Tp ajax

partir d'un projet rails et d'un scaffolding

```
rails new todojs --skip-bundle
cd todojs
# rajouter therubyracer dans le Gemfile, puis bundle install
rails g scaffold Task name:string
rake db:migrate
```

Scénario 1

Quand on clique sur le lien `New Task` dans la vue liste, le formulaire doit s'afficher dans la page courante.

Pour cela, dans `app/views/tasks/index.html.erb`, rajouter `:remote => true` dans la description du lien :

```
<%= link_to 'New Task', new_task_path, :remote => true %>
```

En faisant cela, le click sur le lien génère une requête HTTP avec le header `X-Requested-With: XMLHttpRequest`, qui indique au serveur que c'est une requête Ajax.

Modifiez maintenant l'action `new` du contrôleur pour prendre en compte la requête à ce format :

```ruby
# GET /tasks/new
# GET /tasks/new.json
def new
  @task = Task.new

  respond_to do |format|
    format.html # new.html.erb
    format.json { render json: @task }
    format.js # new.js.erb
  end
end
```

Traduction : à la réception d'une requête Ajax, on va retourner le javascript résultant du template `new.js.erb` au client.

Créer la vue `app/views/tasks/new.js.erb` :

```
$("form").remove();
$("a[href='<%= new_task_path %>']").after("<%= escape_javascript(render 'form') %>");
$("form#new_task").attr("data-remote", true);
$("a[href='<%= new_task_path %>']").fadeOut();
```

1. on enlève tous les éléments HTML de type form de la page.
2. on insère après le lien le rendu du formulaire (la vue partielle `_form.html.erb`)
3. on positionne l'attribut `data-remote` à true dans le formulaire que l'on vient de rajouter, pour que sa validation génère aussi une requête Ajax.
4. on cache le lien de création.

Scénario 2

- La validation du formulaire de création génère maintenant une requête Ajax `POST /tasks`, qui est traité par la méthode `create` du contrôleur. Modifier la méthode pour rendre la vue `create.js.erb`.
- créer la vue `create.js.erb` avec le contenu suivant :

```
$("body table tbody").append("<tr> <td><%= escape_javascript(@task.name) %></td>
  <td><%= escape_javascript(link_to 'Edit', edit_task_path(@task), :remote => true) %>
  </td> <td><%= escape_javascript(link_to 'Destroy', @task, confirm: 'Are you sure?', :remote => true, method: :delete) %></td> </tr>");
$("form#new_task").remove();
$("a[href='<%= new_task_path %>']").fadeIn();
```

1. on rajoute une ligne à la table HTML pour la nouvelle tâche créée
2. on enlève de la page le formulaire de création
3. on réaffiche le lien

Scénario 3

- Quand on clique sur le lien destroy d'une tâche, celle-ci doit disparaître de la liste.

À faire :

1. Modifier les liens "Destroy" pour qu'ils génèrent une requête Ajax
2. Modifier l'action `destroy` du contrôleur pour qu'elle rende la vue `destroy.js.erb`
3. Éditer la vue `destroy.js.erb` pour qu'elle enlève la ligne contenant la tâche détruite de l'affichage.

Scénario 4

Gérer l'édition d'une tâche en Ajax :

- clique sur le lien éditon doit ouvrir un formulaire dans la page
- la validation de ce formulaire doit mettre à jour l'affichage de la tâche