

# Module M2105 - RT web dyna - TD/P 4

Prise en main du micro-framework

[Correction du TP](#)

## -- Installation

- Télécharger ou cloner le projet <https://github.com/jcheron/RT-Cloud>
- Copier les fichiers dans le dossier **htdocs** de votre serveur.
- Renommer éventuellement le dossier **Cloud-master** en **Cloud**
- Exécuter le script de création de la base de données (**app/database/cloud.sql**) à partir de phpMyAdmin (<http://127.0.0.1/phpmyadmin/>)

## -- Ressources

- [Prise en main Micro-framework](#)
- [Documentation API Micro-framework et cloud](#)
- [Twig documentation](#)

## -- Exercices

### Exercice 1 : Contrôleur et chargement de données

1. Créer un contrôleur **Exemples**,
2. charger les services (instances de la classe Service) dans la méthode **index**.
3. Affichez les services directement depuis le contrôleur,
4. Testez le résultat à l'adresse **/Exemples** ou **/exemples/index**

voir [DAO::getAll](#)



### Exercice 2 : contrôleur et vue

1. Dans le contrôleur **Exemples**, charger les Utilisateurs dans la méthode **users**.
2. Créer une vue **users.php** dans le dossier **views/exemples/** permettant de les afficher.

voir [Controller::loadView](#)

🏠 Accueil

# Utilisateurs

eAllman  
jAppelbaum  
kdMitnick  
djbernstein  
linus.torvals

### Exercice 3 : contrôleur et vue Twig

1. Dans le contrôleur **Exemples**, charger les Disques dans la méthode **disques**.
2. Créer une vue **disques.html** (utilisant Twig) dans le dossier **views/exemples/**.
3. Afficher le nom du disque et l'utilisateur associé.

🏠 Accueil

# Disques

**Datas** eric.allman@gmail.com-eAllman (Administrateur)  
**System** eric.allman@gmail.com-eAllman (Administrateur)  
**Jacob datas** jacob.appelbaum@gmail.com-jAppelbaum (Utilisateur)  
**System** jacob.appelbaum@gmail.com-jAppelbaum (Utilisateur)  
**Server web** jacob.appelbaum@gmail.com-jAppelbaum (Utilisateur)  
**Archives** eric.allman@gmail.com-eAllman (Administrateur)

### Exercice 4 : chargement de données avec paramètres

1. Créer la méthode **sortedUsers** pour qu'elle prenne en paramètre le champ **field** sur lequel on effectuera un tri, et un deuxième déterminant l'ordre de tri (**order** ASC ou DESC) : le champ par défaut sera "login" et l'ordre "ASC".
2. Créer le template **sortedUsers.html** pour qu'il affiche le champ sur lequel s'effectue le tri, son ordre, et qu'il puisse le modifier.
3. Afficher les colonnes login, mail et tel dans un tableau

🏠 Accueil		
login ↓	mail	tel
djberstein	djberstein@gmail.com	1
eAllman	eric.allman@gmail.com	
jAppelbaum	jacob.appelbaum@gmail.com	
kdMitnick	kd.mitnick@gmail.com	
linus.torvals	linus.torvals@gmail.com	

### Exercice 5 : chargement de données liées

1. Créer la méthode **usersDisques** chargeant les utilisateurs, et leurs disques (les disques de chaque utilisateur doivent être chargés explicitement)
2. Créer une vue **userDisques.html** affichant un utilisateur et ses disques
3. Le résultat à obtenir doit permettre d'afficher tous les utilisateurs (nom), et leurs disques (nom).

