Synthaxe fetch

1. Get data fetch

> recuperez les enregistrements todo pour un utilisateur donnée

```
var url = `/pro/api/vtodo/${this.user_id}.json/`
console.log("url= " +  url )

fetch(url)
  .then(response => response.json())
  .then(data => this.todos = data )
  .catch(error => { console.log(error)})
  ///
  console.log("mounted! " +   this.todos.length)
},
```

2. Post data fetch

> recuperez les enregistrements todo pour un utilisateur donnée

```
save_todo(){
  // someting
  let url = `/pro/api/vtodo/create/`
  // chercher la cle de securité
  let CLE_CSRF = $("input[name='csrfmiddlewaretoken']" ).attr('value')
  CLE_CSRF = CLE_CSRF.trim(),

  var data_todo = {
    //'csrfmiddlewaretoken' : CLE_CSRF,
    'author' : this.user_id,
    'title' : this.newTodo.trim(),
     //"X-CSRF-Token": "Fetch",
  }
  //console.log("CLE_CSRF = " +  data_todo.csrfmiddlewaretoken);


  fetch(url, {
    method: "POST",
    credentials: "same-origin",
    headers : {
        //"X-CSRFToken": getCookie("csrftoken"),
        "X-CSRFToken": CLE_CSRF,
        "Accept": "application/json",
        "Content-Type": "application/json"
          },
          body: JSON.stringify(data_todo)
      })
      .then(response => response.json())
      .then(function(data) {
          this.message = "Votre todo est bien pris en compte ! "
          console.log("Data is ok", data);
          // refresh
      }).catch(function(ex) {
          console.log("parsing failed", ex);
    })
    ///
    console.log("add ok " +  url)
}
```

3. data DELETE

```
//------------
```

```
// delete todo
removeTodo(todo) {
  // rayer le todo
  this.todos.splice(this.todos.indexOf(todo), 1)
  // delete en vrai de la base
  var url = `/pro/api/vtodo/delete/${todo.id}`
  console.log("url= " +  url )

  fetch(url,
    {
     method: 'delete',
     credentials: "same-origin",
     headers : {
         "X-CSRFToken": getCookie("csrftoken"),
       }
    })
    .then(response => response.json())
    .then(data => this.todos = data )
    .catch(error => { console.log(error)})
    ///
},
```

4. PUT Update todo

```
//------------
// update todo
updateTodo(todo) {
  // rayer le todo
  this.todos.splice(this.todos.indexOf(todo), 1)
  // delete en vrai de la base
  var url = `/pro/api/vtodo/update/${todo.id}`
  console.log("url= " +  url )

  fetch(url,
    {
     method: 'put',
     credentials: "same-origin",
     headers : {
         "X-CSRFToken": getCookie("csrftoken"),
       }
    })
    .then(response => response.json())
    .then(data => this.todos = data )
    .catch(error => { console.log(error)})
    ///
},
```

Côté serveur

1. class serializer :

```
class TodoTodaySerializer(serializers.ModelSerializer):
    """
    ( 'id', 'title', 'author_id',)
    """
    # author = serializers.StringRelatedField(many=False, read_only=True)

    def create(self, validated_data):
        return Vtodo.objects.create(**validated_data)

    class Meta :
        model = Vtodo
        fields = '__all__'
```

2. class API VIEW

```
@method_decorator(login_required, 'dispatch')
### @api_view(['GET', 'PUT', 'DELETE' ])
class TodoTodayList(APIView):
    """
    List all todo
    Retrieve, update or delete a snippet instance.
    curl -X DELETE "http://127.0.0.1:8000/api/vtodo/(?P<pk>[-\d]+)/delete/"
    """
    def get(self, request, user_id, format=None):
        todos = Vtodo.objects.filter(author__id=user_id).all().order_by( '-id' )
        serializer = TodoTodaySerializer(todos, many=True)
        return Response(serializer.data)

    def post(self, request, format=None):
        # serializer
        serializer =  TodoTodaySerializer(data = request.data)

        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        # error !!
        messages.add_message(request, messages.INFO, "error api  TodoTodayList  {}".format(se

        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


    def put(self, request, pk, format=None):
        todo = self.get_object(pk)
        serializer = TodoTodaySerializer(todo, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    def delete(self, request, pk, format=None):
        messages.add_message(request, messages.INFO, "error api todo delete  {}".format(pk))
        # todo = self.get_object(pk)
        todo = Vtodo.objects.get(pk=pk)
        todo.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```