

# Module M2105 - RT web dyna - TD/P 3

Suivre [Procédures pour install de Xampp, composer, Ubiquity](#) pour mettre en place Xampp et Ubiquity

## Notions abordées

- Utilisation de Git
- Accès à une base de données
- ORM et classes métier
- Chargement et affichage de données

## Création de la base de données

- Démarrer Mysql/MariaDb à partir de l'interface de contrôle de Xampp
- Depuis phpMyAdmin (<http://127.0.0.1/phpmyadmin/>), importer la base de données messagerie (fichier [messagerie.sql](#)) :

Seigneur: 127.0.0.1

Bases de données SQL État Comptes utilisateurs Exporter Importer Paramètres

### Importation dans le serveur courant

Import de données

**Fichier à importer :**

Le fichier peut être compressé (gzip, bzip2, zip) ou non.  
Le nom du fichier compressé doit se terminer par `.[format].[compression]`. Exemple: `.sql.zip`

Parcourir les fichiers : Choisir un fichier Sélection du script

Il est également possible de glisser-déposer un fichier sur n'importe quelle page.

Jeu de caractères du fichier : utf-8

**Options spécifiques au format :**

Mode de compatibilité SQL : NONE

Ne pas utiliser AUTO\_INCREMENT pour la valeur zéro

Exécuter Exécution

## - Création du projet & du repository github

### Création du projet

Avant de commencer faire:

```
composer global update
```

Créer le projet **tp3** en invite du commande à partir du dossier **htdocs** de XAMPP

```
cd htdocs  
ubiquity new tp3 -a
```

Ouvrir/créer ce projet avec votre IDE (Eclipse ou PHPStorm)

Démarrer le serveur Mysql à partir de Xampp et démarrer le serveur Ubiquity à partir du dossier **tp3** :

```
Ubiquity serve
```

## Publication sur github

Créer un repository **php-rt-tp3** sur votre compte github (<https://github.com>) :

## Create a new repository

A repository contains all the files for your project, including the revision history.

---

**Owner** jcheron / **Repository name** php-rt-tp3 ✓

Great repository names are short and memorable. Need inspiration? How about **upgraded-giggle**.

**Description (optional)**

---

**Public**  
Anyone can see this repository. You choose who can commit.

**Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

---

**Create repository**

## Création dans UbiquityMyAdmin

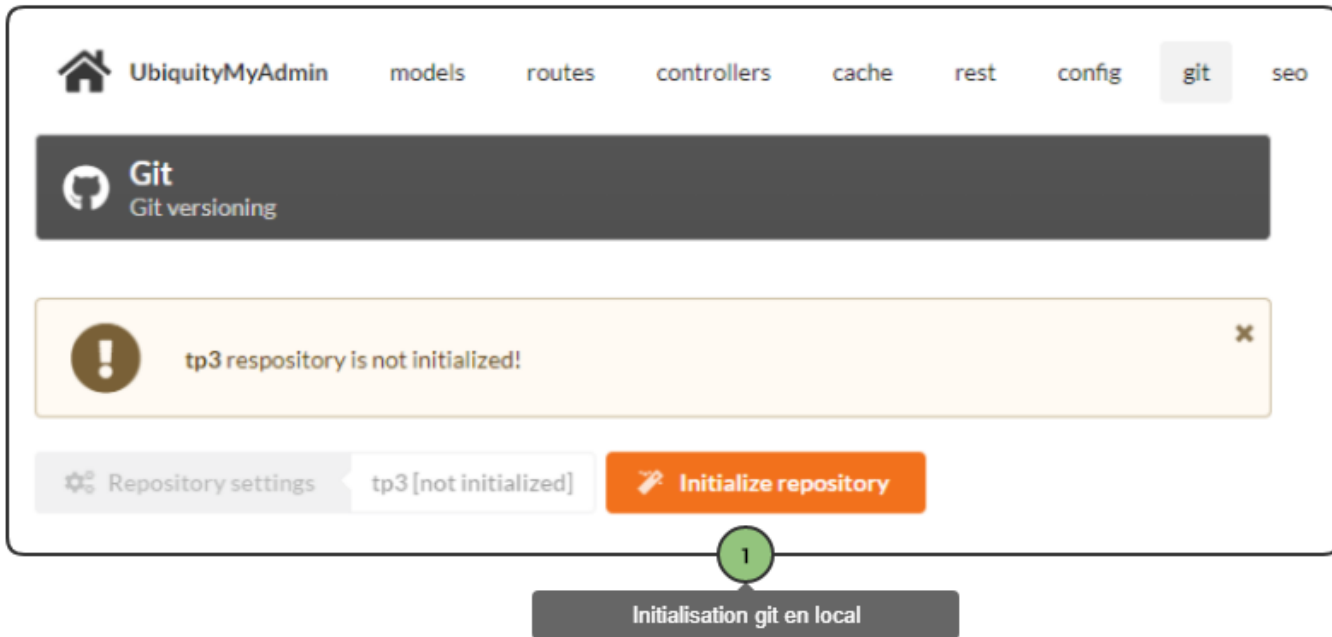
Assurez-vous que git est installé en local.  
Dans l'invite de commande, exécuter :

```
git --version
```

