Helpers

PART 1 - JAVASCRIPT(*JS*) et AJAX (appels à distance)

<?php echo use_helper('Javascript') ?>

JAVASCRIPT ET AJAX HELPERS

JavaScript Helpers

link_to_function ($name, $function, $html_options=array())

```
<?php echo link_to_function("Click me!", "alert('foobar')") ?> // va générer:
<a href="#" onClick="alert('foobar');return none;">Click me!</a>
```

javascript_tag ($content)

```
<?php echo javascript_tag("document.getElementById('indicator').innerHTML=
 '<strong>Data processing complete</strong>';") ?>
```

update_element_function ($element_id, $options=array())

```
<?php echo javascript_tag( update_element_function( 'indicator', array(
"position"=>"after", "content" =>"<strong>Data processing complete</strong>" )))?>
```

Ajax Helpers

Une interaction AJAX est composée de 3 parties:
* **un déclencheur** (un lien, bouton ou n'importe quel contrôle que l'utilisateur peut manipuler pour lancer une action)
* **une action niveau serveur**
* **une zone ou sera affichée la réponse de l'action utilisateur**

Symfony fournit plusieurs helpers pour utiliser des interactions AJAX à l'intérieur de vos templates en mettant le **déclencheur** dans un lien, un bouton, un formulaire ou un timer. Ces helpers produisent du code HTML et non du code JavaScript.

link_to_remote ($name, $options=array(), $html_options=array())

```
<?php echo link_to_remote('Delete this post', array(
   'update' => 'indicator',  'url'  => 'post/delete?id='.$post->getId()  )) ?>
```

remote_function ($options=array())

```
<?php echo javascript_tag(remote_function(array(
   'update'  => 'myzone',
   'url'     => 'mymodule/myaction'
 ))) ?>
```

change une partie d'une page selon la réponse serveur.

form_remote_tag ($options=array(), $options_html=array())

```
<?php echo form_remote_tag(array(
   'update'  => 'item_list',  'url'  => '@item_add'  )) ?>
  <label for="item">Item:</label>
  <?php echo input_tag('item') ?>
  <?php echo submit_tag('Add') ?>
</form>
```

*ouvre un tag <form>, extactement comme le helper **form_tag()**.*

observe_field ($field_id, $options=array())

```
<?php echo form_tag('@item_add_regular') ?>
  <label for="item">Item:</label>
  <?php echo input_tag('item') ?>
  <?php echo submit_tag('Add') ?>
  <?php echo observe_field('item', array(
     'update'  => 'item_suggestion',
     'url'     => '@item_being_typed'
  )) ?>
</form>
```

le module/action indiqué dans @item_being_typed sera appelé chaque fois que la valeur d'un champs sera modifiée et cette action pourra récupérer le paramètre courant du champ. Si vous voulez passer un autre paramètre que la valeur du champ observé, vous pouvez les spécifiez en JavaScript avec le paramètre 'with'

periodically_call_remote ($options=array())

```
<?php echo periodically_call_remote(array(
   'frequency' => 60,
   'update'   => 'notification',
   'url'      => '@watch',
   'with'     => '"param=' + $('mycontent').value"
)) ?>
```

cet helper est une intéraction AJAX déclenchée toutes les x secondes. Elle n'est pas rattachée à un élément HTML particulier mais s'exécute de manière transparente en tâche de fond comme comportement global de la page

Another functions

```
submit_to_remote ($name, $value, $options=array())
evaluate_remote_response ()
observe_form ($form_id, $options=array())
visual_effect ($name, $element_id=false, $js_options=array())
sortable_element ($element_id, $options=array())
draggable_element ($element_id, $options=array())
drop_receiving_element ($element_id, $options=array())
javascript_cdata_section ($content)
input_auto_complete_tag ($name, $value, $url, $tag_options=array(),
                         $completion_options=array())
input_in_place_editor_tag ($name, $url, $editor_options=array())
```

PARAMETRES D'APPEL A DISTANCE

Tout les helpers AJAX peuvent prendre d'autres paramètres que ceux **'update'** et **'url'**:

position

Le paramètre de position peut être définit ainsi:

Valeur	Position
before	Avant l'élément
after	Après l'élément
top	Au dessus du contenu de l'élément
bottom	dessous le contenu de l'élément

conditions

confirm

'confirm'* => *'Etes vous sûrs?'

Un fenêtre popup *JS* montrant ***'Etes vous sûrs?'*** s'affichera quand l'utilisateur cliquera sur le contrôle et le ***module / action*** ne sera appelé que si le choix est confirmé par ***'Oui'***.

condition

***'condition'* => *"$('elementID') == true"*,**

L'appel à distance peut aussi être conditionné par un test côté client. (en JavaScript)

script execution

'script'* => *true

Si le code de retour de l'appel AJAX (code envoyé par le serveur, inséré dans l'élément à mettre à jour) contient du *JS*, ces scripts par défaut ne sont pas exécutés. Cette fonctionnalité ne peut être explicitement spécifiée qu'avec le paramètre script (=on)

callbacks

Callback	Evénement
before	Avant que la requête soit initialisée
after	Immédiatement après que la requête soit initialisée et avant le chargement
loading	Quand la réponse est chargée
loaded	Quand le navigateur à finit de charger la réponse
interactive	Quand l'utilisateur peut intéragir avec la réponse, même si son chargement n'est pas terminé.
success	Quand la 'XMLHttpRequest' est terminée et que le code de statut de réponse est compris dans l'intervale 2XX.
failure	Quand la 'XMLHttpRequest' est terminée et que le code de statut de réponse n'est pas compris dans l'intervale 2XX.
404	Quand la requête retourne un code d'erreur 404
complete	Quand la 'XMLHttpRequest' est terminée (déclenchée après son succès ou son échec, si présent)

```
e.g.: <?php echo link_to_remote('Delete this post', array(
          'update'    => 'indicator',
          'url'       => 'post/delete?id='.$post->getId(),
          'position'  => 'after',
          'confirm'   => 'Are you sure?',
          'script'    => true
          'loading'   => "Element.show('indicator')",
          'complete' => "Element.hide('indicator')"  )) ?>
```

NOTES

***** Les **actions** appelées à distance savent qu'elles le sont en temps que requête AJAX de ce fait n'incluent pas la barre de debug en développement. Aussi, le processus de décoration est aussi évité (leur template n'est pas inclus dans un layout par défaut). Si vous voulez qu'une vue AJAX soit décorée, vous devez le spécifier **"has_layout : true"** dans le fichier **view.yml** adéquat. Les actions appelées en AJAX retournent vrai à l'appel de la fonction suivante:

```
$isAjax = $this->isXmlHttpRequest();
```

* Les helpers AJAX ne fonctionneront pas si l'URL de l'action à distance n'appartient pas au même domaine que la page courante. Cette restriction existe pour des raisons de sécurité et s'appuie sur des limitations des navigateurs qui ne peuvent être outre-passées.

Les lirairies JavaScript Prototype et Script.aculo.us sont fournies par défaut avec symfony