

TD n°7



- Suite Projet **boards**
- Application gestion de projets SCRUM

Commencer, à partir du dossier de votre projet par :

```
composer update
```

Objectifs

- Conception d'interfaces Web client riche
- Manipulations de données avec Doctrine
- Utilisation de composants

-- Eléments à implémenter

-- Route project/{idProject}/board

//TODO 2.1

Boards-EmberJS

Stories

link : project/:project_id/board

+ Add Story in project... Back to projects

Product backlog

Category	Description	Owner	Status
Admin	Partie administration	John Resig	
Analyse	Analyse fonctionnelle	John Resig	
Proj/Stories	Stories par projet	Yehuda Katz	
Tests	Aucune	Kris Selden	
MiseProduction	Mise en production du projet	Fabien Potencier	
SPE-TECH	Spécifications techniques du projet	John Resig	

Liste complète des stories du projet

Elle affiche la liste complète des stories du projet sélectionné.

on lui ajoute la partie tableau de bord, permettant d'afficher par steps (en colonnes), les user stories du projet sélectionné.

Boards-EmberJS

Stories

+ Add Story in project... Back to projects

Affiche la liste des stories non affectées à une step

Product backlog

Category	Description	Owner	Status
MiseProduction	Mise en production du projet	Fabien Potencier	
SPE-TECH	Spécifications techniques du projet	John Resig	

Dashboard

Boutons affichés si une story est sélectionnée

With selected story: Admin

Step	Story	Owner	Progress
Todo	Admin	John Resig	0%
In progress	Proj/Stories	Yehuda Katz	25%
Done	Analyse	John Resig	100%

Base de données

La base de données utilisée a été volontairement dénormalisée :

Par exemple, la table **story** possède un champ **step** mémorisant l'attribut title de la step associée, et non l'id de step, comme on aurait pu s'y attendre.

Il n'y a donc pas de relation effective de type père/fils dans la base entre **step** et **story**, ce qui implique :

- qu'il n'y a pas de membre step de type Step dans la classe **Story**
- pas plus qu'il n'y a de collection de **stories** dans chaque instance de **Step**

Pour charger les instances de Story dans chaque Step, il faut donc le faire programmatiquement :

```
protected function getStepsAndStories($project, StepRepository $stepRepo) {
    $steps=$stepRepo->findAll();
    $stories=$project->getStories()->toArray();
    foreach ($steps as $step){
        $step->stories=array_filter($stories, function($story) use($step){
            return ($story->getStep()==$step->getTitle());
        });
    }
    return $steps;
}
```

Comportement/conception de l'interface

Steps

Les Steps s'affichent de manière complète dans l'interface, même si elles ne contiennent aucune story. On pourra utiliser :

- le composant htmlGrid de phpMv-UI pour leur disposition.
- Chaque **step** est composée d'un **ui top attached segment** pour son titre, suivi d'un **ui attached segment** pour son contenu.

Stories

Lorsque la route **project/{idProject}** est active, toutes les stories sont affichées dans le product backlog. Lorsque la route **project/{idProject}/board** est active, toutes les stories non affectées à une step sont affichées dans le product backlog, les autres sont réparties entre leurs steps respectives.

- Le click sur une story présente dans le board la rend active et affiche la barre de 3 boutons (Open, MoveTo -dropDown des steps- et Remove).
- Les boutons **Open..** renvoient vers l'url **story/{idStory}**
- Il est possible de déplacer les stories entre les steps par un drap and drop

Si le composant HtmlGrid n'est pas utilisé, le tableau suivant pourra servir à élaborer la grille des steps :

'one'	'two'	'three'	'four'	
'five'	'six'	'seven'	'eight'	'nine'
'ten'	'eleven'	'twelve'	'thirteen'	'fourteen'
'fifteen'	'sixteen'			

Exemple d'utilisation du composant HtmlGrid de phpMv

```
$grid=$this->_semantic->htmlGrid("steps-grid");
foreach ($steps as $step){
    $col=$grid->addCol();
    $segTitle=new HtmlSegment("", '<i class="step forward icon"></i>&nbsp;'. $step->getTitle());
    $segTitle->addClass("secondary");
    $segContent=new HtmlSegment("step-".$step->getId());
    $segContent->addClass("drop-zone");
    $segContent->setProperty("data-ajax", $step->getTitle());
    $segTitle->setAttachment($segContent, "top");
    foreach ($step->stories as $story){
        $segContent->addContent($this->displayStory($tagRepo,$story));
    }
    $col->setContent([$segTitle,$segContent]);
}
```

Exemple de mise en oeuvre du drag and drop

Soit les éléments HTML suivants définis dans un template:

```
<div class="drag-item" draggable="true" data-ajax="donnée du transfert">
Contenu draggable
</div>
```

Ajoute la possibilité de déplacer tous les éléments ayant la classe css **drag-item**, et définit leur propriété data à **data-ajax**

```
$this->gui->setDraggable(".drag-item", ["attr"=>"data-ajax"]);
```

Définit une ou plusieurs zones potentielles cible du drag and drop :

```
<div class="drop-zone">
Zone de drop
</div>
```

Définit les éléments ayant pour classe **drop-zone** comme zone de drop :

```
$this->gui->asDropZone( ".drop-zone" );
```

Appel de code js après drop et récupération des données de l'élément déplacé et de la zone de drop :

```
$this->gui->asDropZone( ".drop-zone" , 'console.log("data du drag :
```

```
" +data);console.log("id du drop : "+$(event.target).attr("id"));');
```

Code de requête ajax

Pour obtenir le code d'une requête Ajax, mais sans l'exécuter :

Code d'une requête GET vers l'url **project/settings** dont le résultat sera affiché dans l'élément HTML d'id **settings-elm**

```
$js=$this->gui->getDeferred("project/settings","#settings-elm");
```

Exemple avec utilisation de javascript pour compléter l'url :

```
$js=$this->gui->getDeferred("project/settings","#settings-elm",["attr"=>'js:data+{/5}']);
```

-- Route Stories/edit/{idStory}

//TODO 2.2

Affiche la **story** correspondant au **code** passé dans l'url.



[Back to project stories](#)

⌚ Step : todo

B22 4 tâches bug Admin

👤 James Gosling

[Update tasks...](#)

En tant que créateur, je veux ajouter et gérer les réponses d'une question [methods].

get reponse/all

get reponse/:id

put reponse

post reponse/:id

Comportement de l'interface :

- Une task **done** est barrée

- Les cases à cocher permettent de faire passer une tâche de non réalisée (done=false) à réalisée (done=true) et inversement

Thanks

Si vous avez apprécié :

[phpmv-UI toolkit thanks](#)

[Semantic-UI thanks](#)

[symfony thanks](#)

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



Permanent link:
<http://slamwiki2.kobject.net/framework-web/symfony/td7?rev=1522248960>

Last update: **2019/08/31 14:43**