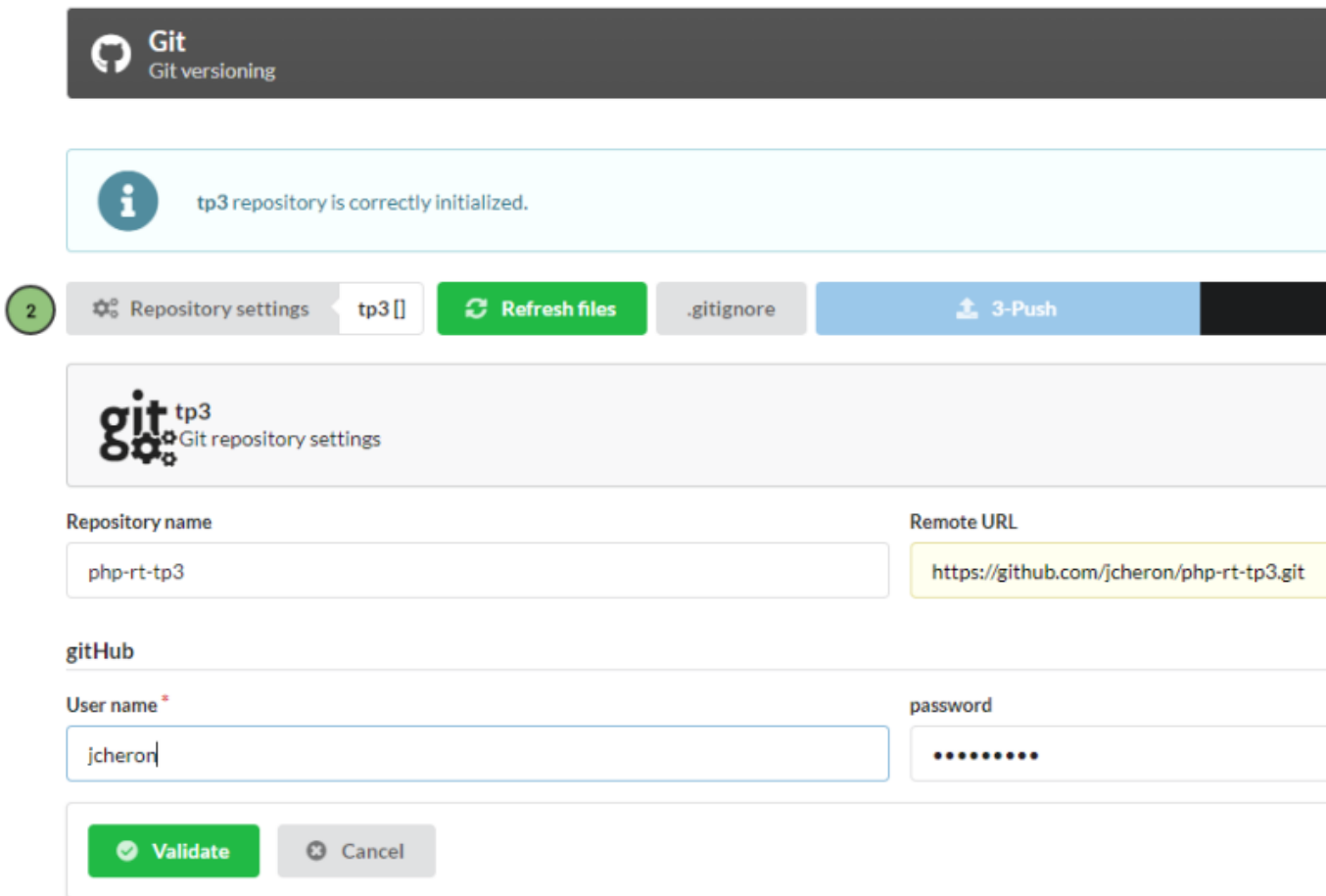
Si git n'est pas installé, téléchargez et installez le : <https://git-scm.com/downloads>

Activer la rubrique git d'UbiquityMyAdmin :

- Cliquer sur Initialize repository



- Saisir les informations de configuration du Repository :



### Premier commit

Effectuer un premier commit (version locale) du projet :

Changes History

 Files

- ? app/config/config.php
- ? app/config/services.php
- ? app/controllers/Admin.php
- ? app/controllers/ControllerBase.php
- ? app/controllers/Main.php
- ? app/models/Groupe.php
- ? app/models/Organization.php
- ? app/models/Organizationsettings.php
- ? app/models/Settings.php
- ? app/models/User.php
- ? app/views/Admin/cache/index.html
- ? app/views/Admin/config/index.html
- ? app/views/Admin/controllers/index.html
- ? app/views/Admin/data/diagClasses.html
- ? app/views/Admin/data/editTable.html

First commit git 2-Commit to master

description


## Push

Envoyer ce commit vers gitHub en faisant un Push

 Commit successfully completed!  
 • 40 new file(s) added

Repository settings php-rt-tp3 Refresh files .gitignore 3-Push

Changes History

| Hash  | Author  | Date          | Summary      |
|---|---------|---------------|--------------|
|  026e252 | jcheron | 3 seconds ago | First commit |

## - Génération des classes métier





Les classes métiers (models) permettent de faire la passerelle avec la base de données :

Il est possible de les générer automatiquement à partir d'**UbiquityMyAdmin** :

Aller dans models puis cliquer sur (Re-)create Models


UbiquityMyAdmin models routes controllers cache rest config git seo logs

**Models**  
Used to perform CRUD operations on data.

Engineering Forward  Conf Database configuration  Connexion Database connexion  Models Models generation  Cache Models cache generation

**Models generation**

- Database is well configured
- The connection to the database messagerie is established.
- Models namespace models is ok.
- The directory for models namespace C:\xampp\htdocs\tp3\app\models\ exists.

