MVC – transmission de paramètres d'une requête dans la "query String"

On a d'abord envoyé le formulaire de façon basique

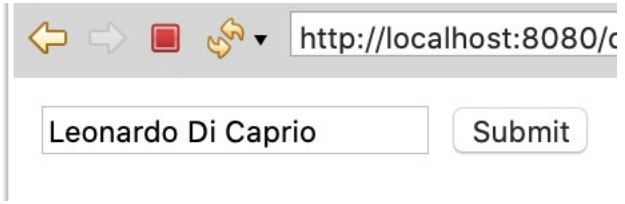
salutation-form.jsp

```
<form
      action="processForm"
      method="GET">

     <input type="text"
            name="userName"/>

     <input type="submit"/>

</form>
```

SalutationControleur.java

```
@Controller
...
     //réceptionner le formulaire
     @RequestMapping("/processForm")
     public String traiteSalutationForm() {
          return "salutation-message";
     }
...
```

!!



Message statique

Rappel : HttpServletRequest

- On a vu une première méthode héritée de JEE .
- Dans notre Controller , dans la méthode qui attend la soumission du formulaire, on a déclaré un objet HttpServletRequest en paramètre.
- De cet objet on a pu extraire la valeurs des input qu'on a rempli dans le formulaire.

Paramètres "natifs"

