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

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 : 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**

```
class Exemples extends \BaseController {

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

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/**.

```
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 : 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.

```
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 : 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

```
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 : 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).

```
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 : 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**

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

```
<div class="panel panel-default">
    <div class="panel-heading">
        <b>Service</b>
    </div>
    <div class="panel-body"><b>{{service.getNom()}}</b></div>
    <ul class="list-group">
        <li class="list-group-item">
            <h5>Description : </h5> <span>{{service.getDescription()}}</span>
        </li>
        <li class="list-group-item">
            <b>Prix : </b> {{service.getPrix()}}
        </li>
    </ul>
</div>
```

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/**

```
class Exemples extends \BaseController {
    ...
    public function serviceAdd($nom,$prix=0){
        $service=new Service();
        $service->setNom($nom);
        $service->setPrix($prix);
        DAO::insert($service);
        $this->forward("Exemples","displayService",$service->getId());
    }
}
```

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

```
<div class="panel panel-default">
    <div class="panel-heading">
        <b>Service</b>
    </div>
    <div class="panel-body"><b>{{service.getNom()}}</b></div>
    <ul class="list-group">
        <li class="list-group-item">
            <h5>Description : </h5> <span>{{service.getDescription()}}</span>
        </li>
        <li class="list-group-item">
            <b>Prix : </b> {{service.getPrix()}} {%if edit %} <a class="btn btn-
primary" href="exemples/updatePrix/{{service.getId()}}/1">Augmenter</a>{% endif %}
        </li>
    </ul>
</div>
```

```
class Exemples extends \BaseController {
    ...
    public function updatePrix($idService,$update=1){
        $service=DAO::getOne("Service", $idService);
        if(isset($service)){
            $service->setPrix($service->getPrix()+$update);
            DAO::update($service);
            $this->forward("Exemples","displayService",$service->getId());
        }else{
```

```
            $this->_showMessage("Impossible de charger le service","warning");
        }
    }
}
```

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.

```
<div class="panel panel-default">
    <div class="panel-heading">
        <b>Service</b>
    </div>
    <div class="panel-body"><b>{{service.getNom()}}</b></div>
    <ul class="list-group">
        <li class="list-group-item">
            <h5>Description : </h5> <span>{{service.getDescription()}}</span>
        </li>
        <li class="list-group-item">
            <b>Prix : </b> {{service.getPrix()}}
            {%if edit %}
            <a class="btn btn-primary"
href="exemples/updatePrix/{{service.getId()}}/1">Augmenter prix</a>
            <a class="btn btn-warning"
href="exemples/deleteService/{{service.getId()}}/">Supprimer service</a>
            {% endif %}
        </li>
    </ul>
</div>
```

```
class Exemples extends \BaseController {
    ...
    public function deleteService($idService){
        $service=DAO::getOne("Service", $idService);
        if(isset($service)){
            DAO::delete($service);
            $this->_showMessage($service." supprimé","success");
        }else{
            $this->_showMessage("Impossible de charger le service","warning");
        }
    }
}
```

Exercice 10 : CRUD avec _defaultController

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

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

```
class Services extends \_DefaultController {

    public function __construct(){
        parent::__construct();
        $this->title="Services";
        $this->model="Service";
    }
}
```

1. Implémenter la méthode **frm($id=NULL)** permettant d'afficher le formulaire de modification d'un service :

```
    public function frm($id=NULL){
        $service=$this->getInstance($id);
        $this->loadView("exemples/frmService.html",array("service"=>$service));
    }
```

1. Créer la vue frmServices.html affichant un formulaire de modification du service, et dont l'action est **Services/update** :

```
<form action="services/update" 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 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 :

```
<div class="button-group">
    <a class="btn btn-primary disques" href="exemples/disques" data-
```

```
ajax="exemples/disques">Disques</a>
    <a class="btn btn-primary users" href="exemples/users" data-
ajax="exemples/users">Utilisateurs</a>
</div>
<div class="alert alert-info" id="message"></div>
<div id="response"></div>
```

- 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.

```
    public function ajaxTest(){
        $this->loadView("exemples/ajaxTest.html");
        Jquery::getOn("click", "a[data-ajax]", "","#response",array("attr"=>"data-
ajax"));
        Jquery::doJqueryOn(".disques", "mouseenter", "#message", "html","Affiche
les disques existants");
        Jquery::doJqueryOn(".users", "mouseenter", "#message", "html","Affiche les
utilisateurs existants");
        Jquery::doJqueryOn(".btn", "mouseout", "#message", "html");
        echo Jquery::compile();
    }
```

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/php-rt/tp4-corr

Last update: **2019/08/31 14:21**