

Comment programmer un tableau d'évaluation des stocks méthode PEPS en symfony avec les getters et les setters


Pour programmer un tableau d'évaluation des stocks avec la méthode PEPS (Premier Entré, Premier Sorti) en Symfony, il faut procéder en plusieurs étapes. Nous allons aborder la conception de l'entité représentant un article de stock et le calcul de l'évaluation des stocks en utilisant la méthode PEPS, tout en intégrant les getters et setters pour interagir avec les données.

Voici un guide pour y parvenir :

1. Créer l'entité "Stock" (ou "ArticleStock")

Cette entité représente un article en stock avec les informations nécessaires pour appliquer la méthode PEPS. Elle contient des attributs tels que id, quantité, prix_achat, et date_entrée pour suivre l'ordre des entrées et sorties.

// src/Entity/Stock.php

 Copier le code

```
namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass=StockRepository::class)
 */
class Stock
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="integer")
     */
    private $quantite;

    /**
     * @ORM\Column(type="float")
     */
    private $prixAchat;

    /**
     * @ORM\Column(type="datetime")
     */
}
```



```
private $dateEntree;

// Getters et Setters
public function getId(): ?int
{
    return $this->id;
}

public function getQuantite(): ?int
{
    return $this->quantite;
}

public function setQuantite(int $quantite): self
{
    $this->quantite = $quantite;
    return $this;
}

public function getPrixAchat(): ?float
{
    return $this->prixAchat;
}

public function setPrixAchat(float $prixAchat): self
{
    $this->prixAchat = $prixAchat;
    return $this;
}
```

```
public function getDateEntree(): ?\DateTimeInterface
{
    return $this->dateEntree;
}

public function setDateEntree(\DateTimeInterface $dateEntree): self
{
    $this->dateEntree = $dateEntree;
    return $this;
}
}
```

2. Créer le tableau d'évaluation des stocks avec la méthode PEPS


Dans la méthode PEPS, les premiers articles entrés dans le stock sont les premiers à être sortis. Cela signifie que pour chaque sortie, nous devons calculer le prix d'achat en prenant en compte les articles les plus anciens.

Pour cela, il faut une méthode qui simule une sortie de stock en appliquant la méthode PEPS.

3. Implémentation de la méthode de sortie et calcul de l'évaluation PEPS

Vous pouvez créer une méthode dans votre service ou dans votre contrôleur pour effectuer une sortie de stock en utilisant la méthode PEPS. Voici un exemple dans un service dédié au calcul :

// src/Service/StockService.php

 Copier le code

```
namespace App\Service;

use App\Entity\Stock;
use Doctrine\ORM\EntityManagerInterface;

class StockService
{
    private $em;

    public function __construct(EntityManagerInterface $em)
    {
        $this->em = $em;
    }

    /**
     * Méthode pour évaluer les stocks avec la méthode PEPS
     *
     * @param int $quantiteSortie
     * @return float Le coût total de la sortie selon la méthode PEPS
     */
    public function sortiePeps(int $quantiteSortie): float
    {
        // Récupérer tous les stocks ordonnés par date d'entrée (ancien au plus récent)
        $stocks = $this->em->getRepository(Stock::class)->findBy([], ['dateEntree' => 'ASC'])

        $quantiteRestante = $quantiteSortie;
        $coutTotal = 0;

        foreach ($stocks as $stock) {
```

```
// Si la quantité à sortir est inférieure ou égale à la quantité en stock, utili
if ($quantiteRestante <= $stock->getQuantite()) {
    $coutTotal += $quantiteRestante * $stock->getPrixAchat();
    // Réduire la quantité de stock après la sortie
    $stock->setQuantite($stock->getQuantite() - $quantiteRestante);
    $this->em->persist($stock);
    break;
} else {
    // Sinon, sortir tout le stock disponible et passer au suivant
    $coutTotal += $stock->getQuantite() * $stock->getPrixAchat();
    $quantiteRestante -= $stock->getQuantite();
    $stock->setQuantite(0);
    $this->em->persist($stock);
}
}

$this->em->flush(); // Sauvegarder les modifications

return $coutTotal; // Retourner le coût total de la sortie
}
}
```

4. Utilisation dans un contrôleur

Vous pouvez appeler ce service depuis un contrôleur pour effectuer une sortie de stock.

```
// src/Controller/StockController.php

namespace App\Controller;

use App\Service\StockService;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class StockController extends AbstractController
{
    private $stockService;

    public function __construct(StockService $stockService)
    {
        $this->stockService = $stockService;
    }

    /**
     * @Route("/sortie-stock", name="sortie_stock")
     */
    public function sortieStock(Request $request): Response
    {
        $quantiteSortie = $request->query->getInt('quantite', 0);

        if ($quantiteSortie > 0) {
            $coutTotal = $this->stockService->sortiePeps($quantiteSortie);
            return new Response("La sortie ↓ stock a coûté : " . $coutTotal . "€");
        }

        return new Response("Veuillez fournir une quantité valide.");
    }
}
```

5. Tests et gestion des stocks

1. Vous pouvez maintenant tester votre code en effectuant des sorties de stock via l'URL `/sortie-stock?quantite=10` et observer comment la méthode PEPS s'applique sur vos entrées de stock.
2. N'oubliez pas de gérer correctement les cas où il n'y a pas assez de stock disponible ou de mettre en place des validations supplémentaires selon votre logique métier.

6. Conclusion

Vous avez ainsi créé un système qui permet de gérer un stock avec la méthode PEPS en Symfony, en utilisant les entités, les getters et setters, ainsi que les services pour la gestion des sorties de stock et le calcul de leur coût. Vous pouvez adapter ce code à vos besoins en fonction des spécificités de votre projet.