ContainerInterface $container
     */
    private function createPage(ContainerInterface $container) {

        // 固定カラムであるurlを検索し、そのレコードの存在をチェックする。なければif文より下を実行。
        $pageRepository = $container->get(PageRepository::class);
        $pageFindResult = $pageRepository->findOneBy(["url" => $this::ADD_PAGE_URL]);
        if (is_null($pageFindResult) == false) return;

        // dtb_layout id = 2のレコードを検索し、取得する。
        $layoutRepository = $container->get(LayoutRepository::class);
        $underLayout = $layoutRepository->findOneBy(["id" => $this::ADD_PAGE_LAYOUT_ID]);

        // 今あるレコードのsort_noカラムを降順にし、それに+1入れた値を取得し、レコード作成の際にsetする。（連番）
        // ただし、sort_noカラム自体が何の役割をしているのかは不明。連番になっていなかったり、値が重複していたりするため、尚のこと不明。
        $pageLayoutRepository = $container->get(PageLayoutRepository::class);
        $LastPageLayout = $pageLayoutRepository->findOneBy([], ['sort_no' => 'DESC']);
```

```
        $nextSortNo = $LastPageLayout->getSortNo() + 1;

        // エンティティマネージャーの呼び出し
        $em = $container->get('doctrine.orm.entity_manager');
        // commit が実行されるまで自動コミットがストップする。※pageとpagelayoutは密接しているため、どちらか一方が先にコミットされるとエラーを起こす。
        $em->beginTransaction();

        // レコード作成にあたって、各カラムにデータをセット
        $page = $pageRepository->newPage();
        $page->setName($this::ADD_PAGE_NAME)
            ->setUrl($this::ADD_PAGE_URL)
            ->setFileName($this::ADD_PAGE_FILE_NAME)
            ->setEditType($this::ADD_PAGE_EDIT_TYPE)
            ->setMetaRobots($this::ADD_PAGE_META_ROBOTS);
        $em->persist($page);
        $em->flush($page);

        // レコード作成にあたって、各カラムにデータをセット
        $pageLayout = new PageLayout();
        $pageLayout->setLayout($underLayout)
            ->setLayoutId($underLayout->getId())
            ->setPageId($page->getId())
            ->setSortNo($nextSortNo)
            ->setPage($page);
        $em->persist($pageLayout);
        $em->flush($pageLayout);
        // 自動コミットがストップされていたので、pageとpagelayoutの両方がまとめて実行される。
        $em->commit();
    }

    /**
     * dtb_page & dtb_page_layout レコード削除
     *
     * @param ContainerInterface $container
     */
    private function deletePage(ContainerInterface $container) {

        // 固定カラムであるurlを検索し、そのレコードの存在をチェックする。あればif文より下を実行
        $pageRepository = $container->get(PageRepository::class);
        $page = $pageRepository->findOneBy(["url" => $this::ADD_PAGE_URL]);
        if (is_null($page)) return;
```

```
        // エンティティマネージャーの呼び出し
        $em = $container->get('doctrine.orm.entity_manager');
        // commit が実行されるまで自動コミットがストップする。※pageとpagelayoutは密接しているため、どちらか一方が先にコミットされるとエラーを起こす。
        $em->beginTransaction();

        $em->remove($page);
        $em->flush($page);

        // dtb_pagelayoutのレコードを有効化の際に与えられたpage_idで検索して取得する。※連動しているdtb_pageのidは消滅しているため、何も取得できないのが正常な動作。
        $pageLayoutRepository = $container->get(PageLayoutRepository::class);
        $pageLayout = $pageLayoutRepository->findOneBy(["page_id" => $page->getId()]);

        // $pagelayout に何も入っていなければif文以下を実行。
        if(is_null($pageLayout) === false){
            $em->remove($pageLayout);
            $em->flush($pageLayout);
        }
        // 自動コミットがストップされていたので、pageとpagelayoutの両方がまとめて実行される。
        $em->commit();
    }

    /**
     * dtb_mail_template レコード作成
     *
     * @param ContainerInterface $container
     */
    private function createMailTemplate(ContainerInterface $container) {

        // 固定データのカラムであるfile_nameを検索し、そのレコードが存在するかチェックする。なければif文より下を実行。
        $mailTemplateRepository = $container->get(MailTemplateRepository::class);
        $mailTemplateFindResult = $mailTemplateRepository->findOneBy(["file_name" => $this::ADD_MAIL_TEMPLATE_FILE_NAME]);
        if (is_null($mailTemplateFindResult) == false) return;

        // エンティティマネージャーの呼び出し
        $em = $container->get('doctrine.orm.entity_manager');

        // クラスのインスタンス化
        $mailTemplate = new MailTemplate();
```

```
        // レコード作成にあたって、各カラムにデータをセット
        $mailTemplate->setName($this::ADD_MAIL_TEMPLATE_NAME)
            ->setFileName($this::ADD_MAIL_TEMPLATE_FILE_NAME)
            ->setMailSubject($this::ADD_MAIL_TEMPLATE_MAIL_SUBJECT);
        // $em  に $mailTemplateで作成したSQL文を代入する。
        $em->persist($mailTemplate);
        // SQL文の実行
        $em->flush($mailTemplate);
    }

    /**
     * dtb_mail_template レコード削除
     *
     * @param ContainerInterface $container
     */
    private function deleteMailTemplate(ContainerInterface $container) {

        // 固定データのカラムであるfile_nameを検索し、それが存在するかチェックする。あればif文より下を実行。
        $mailTemplateRepository = $container->get(MailTemplateRepository::class);
        $mailTemplate = $mailTemplateRepository->findOneBy(["file_name" => $this::ADD_MAIL_TEMPLATE_FILE_NAME]);
        if (is_null($mailTemplate)) return;

        // エンティティマネージャーの呼び出し
        $em = $container->get('doctrine.orm.entity_manager');

        // $em  に $mailTemplateで作成したSQL文を代入し、削除予定状態する。
        $em->remove($mailTemplate);
        // SQL文の実行
        $em->flush($mailTemplate);
    }

    /**
     * @param ContainerInterface $container
     */
    public function createConfig(ContainerInterface $container)
    {
        // dtb_mail_templateの固定カラムであるfile_nameを検索し、そのレコードのidを取得する。※本テーブルのmail_template_idカラムにセットするため。
        $mailTemplateRepository = $container->get(MailTemplateRepository::class);
        $mailTemplate = $mailTemplateRepository->findOneBy(["file_name" => $this::ADD_MAIL_TEMPLATE_FILE_NAME]);
```

```
        $last_insert_id = $mailTemplate->getId();

            $em = $container->get('doctrine.orm.entity_manager');

        /** @var ConfigRepository $configRepository */
        $configRepository = $em->getRepository(Config::class);
        $config = $configRepository->get();

        if (!$config) {
            $config = new Config();
            $config->setName($this::ADD_CONFIG_NAME);
            $config->setMailTemplateId($last_insert_id);
            $config->setKensaku($this::ADD_CONFIG_KENSAKU);
            $em->persist($config);
            $em->flush();
        }
    }

    /**
     * @param ContainerInterface $container
     */
    private function deleteConfig(ContainerInterface $container) {
        $configRepository = $container->get(ConfigRepository::class);
        $config = $configRepository->findOneBy(["kensaku" => $this::ADD_CONFIG_KENSAKU]);
        if (is_null($config)) return;

        $em = $container->get('doctrine.orm.entity_manager');

        $em->remove($config);
        $em->flush($config);
    }
}
```