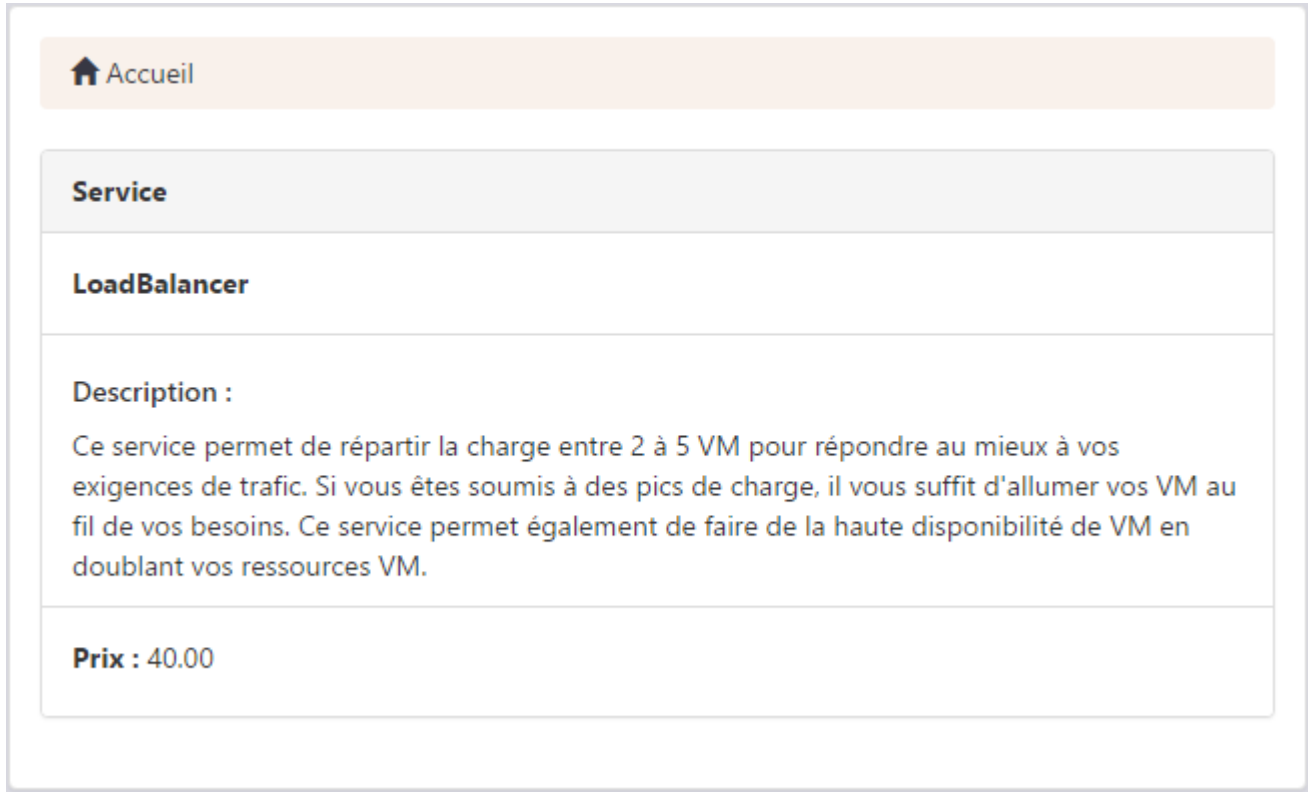
voir [DAO::getOneToMany](#)

🏠 Accueil	
<b>eric.allman@gmail.com-eAllman (Administrateur)</b>	
Datas:eAllman	
System:eAllman	
Archives:eAllman	
<b>jacob.appelbaum@gmail.com-jAppelbaum (Utilisateur)</b>	
Jacob datas:jAppelbaum	
System:jAppelbaum	
Server web:jAppelbaum	

## Exercice 6 : chargement d'un objet

1. Créer une méthode **displayService(\$id=null)** permettant de charger un service pas son id, ou d'en instancier un nouveau
2. **displayService** doit ensuite afficher la vue **views/exemples/displayService.html** affichant l'objet **\$service**

voir [DAO:getOne](#)



The screenshot shows a web application interface. At the top, there is a navigation bar with a home icon and the text 'Accueil'. Below this, there is a main content area with a header 'Service'. Under the header, the name 'LoadBalancer' is displayed. Below the name, there is a section titled 'Description :'. The description text reads: 'Ce service permet de répartir la charge entre 2 à 5 VM pour répondre au mieux à vos exigences de trafic. Si vous êtes soumis à des pics de charge, il vous suffit d'allumer vos VM au fil de vos besoins. Ce service permet également de faire de la haute disponibilité de VM en doublant vos ressources VM.' At the bottom of the card, the price is listed as 'Prix : 40.00'.