 **Models generation**

- No file found in C:\xampp\htdocs\tp3\app\models\ folder.

(Re-)Create models Classes diagram Models cache generation Cache >>

## Classe Organization

Exemple de la classe **Organization** :

Cette classe correspond à la table **organization** de la base de données. Les informations de mapping Objet/relationnel sont définies grâce aux annotations (voir [Models](#)).



```
<?php
namespace models;
class Organization{
    /**
     * @id
     * @column("name"=>"id","nullable"=>"","dbType"=>"int(11)")
     */
    private $id;

    /**
     * @column("name"=>"name","nullable"=>"","dbType"=>"varchar(100)")
     */
    private $name;

    /**
     * @column("name"=>"domain","nullable"=>"","dbType"=>"varchar(255)")
     */
    private $domain;

    /**
     * @column("name"=>"aliases","nullable"=>1,"dbType"=>"text")
     */
    private $aliases;

    /**
     * @oneToMany("mappedBy"=>"organization","className"=>"models\\Groupe")
     */
    private $groupes;

    /**
     *
     * @oneToMany("mappedBy"=>"organization","className"=>"models\\Organizationsettings")
     */
    private $organizationsettings;

    /**
     * @oneToMany("mappedBy"=>"organization","className"=>"models\\User")
     */
    private $users;

    public function getId(){
        return $this->id;
    }

    public function setId($id){
        $this->id=$id;
    }

    public function getName(){
        return $this->name;
    }

    public function setName($name){
        $this->name=$name;
    }
}
```

```
    public function getDomain(){
        return $this->domain;
    }

    public function setDomain($domain){
        $this->domain=$domain;
    }

    public function getAliases(){
        return $this->aliases;
    }

    public function setAliases($aliases){
        $this->aliases=$aliases;
    }

    public function getGroupes(){
        return $this->groupes;
    }

    public function setGroupes($groupes){
        $this->groupes=$groupes;
    }

    public function getOrganizationsettingss(){
        return $this->organizationsettingss;
    }

    public function setOrganizationsettingss($organizationsettingss){
        $this->organizationsettingss=$organizationsettingss;
    }

    public function getUsers(){
        return $this->users;
    }

    public function setUsers($users){
        $this->users=$users;
    }

    public function __toString(){
        return $this->domain;
    }
}
```

## Génération du cache

Les données ne sont accessibles qu'à condition que les informations de mappage aient été mises en cache (pour des raisons de performances) :

A partir de l'interface d'administration, choisir (Re-)init Models cache :

**Models cache generation**

- ! The models cache entry does not exists for the class models\Groupe.
- ! The models cache entry does not exists for the class models\Organization.
- ! The models cache entry does not exists for the class models\Organizationsettings.
- ! The models cache entry does not exists for the class models\Settings.
- ! The models cache entry does not exists for the class models\User.

(Re-)Init models cache    See datas

## Consultation des données

UbiquityMyAdmin    **models**    routes    controllers

**Models**  
Used to perform CRUD operations on data.

**Engineering**  
Forward    **Conf**  
Database configuration

|                      |     |
|----------------------|-----|
| Groupe               | 6   |
| Organization         | 2   |
| Organizationsettings | 2   |
| Settings             | 1   |
| User                 | 100 |

## Consultation des méta-données

Cliquer sur l'onglet Structure

# models\Organization

Data administration

Datas **Structure**

|                  |   |
|------------------|---|
| #tableName       | Organization  |
| #primaryKeys     | Array ( [0] => id )   |
| #manyToOne       | Array ( )   |
| #fieldNames      | Array ( [id] => id [name] => name [domain] => domain [aliases] => aliases [groupes] => groupes [organizationsettingss] => organizationsettingss [users] => users )  |
| #fieldTypes      | Array ( [id] => int(11) [name] => varchar(100) [domain] => varchar(255) [aliases] => text [groupes] => mixed [organizationsettingss] => mixed [users] => mixed )  |
| #nullable        | Array ( [0] => aliases )  |
| #notSerializable | Array ( [0] => groupes [1] => organizationsettingss [2] => users )  |
| #oneToMany       | Array ( [groupes] => stdClass Object ( [mappedBy] => organization [className] => models\Groupe ) [organizationsettingss] => stdClass Object ( [mappedBy] => organization [className] => models\Organizationsettingss ) [users] => stdClass Object ( [mappedBy] => organization [className] => models\User ) ) |

Class diagram

Cliquer sur le bouton Class diagram




A partir des données et méta-données, rédiger les règles de gestion relatives aux modèles :

- Organization
- User
- Group

## Commit git

Effectuer un second Commit pour la génération des classes métier :