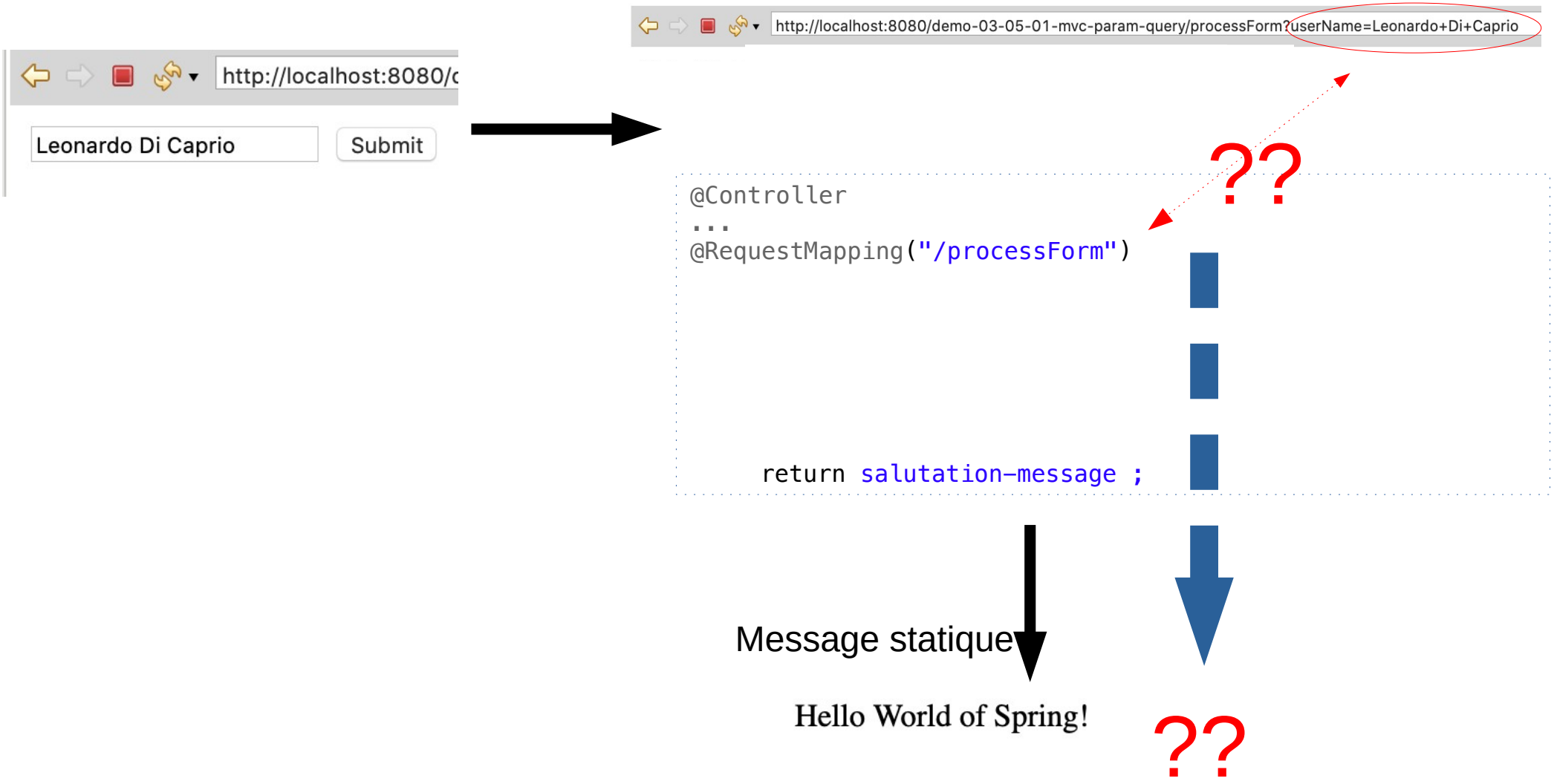
SalutationControleur.java

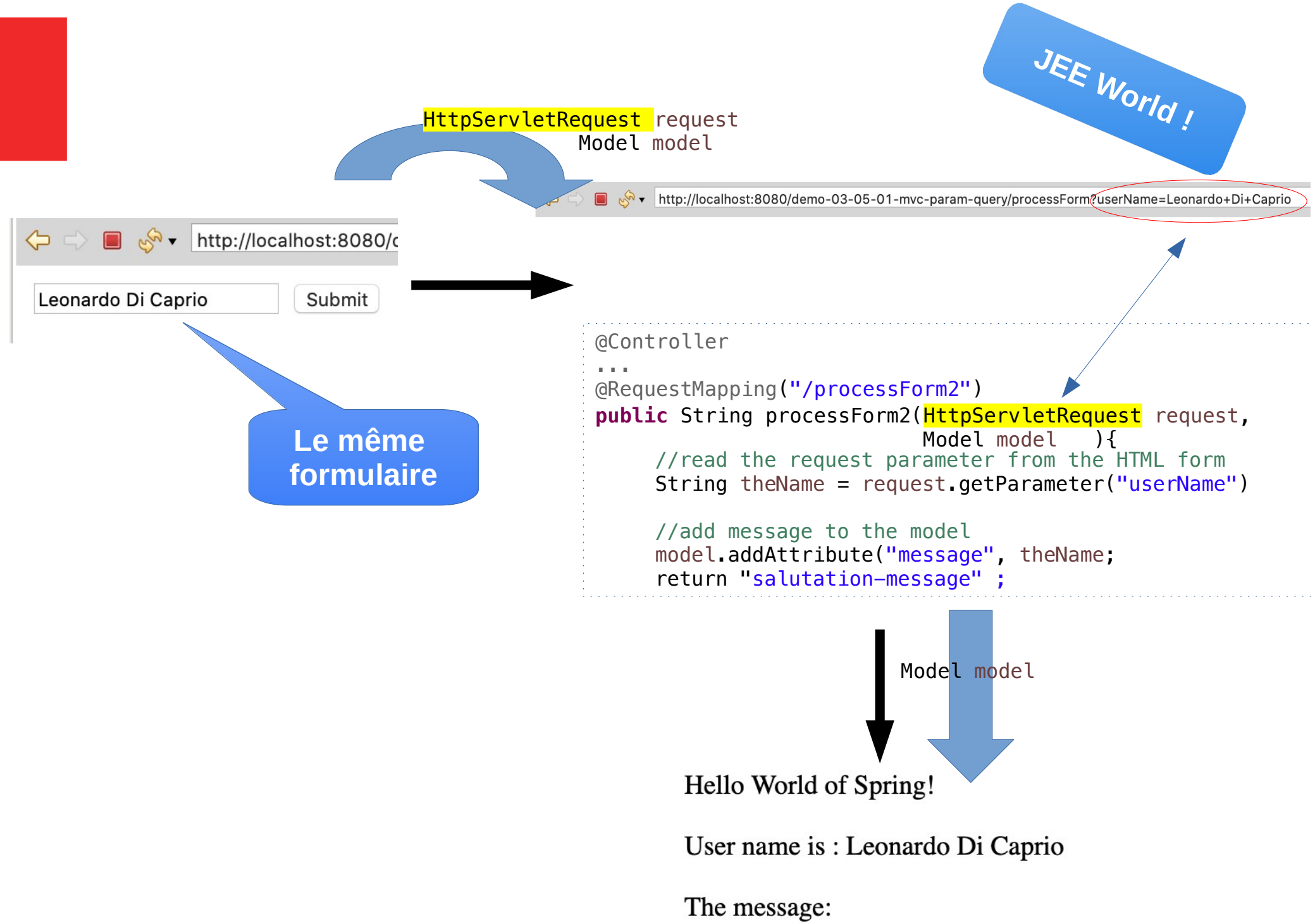
```
@Controller
     @RequestMapping("/processForm2")
     public String processForm2(HttpServletRequest request,
                                                    Model model    ){
          // read the request parameter from the HTML form
          String theName = request.getParameter("userName");

          // convert the data to all caps
          theName = theName.toUpperCase();
          // create the message
          String result = "Yo ! " + theName;
          // add message to the model
          model.addAttribute("message", result);

          return "salutation-message";
}
```

