Message broker et programmation asynchrone - Symfony Messenger

Objectif

L'objectif de ce TP est de créer une application Symfony qui utilise Symfony Messenger pour traiter les tâches asynchrones. Vous allez créer un système de gestion de commandes qui utilise les messages pour séparer les tâches de traitement de commandes. Vous apprendrez à configurer le bus de message, à créer des messages et des handlers, à utiliser les middleware pour ajouter des fonctionnalités supplémentaires, et à tester votre application.

Livrable

Un projet symfony fonctionnel avec un docker compose permettant de démarrer toute la stack nécessaire (à minima PHP + serveur web + RabbitMQ).

Étapes

Étape 1 : Configuration

Commencez par installer Symfony en utilisant Composer (ou autre). Ensuite, installez le composant Symfony Messenger.

Étape 2 : Messages

Créez un message `OrderMessage` qui contient les informations sur une commande, telles que l'identifiant de la commande, le nom du client et l'article commandé (ce ne sont que des exemples, ne vous focalisez pas là-dessus). Créez également un message `OrderProcessedMessage` qui est utilisé pour signaler que la commande a été traitée avec succès.

Étape 3 : Handlers

Créez un handler pour le message `OrderMessage` qui traite la commande. Ce handler doit utiliser une ou plusieurs classes de services pour effectuer les actions nécessaires, telles que la mise à jour de la base de données, l'envoi d'e-mails de confirmation, etc (encore une fois faites simple : ce qui m'intéresse est de voir du code consommer le message, pas tellement la logique métier que vous allez implémenter)..

Créez également un handler pour le message `OrderProcessedMessage`.

Étape 4 : Bus de message

Configurez le bus de message pour utiliser RabbitMQ. Utilisez les annotations/attributs pour lier les messages aux handlers correspondants. Ensuite, utilisez le `MessageBusInterface` pour envoyer les messages (pour cela, utilisez un contrôleur, une commande symfony, … Ce que vous voulez tant que c'est simple à déclencher).

Étape 5 : Middleware

Utilisez un middleware de votre choix pour ajouter des fonctionnalités supplémentaires à votre application. Par exemple, vous pouvez utiliser le middleware `ValidationMiddleware` pour valider les données envoyées avec les messages.

Bonus : Tests

Créez des tests pour votre application en utilisant PHPUnit. Écrivez des tests pour chaque fonctionnalité de votre application, y compris les tests pour l'envoi de messages, les tests pour la validation des données, les tests pour la mise à jour de la base de données si vous en avez une, etc.