Changes History

 Files

- ? app/models/Groupe.php
- ? app/models/Organization.php
- ? app/models/Organizationsettings.php
- ? app/models/Settings.php
- ? app/models/User.php

Add models| git 2-Commit to master

Models classes generation

### Modification de l'affichage des objets





La méthode **\_\_toString** des classes métier permet d'afficher un objet sous forme de chaîne ; le **\_\_toString** généré par défaut n'affiche pas forcément les informations adéquates d'un objet, comme le montre l'écran suivant :

## models\Organization

Data administration

Datas Structure

+ Add a new models\Organization...

| Id | Name                                       | Domain     | Aliases                  |   |
|----|--|------------|--------------------------|---|
| 1  | Conservatoire National des Arts et Métiers | lecnam.net | cnam-basse-normandie.fr; |   |
| 2  | Université de Caen-Normandie               | unicaen.fr |                          |   |

groupes (2)

|   |
|---|
| 1 |
| 2 |

organizationsettings (1)

|   |
|---|
| 1 |
|---|





users (12)

|                 |
|-----------------|
| wyatt.higgins   |
| jane.leon       |
| maggy.weber     |
| allistair.leon  |
| alvin.hatfield  |
| kendall.collier |
| steven.norman   |
| joan.hatfield   |
| ivor.logan      |
| hyacinth.finley |
| madeson.larsen  |
| maris.mosley    |

Modifier la méthode `__toString` des classes **User**, **Group** et **Organizationsettings** pour obtenir le résultat suivant :

Datas    Structure

[+ Add a new models\Organization...](#)

| Id | Name                                       | Domain     | Aliases                  |   |
|----|--|------------|--------------------------|---|
| 1  | Conservatoire National des Arts et Métiers | lecnam.net | cnam-basse-normandie.fr; |   |
| 2  | Université de Caen-Normandie               | unicaen.fr |                          |   |

groupes (2)

organizationsettings (1)

users (12)

Personnels

{{firstname}}{{lastname}}

Wyatt HIGGINS

Etudiants

Jane LEON

Maggy WEBER

Allistair LEON

Alvin HATFIELD

Kendall COLLIER

Steven NORMAN

Joan HATFIELD

Ivor LOGAN

Hyacinth FINLEY

Madeson LARSEN

Maris MOSLEY

## - Lecture et affichage de données

Créer un contrôleur **Organizations** et sa vue associée (via l'interface d'administration)

Créer un template de base :

```
{% block header %}
  <style>
    a.active{
      font-weight: bold;
      color: red;
    }
    a.active::after {
      content: " →";
    }
  </style>
  <h1>Messagerie Administration</h1>
{% endblock %}

{% block message %}
  {{ message | raw }}
{% endblock %}
```

```
{% block body %}  
{% endblock %}
```

## - Affichage des organisations

**Dans le contrôleur :** Chargement des organisations

```
namespace controllers;  
use Ubiquity\orm\DAO;  
  
/**  
 * Controller Organizations  
 **/  
class Organizations extends \Ubiquity\controllers\ControllerBase{  
  
    public function index(){  
        $organizations=DAO::getAll("models\\Organization");  
        $this->loadView("Organizations/index.html",["orgas"=>$organizations]);  
    }  
}
```

**Dans la vue :** affichage des organisations

```
<h1>Organisations</h1>  
<ul>  
    {% for orga in orgas %}  
        <li>{{orga}}</li>  
    {% endfor %}  
</ul>
```

**Résultat /Organizations/**

# Messagerie Administration

## Organisations

- lecnam.net
- unicaen.fr

## - Chargement et affichage d'une organisation

Créer l'action **display(\$idOrga)** dans le contrôleur **Organizations** à partir de l'interface d'administration :

L'accès à l'adresse **/Organizations/display/2** devra permettre d'afficher l'organisation d'id 2.

**Dans le contrôleur :** Chargement d'une organisation par son id :

```
namespace controllers;
use Ubiquity\orm\DAO;

/**
 * Controller Organizations
 **/
class Organizations extends \Ubiquity\controllers\ControllerBase{

    ...
    public function display($idOrga){
        $orga=DAO::getOne("models\\Organization", $idOrga);
        $this->loadView("Organizations/display.html",["orga"=>$orga]);
    }
}
```

**Dans la vue :** affichage d'une organisation

```
{% extends "base.html" %}
{% block body %}
<h2 class="ui header">
    {{orga.name}}
    <div class="sub header">{{orga.domain}}</div>
</h2>
{% endblock %}
```

### Chargement et affichage des objets liés (en relation)

Modifier l'appel de la méthode **DAO::getOne** pour charger les instances de type **manyToOne** et **oneToMany** :

La méthode **getOne** charge dans ce cas les utilisateurs **users** et les **groupes** de l'organisation :

```
namespace controllers;
use Ubiquity\orm\DAO;

/**
 * Controller Organizations
 **/
class Organizations extends \Ubiquity\controllers\ControllerBase{

    ...
    public function display($idOrga){
        $orga=DAO::getOne("models\\Organization", $idOrga, true, true);
        $this->loadView("Organizations/display.html",["orga"=>$orga]);
    }
}
```