salutation-form.jsp

```
<form
      action="processForm2
      method="GET">

     <input type="text"
            name="userName"/>

     <input type="submit"/>

</form>
```

Le même formulaire

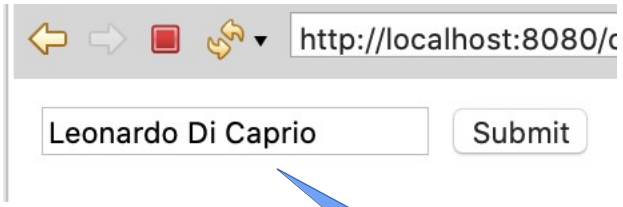

http://localhost:8080/
Leonardo Di Caprio
Submit
http://localhost:8080/demo-03-05-01-mvc-param-query/processForm?userName=Leonardo+Di+Caprio
??
@Controller
...
@RequestMapping("/processForm")
return salutation-message ;
Message statique
Hello World of Spring!
??


HttpServletRequest request
Model model
JEE World !
http://localhost:8080/demo-03-05-01-mvc-param-query/processForm?userName=Leonardo+Di+Caprio
http://localhost:8080/
Leonardo Di Caprio
Submit
Le même formulaire
@Controller
...
@RequestMapping("/processForm2")
public String processForm2(HttpServletRequest request,
Model model){
//read the request parameter from the HTML form
String theName = request.getParameter("userName")
//add message to the model
model.addAttribute("message", theName;
return "salutation-message" ;
Model model
Hello World of Spring!
User name is : Leonardo Di Caprio
The message:

Spring propose de gérer les paramètres présents dans la "query string".

- L' annotation @RequestParam permet de spécifier les paramètres directement en tant qu'arguments attendus par la méthode qui attend la soumission du formulaire

@RequestParam



Le même formulaire

salutation-form.jsp

```
<form
      action="processForm3
      method="GET">

     <input type="text"
            name="userName"/>

     <input type="submit"/>

</form>
```

```
@Controller
...
@RequestMapping("/processForm3")
public String processForm3(
          @RequestParam("userName") String theName ,
                                      Model model    ){

     // le userName est directement accessible


     //add message to the model
     model.addAttribute("message", theName;
     return salutation-message ;
```

Model model

Hello World of Spring!