## Exercice 7 : ajout d'instance

1. Dans le contrôleur **Exemples**, créer une méthode **serviceAdd(\$nom,\$prix=0)** permettant de créer un service de nom \$nom et de prix \$prix
2. Afficher ensuite le service ajouté en appelant la méthode **displayService**
3. Tester en allant à l'adresse **Exemples/serviceAdd/Test de service offert/** puis **Exemples/serviceAdd/Service à 5 euros/5/**

voir [DAO:insert](#)

## Exercice 8 : modification d'instance

1. Dans la vue **displayService**, afficher un bouton "Augmenter prix" si le service est existant (\$edit=true)
2. Créer la méthode **updatePrix(\$idService,\$update=1)** augmentant le prix de \$update

voir [DAO:update](#)

<b>Service</b>
<b>LoadBalancer</b>
<b>Description :</b> Ce service permet de répartir la charge entre 2 à 5 VM pour répondre au mieux à vos exigences de trafic. Si vous êtes soumis à des pics de charge, il vous suffit d'allumer vos VM au fil de vos besoins. Ce service permet également de faire de la haute disponibilité de VM en doublant vos ressources VM.
<b>Prix : 40.00</b> <input type="button" value="Augmenter prix"/>

## Exercice 9 : suppression d'instance

1. Dans la vue **displayService**, afficher un bouton "Supprimer service" si le service est existant (\$edit=true)
2. Créer la méthode **deleteService(\$idService)** permettant de supprimer le service d'id **\$idService**
3. Afficher un message en cas de succès ou d'échec.

voir [DAO:delete](#)

<b>Service</b>
<b>LoadBalancer</b>
<b>Description :</b> Ce service permet de répartir la charge entre 2 à 5 VM pour répondre au mieux à vos exigences de trafic. Si vous êtes soumis à des pics de charge, il vous suffit d'allumer vos VM au fil de vos besoins. Ce service permet également de faire de la haute disponibilité de VM en doublant vos ressources VM.
<b>Prix : 40.00</b> <input type="button" value="Augmenter prix"/>
<input type="button" value="Supprimer service"/>

## Exercice 10 : CRUD avec \_defaultController

1. Créer le contrôleur **Services** héritant de **\_DefaultController**
2. Affecter "Service" à son membre **model**

voir [Controller\\_DefaultController](#)

Accéder à l'url **/services** ou **/services/index** pour visualiser le résultat

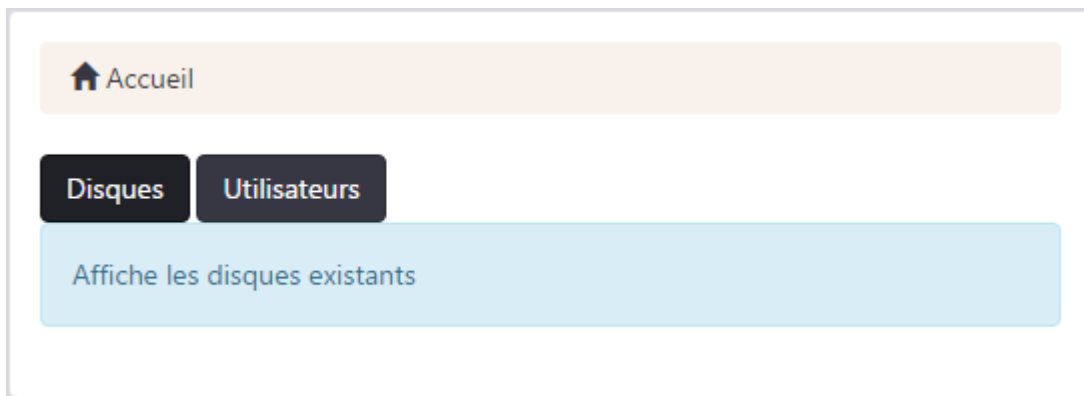
1. Implémenter la méthode **frm(\$id=NULL)** permettant d'afficher le formulaire de modification d'un service :
2. Créer la vue frmServices.html affichant un formulaire de modification du service, et dont l'action est **Services/update** :

Tester l'ajout, la modification et la suppression de services.

## Exercice 11 : Ajax

1. Créer une méthode **ajaxTest()** dans le contrôleur **Exemples**
2. La méthode **ajaxTest** doit afficher la vue **views/exemples/ajaxTest.html** ci-dessous :

voir [jQuery](#)



- Les boutons "Disques" et "Utilisateurs" doivent afficher via ajax les résultats des urls **exemples/disques** et **exemples/users**
- Le passage de la souris au dessus des boutons doit afficher un message dans l'alert Bootstrap sur le rôle respectif des boutons.

From:  
<http://slamwiki2.kobject.net/> - **Broken SlamWiki 2.0**

Permanent link:  
<http://slamwiki2.kobject.net/php-rt/tp4-old?rev=1526204380>

Last update: **2019/08/31 14:25**