Les groupes sont affichés dans la vue **display.html** :

```
{% extends "base.html" %}
{% block body %}
<h2 class="ui header">
  {{orga.name}}
  <div class="sub header">{{orga.domain}}</div>
</h2>
<div class="ui two columns grid">
  <div class="four wide column">
    <h3 class="ui header">
      <i class="ui users icon"></i>
      <div class="content">Groupes</div>
    </h3>
    <ul>
      {% for groupe in orga.groupes %}
        <li>{{ groupe }}</li>
      {% endfor %}
    </ul>
  </div>
  <div class="twelve wide column">
    <div id="users">
    </div>
  </div>
</div>
{% endblock %}
```

Résultat /Organizations/display/1

# Messagerie Administration

## Université de Caen-Normandie

unicaen.fr

### Groupes

- Personnels
- Etudiants
- Enseignants
- Vacataires

### - Affichage des utilisateurs de l'organisation

Création d'une méthode retournant une liste d'utilisateurs au format HTML :

- Affichage de tous les utilisateurs (paramètre **\$users**)
- Appel d'une vue et retour dans une chaîne **loadView(...,true)**

```
class Organizations extends \Ubiquity\controllers\ControllerBase{
    ...

    protected function users($users=null){
        $title="Tous les utilisateurs";
        return
    }
    $this->loadView("Organizations/users.html",compact("users","title"),true);
}
```

Vue affichant les utilisateurs :

```
<h3 class="ui header">
    <i class="ui user icon"></i>
    <div class="content">{{title}}</div>
</h3>
<div class="ui four columns grid">
    {% for user in users %}
        <div class="column"><span class="ui label"><i class="ui user icon"></i> {{ user
    | raw }}</span></div>
    {% endfor %}
</div>
```

Modification de l'action **display** pour qu'elle affiche les utilisateurs (appel de la méthode **users** précédemment créée) :

```
class Organizations extends \Ubiquity\controllers\ControllerBase{
    ...
    public function display($idOrga,$idGroupe=null){
        $orga=DA0::getOne("models\\Organization", $idOrga,true,true);
        $users=$this->users($orga->getUsers());
    }
    $this->jquery->renderView("Organizations/display.html",["orga"=>$orga,"users"=>$users]);
}
```

Modification de la vue display.html pour qu'elle affiche les utilisateurs (variable **users**) :

```
...
<div class="twelve wide column">
    <div id="users">
        {{users | raw}}
    </div>
</div>
...
```

Résultat /Organizations/display/1

# Messagerie Administration

## Université de Caen-Normandie

unicaen.fr

### Groupes

- Personnels
- Etudiants
- Enseignants
- Vacataires

### Tous les utilisateurs

|  |  |   |   |
|--|--|---|---|
|  Benjamin SHERMAN |  Acton CARRILLO |  Zorita RODRIGUEZ |  Henry BEASLEY |
|  Kelsey WEBER     |  Olympia HUBER  |  Carolyn PACE     |  Levi BISHOP   |
|  Lionel MCCRAY    |  Jeremy BRYAN   |  Ava POLLARD      |  Baxter WISE   |

### - Affichage des Utilisateurs par groupe

Modifier la méthode **users** pour qu'elle affiche éventuellement les utilisateurs d'un groupe :

```
class Organizations extends \Ubiquity\controllers\ControllerBase{
    ...

    protected function users($idOrga,$idGroupe=null,$users=null){
        if(isset($idGroupe)){
            $group=DAO::getOne("models\\Groupe",$idGroupe,true,true);
            $title=$group->getName();
            $users=DAO::getManyToMany($group, "users");
        }else{
            $title="Tous les utilisateurs";
        }
        return
    }
    $this->loadView("Organizations/users.html",compact("users","title"),true);
}
```

Modifier la méthode display pour qu'elle prenne en compte l'affichage des utilisateurs d'un groupe :

```
...
public function display($idOrga,$idGroupe=null){
    $orga=DAO::getOne("models\\Organization", $idOrga,true,true);
    $users=$this->users($idOrga,$idGroupe,$orga->getUsers());
    $this->jquery->renderView("Organizations/display.html",["orga"=>$orga,"users"=>$users]);
}
...
```

Ajouter enfin des liens pour afficher les utilisateurs de chaque groupe :

```

...
    <ul>
      {% for groupe in orga.groupe %}
        <li><a href="Organizations/display/{{orga.id}}/{{groupe.id}}" data-
target="#users">{{ groupe }}</a></li>
      {% endfor %}
    </ul>
...

```

Resultat /Organizations/display/1/2

# Messagerie Administration

## Conservatoire National des Arts et Métiers

lecnam.net

### Groupes

- Personnels
- Auditeurs

### Auditeurs

|                 |                 |               |               |
|-----------------|-----------------|---------------|---------------|
| Levi BISHOP     | Cyrus ROSARIO   | Gil BRIGHT    | Jane HOLDEN   |
| Clare MACDONALD | Barrett HOLCOMB | Alana LAMBERT | Kenyon HINTON |
| Vera POWERS     | Brennan SHAW    |               |               |

### Application

- Ajouter du css pour repérer le groupe sélectionné (classe css "active")
- Ajouter un lien pour afficher tous les utilisateurs

