Module M2105 - RT web dyna - TD/P 4

Prise en main du projet RT-Cloud et du micro-framework

-- 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 **RT-Cloud-master** en **RT-Cloud**

-- Exercices

Exercice 1

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**

```
class Exemples extends \BaseController {

    public function index() {
        $services=DAO::getAll("Service");
        foreach ($services as $service){
            echo $service->getNom()."<br>";
        }
    }
}
```

Exercice 2

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

```
class Exemples extends \BaseController {
    ...

    public function users(){
        $users=DAO::getAll("Utilisateur");
        $this->loadView("Exemples/users",array("users"=>$users));
    }
}
```

```

```

```
<h1>Utilisateurs</h1>
<?php
foreach ($users as $user){
    echo $user->getLogin()."<br>";
}
```

Exercice 3

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.

```
class Exemples extends \BaseController {
    ...

    public function disques(){
        $disques=DAO::getAll("Disque");
        $this->loadView("Exemples/disques.html",array("disques"=>$disques));
    }
}
```

```
<h1>Disques</h1>
{% for disque in disques %}
<b>{{ disque.getNom() }}</b> {{ disque.getUtilisateur() }}<br>
{% endfor %}
```

Exercice 4

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

```
class Exemples extends \BaseController {
    ...
    public function sortedUsers($field="login",$order="ASC"){
        $fields=["login","mail","tel"];
        if(!array_search($field,$fields))
            $field=$fields[0];
        $header="";
        foreach ($fields as $f){
            $header.="<th>";
            $icon="sort-by-alphabet";
            $url="Exemples/sortedUsers/".$f;
            if($field==$f){
                if(strtoupper($order)=="ASC"){
```

```
                    $icon="sort-by-alphabet-alt";
                    $url.="/DESC";
                }
                $header.="<a href='{$url}'>{$f} <span class='glyphicon
glyphicon-{$icon}' aria-hidden='true'></span></a>";
            }else{
                $header.="<a href='{$url}'>{$f}</a>";
            }
            $header.="</th>";
        }
        $users=DAO::getAll("Utilisateur","1=1 ORDER BY ".$field." ".$order);


$this->loadView("Exemples/sortedUsers.html",array("users"=>$users,"header"=>$header
));
    }
```

```
<table class='table table-striped'>
    <thead>
    <tr>
        {{header|raw}}
    </tr>
    </thead>
        <tbody>
        {% for user in users %}
            <tr>
            <td>{{user.getLogin()}}</td>
            <td>{{user.getMail()}}</td>
            <td>{{user.getTel()}}</td>
            </tr>
        {% endfor %}
        </tbody>
</table>
```

Exercice 5

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).

```
class Exemples extends \BaseController {
    ...
    public function usersDisques(){
        $users=DAO::getAll("Utilisateur");
        foreach ($users as $user) {
            DAO::getOneToMany($user, "disques");
            if(sizeof($user->getDisques())>0)
            $this->loadView("exemples/userDisques.html",array("user"=>$user));
        }
    }
```

```
}
```

```
<h4>{{user}}</h4>
<ul class="list-group">
{% for disque in user.getDisques() %}
    <li class="list-group-item">{{disque}}</li>
{% endfor %}
</ul>
```

Exercice 6

1. Créer une méthode **frmService($id=null)** permettant de charger un service pas son id, ou d'en instancier un nouveau
2. **frmService** doit ensuite afficher la vue views/exemples/frmService.html affichant un formulaire d'ajout ou de modification de l'objet **$service**
3. L'action du formulaire doit renvoyer vers **Exemples/submitService/**

```
class Exemples extends \BaseController {
    ...
    public function frmService($id=null){
        $edit=false;
        $service=null;
        if(isset($id)){
            $service=DAO::getOne("Service", $id);
            $edit=isset($service);
        }
        if(!$edit)
            $service=new Service();
        $this->loadView("exemples/frmService.html",array("service"=>$service));
    }
}
```

```
<form action="Exemples/submitService" method="post">
    <fieldset>
        <legend>Service</legend>
        <div class="form-group">
            <input type="hidden" name="id" id="id" value="{{service.getId()}}">
            <input placeholder="Entrez un nom" type="text" class="form-control"
name="nom" id="nom" value="{{service.getNom()}}">
            <input placeholder="Enrez un prix" type="number" class="form-control"
name="prix" id="prix" value="{{service.getPrix()}}">
            <textarea rows="5" class="form-control" name="description"
id="description">{{service.getDescription()}}</textarea>
        </div>
        <div class="form-group">
            <button class="btn btn-primary">Valider</button>
        </div>
```

```
    </fieldset>
</form>
```

Exercice 7

1. Créer la méthode **submitService** du contrôleur **Exemples** permettant de faire l'ajout ou la modification du service posté dans le formulaire précédent.
2. Afficher un message relatif à l'opération effectuée : Ajout, modification ou erreur

```
class Exemples extends \BaseController {
    ...
    public function submitService(){
        $edit=false;
        $service=null;
        if(isset($_POST["id"])){
            $service=DAO::getOne("Service", $_POST["id"]);
            $edit=isset($service);
        }
        try{
            if($edit){
                $this->_updateService($service);
                if(DAO::update($service)){
                    $this->_showMessage("`<b>".$service."</b>` mis à jour");
                }else{
                    $this->_showMessage("Impossible de sauvegarder
`<b>".$service."</b>`","warning");
                }
            }else{
                $service=new Service();
                $this->_updateService($service);
                if(DAO::insert($service)){
                    $this->_showMessage("`<b>".$service."</b>` ajouté");
                }else{
                    $this->_showMessage("Impossible d'ajouter
`<b>".$service."</b>`","warning");
                }
            }
        }catch(Exception $e){
            $this->_showMessage("Impossible de sauvegarder `<b>".$service."</b>` :
".$e->getMessage(),"danger");
        }
    }

    private function _showMessage($message,$type="success"){
$this->loadView("main/vInfo",array("message"=>$message,"type"=>$type,"dismissable"=
>false,"visible"=>true));
    }

    private function _updateService(&$service){
        if(StrUtils::isNotNull($_POST["nom"]))
            $service->setNom($_POST["nom"]);
        $service->setDescription($_POST["description"]);
        $service->setPrix($_POST["prix"]);
```

```
    }
}
```