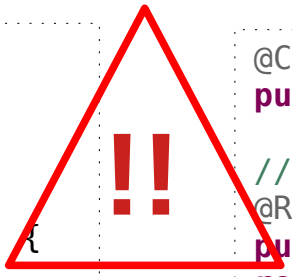
User name is : Leonardo Di Caprio

The message:

@RequestMapping sur la classe

- On peut organiser notre mapping par controller
- Cela permet d'avoir des méthodes qui ont le même mapping (même fonction) mais situées dans des contrôleurs différents

```java
@Controller
public class ClientControleur {

//afficher tous les clients
@RequestMapping("/listAll")
public String listeTousLesClients() {
return "client-list";
}
```




```java
@Controller
public class ProduitControleur {

//afficher tous les produits
@RequestMapping("/listAll")
public String listeTousLesProduits() {
return "client-list";
}
```

- On peut annoter les contrôleurs

```java
@Controller
@RequestMapping("/clients")
public class ClientControleur {

//afficher tous les clients
@RequestMapping("/listAll")
public String listeTousLesClients() {
return "client-list";
}
```

=> */clients/*listAll

```java
@Controller
@RequestMapping("/produits")
public class ProduitControleur {

//afficher tous les produits
@RequestMapping("/listAll")
public String listeTousLesProduits() {
return "client-list";
}
```

=> */produits/*listAll

TD : requestParam & mapping

- Codez une web app dans laquelle vous aurez un lien de navigation vers une autre page qui transmet un paramètre dans l'url appelée et l'affiche dans une nouvelle page.
- Vous ajouterez un second lien , qui transmet un autre paramètre , vers un nouveau contrôleur qui écoute la même url mais pour orienter l'utilisateur vers une autre page . Vous afficherez aussi les informations transmises
- Démarrez l'application et constatez l'erreur au démarrage
- Résolvez le conflit de mapping en annotant les classes de contrôleur, et mettez à jour les url appelées dans les liens de navigation
- Enfin vous gèrerez les informations transmises en utilisant @RequestParam, et vous gèrerez les informations à transmettre vial l'objet implicite Model

Ambiguous mapping

```java
@Controller
public class UserController {
	@GetMapping("/getMethod")
	public String  getInfos() {
		return "user-view";
	}
}
```

```jsp
<!DOCTYPE html>
<html>
	<body>
		Page   USER
		name=    ${param.name}
		The message:  ${message}
	</body>
</html>
```

main-menu.jsp

```jsp
<a href="getMethod?name=toto">
user get transmitter </a>


<a href="getMethod?product=abc">
product get transmitter </a>
```

```
... org.apache.catalina.core.StandardContext loadOnStartup
SEVERE: Le Servlet [dispatcher] dans l'application web [/demo-
03-05-02-mvc-param-query-td] a retourné une exception lors de
son chargement
java.lang.IllegalStateException: Ambiguous mapping. Cannot map
'userController' method
com.mvcdemo.controller.UserController#getInfos()
to {GET /getMethod}: There is already 'productController' bean
method
com.mvcdemo.controller.ProductController#getInfos() mapped.
```

```java
@Controller
public class ProductController {
	@GetMapping("/getMethod")
	public String  getInfos() {
		return "product-view";
	}
}
```

```jsp
<!DOCTYPE html>
<html>
	<body>
		Page   PRODUCT
		product=    ${param.product}
		The message:  ${message}
	</body>
</html>
```

Solution

```java
@Controller
@RequestMapping("users")
public class UserController {
	@GetMapping("/getMethod")
	public String  getInfos() {
		return "user-view";
	}
}
```

```jsp
<!DOCTYPE html>
<html>
	<body>
		Page   USER
		name=    ${param.name}
		The message:  ${message}
	</body>
</html>
```

main-menu.jsp

```jsp
<a href="users/getMethod?name=toto">
user get transmitter </a>


<a href="products/getMethod?product=abc">
product get transmitter </a>
```

Les paramètres de la Query-String sont disponibles dans l'objet implicite HttpServletRequest également dans la page jsp via la syntaxe ${param.xxx}
À éviter cependant :-(

```java
@Controller
@RequestMapping("products")
public class ProductController {
	@GetMapping("/getMethod")
	public String  getInfos() {
		return "product-view";
	}
}
```

```jsp
<!DOCTYPE html>
<html>
	<body>
		Page   PRODUCT
		product=    ${param.product}
		The message:  ${message}
	</body>
</html>
```

Argumentez votre méthode d'écoute avec @RequestParam et Model

- Je suis sûr que vous avez déjà tous compris.
- je vous laisse réaliser cette opération...