## Messagerie Administration

### Conservatoire National des Arts et Métiers

lecnam.net

#### Groupes

- Personnels
- Auditeurs
- **Tous les utilisateurs →**

#### Tous les utilisateurs

|                |                 |                |                |
|----------------|-----------------|----------------|----------------|
| Wyatt HIGGINS  | Jane LEON       | Maggy WEBER    | Allistair LEON |
| Alvin HATFIELD | Kendall COLLIER | Steven NORMAN  | Joan HATFIELD  |
| Ivor LOGAN     | Hyacinth FINLEY | Madeson LARSEN | Maris MOSLEY   |

## - Optimisation de la navigation

Ouvrir la console du navigateur (Bouton droit de la souris puis inspecter ou ctrl+MAJ+c au clavier)

Activer l'onglet Réseau :

| Name   | Status | Type       | Initiator            | Size              | Time   | Waterfall |
|--|--------|------------|----------------------|-------------------|--------|-----------|
| 1  | 200    | document   | Other                | 3.6 KB            | 91 ms  |           |
| jquery-3.3.1.min.js                                  | 200    | script     | 1                    | (from memor...)   | 0 ms   |           |
| semantic.min.js                                      | 304    | script     | 1                    | (from memor...)   | 0 ms   |           |
| semantic.min.css                                     | 200    | stylesheet | 1                    | (from disk ca...) | 109 ms |           |
| css?family=Lato:400,700,400italic,700italic&subse... | 200    | stylesheet | 1                    | (from memor...)   | 0 ms   |           |
| S6uyw4BMUTPHjx4wXg.woff2                             | 200    | font       | 1                    | (from memor...)   | 0 ms   |           |
| S6u9w4BMUTPHh6UVSwiPGQ.woff2                         | 200    | font       | 1                    | (from memor...)   | 0 ms   |           |
| icons.woff2  | 304    | font       | 1                    | (from memor...)   | 0 ms   |           |
| in-page-script.js                                    | 200    | script     | content-script.js:83 | (from disk ca...) | 65 ms  |           |
| favicon.ico  | 200    | x-icon     | Other                | 30.5 KB           | 5 ms   |           |

10 requests | 34.1 KB transferred | Finish: 463 ms | DOMContentLoaded: 388 ms | Load: 454 ms

Afficher les pages :

- /Organizations/display/1

Puis en cliquant sur les liens des groupes :

- /Organizations/display/1/1
- /Organizations/display/1/2

Notez les temps de chargement : environ **450 ms** pour chaque page (10 requêtes)

Chaque URL recharge complètement la page, sa présentation (HTML + CSS) et ses données (organisation, Groupes, Users), alors que l'accès à l'URL /Organizations/display/1/1 permettant de charger les utilisateurs du groupe 1 de l'organisation 1 ne devrait charger que ces utilisateurs (le reste ayant déjà été affiché).

L'utilisation d'AJAX (acronyme d'asynchronous JavaScript and XML) va permettre d'éviter ce surplus de chargement, en effectuant des requêtes partielles, n'affichant que les informations nouvelles.

Introduction d'ajax sur les liens vers les utilisateurs des groupes dans le contrôleur :

...

```

public function display($idOrga,$idGroupe=null){
    if(URrequest::isAjax()){
        echo $this->users($idOrga,$idGroupe);
    }else{
        $orga=DA0::getOne("models\\Organization", $idOrga,true,true);
        $users=$this->users($idOrga,$idGroupe,$orga->getUsers());
        $this->jquery->getHref("a","#users");
    }
    $this->jquery->renderView("Organizations/display.html",["orga"=>$orga,"users"=>$users]);
}
}
...

```

Dans la vue :

- Ajout de l'attribut **data-target** sur les balises href
- Ajout de la variable **script\_foot** intégrant le script js généré (remarquer le filtre **raw** sur la variable script\_foot)

```

{% extends "base.html" %}
{% block body %}
<h2 class="ui header">
    {{orga.name}}
    <div class="sub header">{{orga.domain}}</div>
</h2>
<div class="ui two columns grid">
    <div class="four wide column">
        <h3 class="ui header">
            <i class="ui users icon"></i>
            <div class="content">Groupes</div>
        </h3>
        <ul>
            {% for groupe in orga.groupes %}
                <li><a href="Organizations/display/{{orga.id}}/{{groupe.id}}" data-
target="#users">{{ groupe }}</a></li>
            {% endfor %}
        </ul>
    </div>
    <div class="twelve wide column">
        <div id="users">
            {{users | raw}}
        </div>
    </div>
</div>
{{ script_foot | raw }}
{% endblock %}

```

Afficher à nouveau les pages :

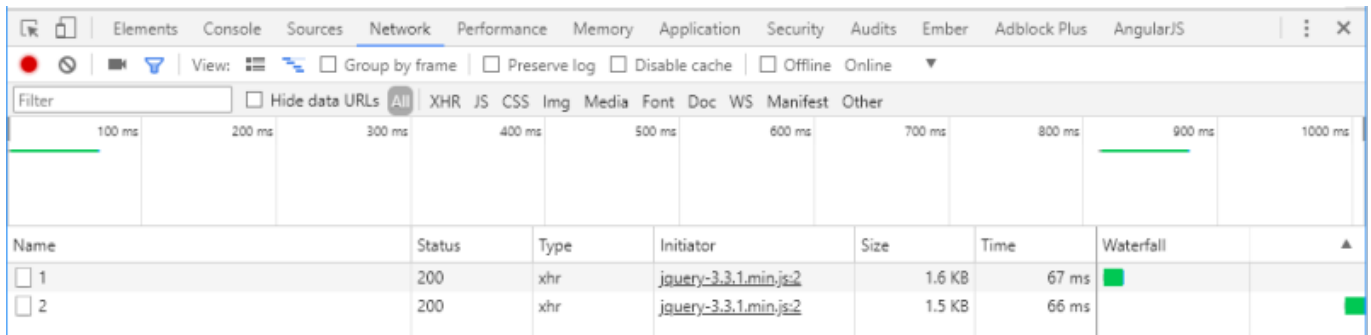
- /Organizations/display/1

Puis en cliquant sur les liens des groupes :

- /Organizations/display/1/1

- /Organizations/display/1/2

Notez les temps de chargement : pas de changements pour la première page mais environ **<fc #008000>60 ms</fc>** pour chaque page suivante (1 seule requête)



## - Sécurisation de la navigation directe

L'accès direct à l'url (dans la barre de navigation) peut provoquer des erreurs

- l'adresse **Organizations/display/{idOrga}** si idOrga ne correspond à aucune organisation
- l'adresse **Organizations/display/{idOrga}/{idGroupe}**
  - si idGroupe ne correspond à aucun groupe
  - si idGroupe correspond à un groupe d'une autre organisation

## Vue message

Créer une vue **message.html** permettant d'afficher un message semantic ui :

```
<div class="ui {{type}} icon message">
  <i class="{{icon}} icon"></i>
  <div class="content">
    <div class="header">
      {{ header | raw }}
    </div>
    {{ body | raw }}
  </div>
</div>
```

Créer une méthode dans la classe ControllerBase permettant d'afficher un message à partir du chargement de cette vue :

```
...
public function message($type,$header,$body,$icon="info"){
  //A implémenter
}
...
```

Modifier le template **base.html** pour qu'il affiche éventuellement un message :

```
{% block header %}
  <h1>Messagerie Administration</h1>
{% endblock %}

{% block message %}
  {{ message | raw }}
{% endblock %}

{% block body %}
{% endblock %}
```

Implémenter les contrôles permettant d'obtenir les résultats ci-dessous.

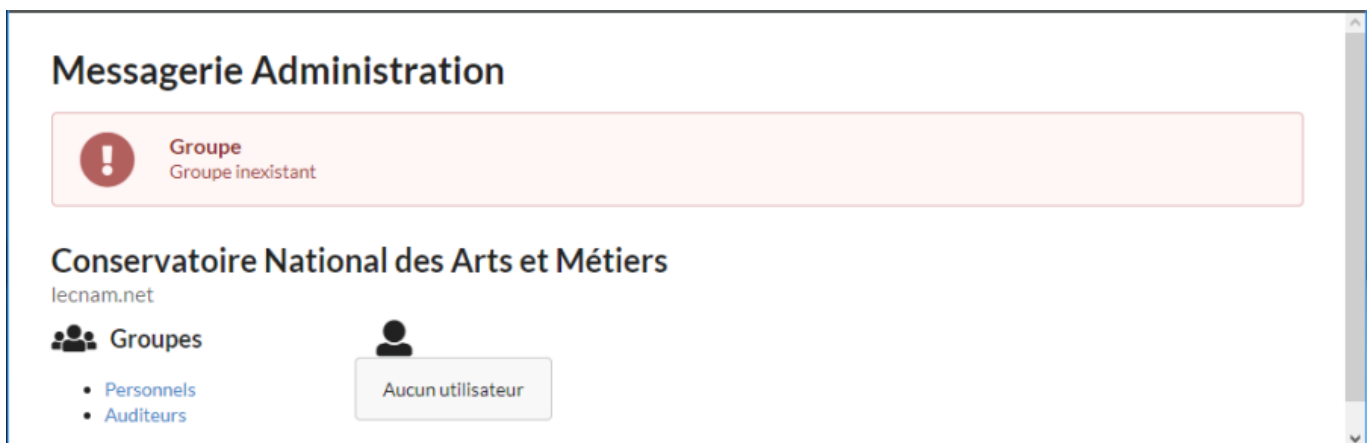
### Accès à une organisation inexistante

Exemple : **Organizations/display/144**



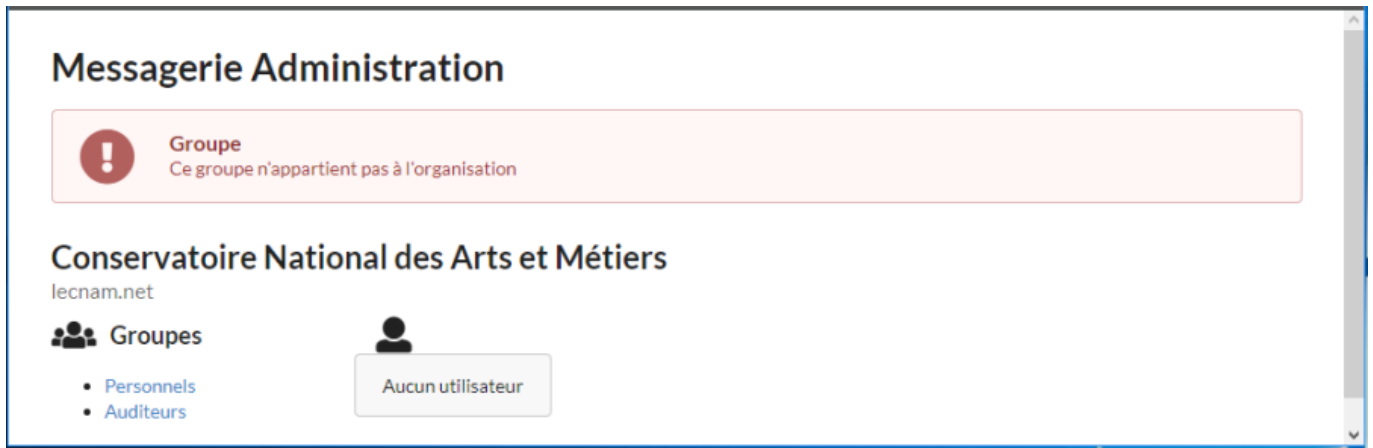
### Accès à un groupe inexistant

Exemple : **Organizations/display/1/144**



### Accès au groupe d'une autre organisation

Exemple : **Organizations/display/1/3**



## - Applications

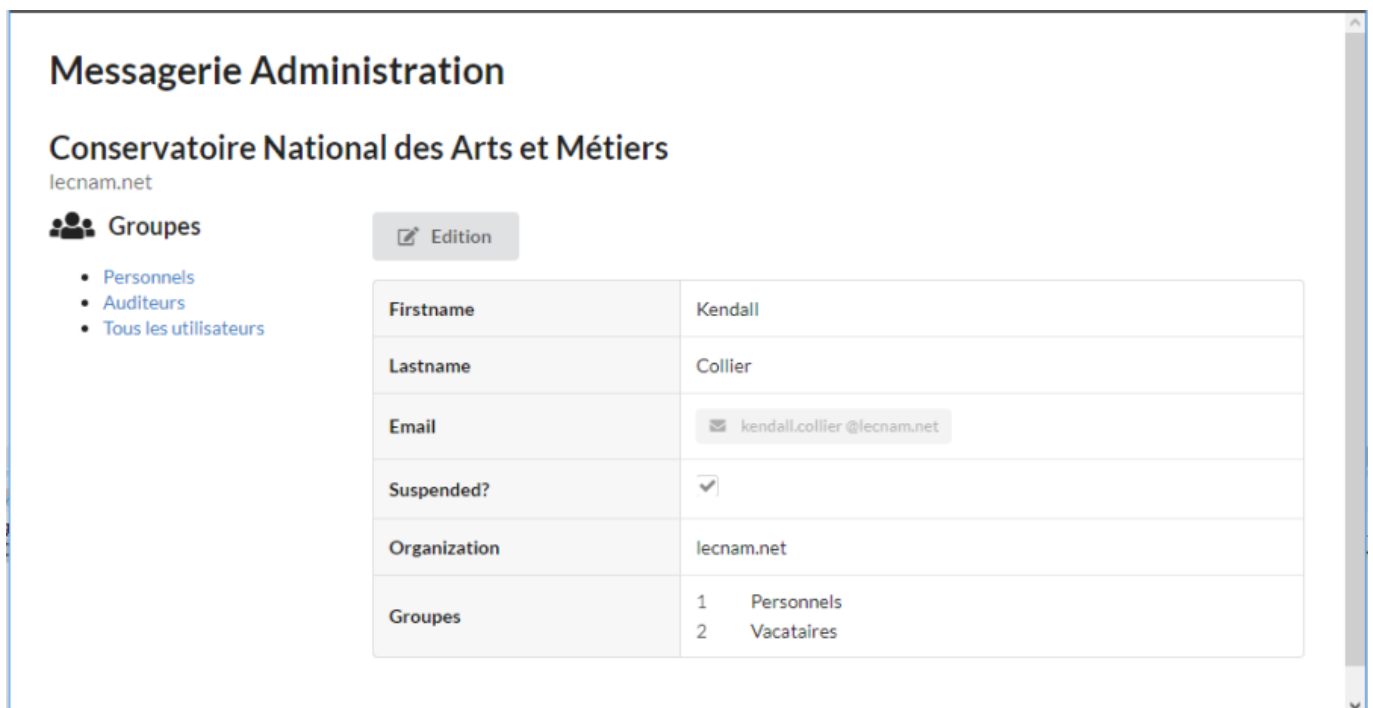
### - Navigation

Ajouter la navigation de l'adresse **/Organizations** vers **/Organizations/display/{idOrga}** sur le click d'une organisation (sans Ajax).

### - Affichage d'un utilisateur

Le click sur 1 utilisateur doit permettre d'accéder à l'url **/Users/display/{idUser}** affichant l'utilisateur et les groupes auxquels il appartient.

Le résultat doit s'afficher par une requête ajax dans la zone **users** existante :



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

Permanent link:  
<http://slamwiki2.kobject.net/php-rt/tp3?rev=1558577125>

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

