##### 

##### LEARN TO PROG :

JAVA SCRIPT :

# Chapitre 1

**Premiers pas en *JavaScript***

* 1. **Balise <script> et Hello world en *JavaScript***
     1. ***Hello Word* Avec document.write**

##### Une première manière d’insérer un script *JavaScript* dans un fichier *HTML* est de mettre le code *JavaScript* dans une balise <script></script>. On a alors accès au document dans le code *JavaScript* et on peut sortir du code *HTML* :

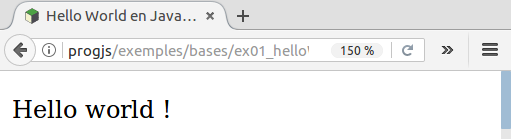


Figure 1.1 : Illustration du code source [1.1](#_bookmark6)

Code Source 1.1 : /bases-js/ex01-helloWorldTest.html (cf. Fig [1.1](#_bookmark5))

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**He l l o World en Ja v a s c r i p t**</ t i t l e>**

**</head>**

**<body>**

**<p>**

**<s cript>**

document . w r i te ( *” Hello world !”* ) ;

**</s cript>**

**</p>**

**</body>**

**</html>**

2

3

4

5

6

7

8

9

10

11

12

13

14

* + 1. ***Hello Word* Avec la Méthode alert de *Popup***

##### Une autre manière d’insérer un script *JavaScript* dans un fichier *HTML* est de mettre le code *JavaScript* dans un fichier .js séparé, qui est inclus dans le *HTML* au niveau du header par une balise <script src=''...''></script>.

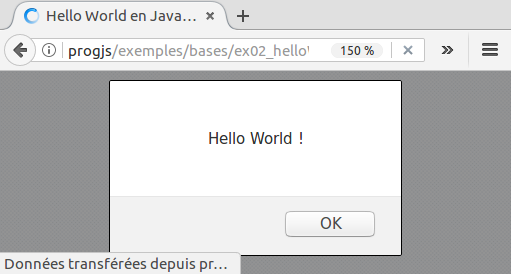


Figure 1.2 : Illustration du code source [1.2](#_bookmark9)

Code Source 1.2 : /bases-js/ex02-helloWorldTest.html (cf. Fig [1.2](#_bookmark8))

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**H e l l o World en Ja v a s c r i p t**</ t i t l e>**

**<p>**

**</head>**

**<s cript src=***” . /ex 02 −hello World . j s ”***></s cript>**

**<**

**/p>**

**<body>**

**</body>**

**</html>**

2

3

4

5

6

7

8

9

10

11

12

Code Source 1.3 : /bases-js/ex02-helloWorld.js

1

a l e r t ( *” Hello World !”* ) ;

Dans ce dernier cas, on ne peut pas générer de code directement dans le document *HTML* (avec document.write) dans le fichier *JavaScript*, mais il y a d’autres avantages (factorisation et mise en cache du code *JavaScript* partagé entre plusieurs pages *HTML* par exemple).

La fonction alert peut par exemple utilisé pour du débuggage si l’on ne dispose pas d’un outil de *debug* intégré dans notre *IDE*.

* + 1. **Document *Template* et Modification Dynamique de la Vue**

Dans une application *HTML/JavaScript* côté client, on organise souvent le code avec, d’une part, un squelette de document *HTML*, appelé *template*, et d’autre part des fichiers *JavaScript*, inclus le plus tard possible dans la page *HTML*, qui modifie *a posteriori* le document pour générer certaines parties dynamiquement.

Code Source 1.4 : /bases-js/ex02bis-helloWorldTemplateTest.html

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**He l l o World en Ja v a s c r i p t**</ t i t l e>**

**</head>**

**<body>**

2

3

4

5

6

7

8

**<***!−− Cré ation d ’ une* **page** *” Template ” , s q u e l e t t e de notre* **page** *−−***>**

**<***!−− Paragraphe qui va contenir l e code −−***>**

**<p id=***” paragraphe Resultat ”***></p>**

**<s cript src=***” . / ex 02 bis−hello World Template . j s ”***></s cript>**

**<**

**<***!−− Inclusion du f i c h i e r Java Script pour l e code dynamique −−***>**

**</html>**

**/body>**

9

10

11

12

13

14

Code Source 1.5 : /bases-js/ex02bis-helloWorldTemplate.js

1

*// Géné ration dynamique de code HTML :*

**var** codeHTML = *” Hello world !”* ;

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

2

3

4

* 1. **Types, Variables et Portée**

Le *JavaScript* est un langage faiblement typé, car on n’indique pas le type des variables lors de la déclaration. Lors le la déclaration des variables, le type est fixé implicitement par le type de la donnée affectée à la variable.

La déclaration de la variable peut contenir ou non le mot clef var. Un variable déclarée avec le mot clef var (on parle de déclaration *explicite*) est locale à la fonction où la variable est déclarée. Une variable déclarée sans le mot clef var (on parle de déclaration *implicite*) est globale.

Il n’y a pas, contrairement au *C++* ou *Java*, de visibilité locale à un bloc. Un variable déclarée n’importe où dans une fonction est visible dans toute la fonction au moins. Pour cette raison, on déclarera systématiquement les variables locales à la fonction au début du corps de la fonction, contrairement aux bonnes pratiques dans d’autres langages où on déclare la variable au plus près de son point de première utilisation.

Dans les programmes assez gros structurés en modules ou *packages*, on peut créer en *JavaS- cript* l’équivalent d’un *namespace* par un patron de conception consistant à mettre le code de l’ensemble d’un module dans le corps de définition d’une fonction ou dans un littéral définissant

un objet (voir plus loin pour la notion d’objet).

* 1. **Fonctions**

Les fonctions en *JavaScript* sont déclarées par le mot clef function. c’est un type de données comme un autre, et une fonction peut ainsi être affectée à une variable. Voici un exemple de fonction qui calcule le prix *TTC* d’un produit à partir de son prix hors taxes. Comme les paramètres des fonctions ne sont pas typés, on peut vérifier le type des paramètres dans la fonction et éventuellement renvoyer une exception si le type du paramètre effectif n’est pas le bon.

Code Source 1.6 : /bases-js/ex03-functionTest.html

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

2

3

4

5

**<t i t l e>**Fonctions**</ t i t l e>**

**</head>**

**<body>**

**<***!−− Cré ation d ’ une* **page** *” Template ” , s q u e l e t t e de notre* **page** *−−***>**

**<***!−− Paragraphe qui va contenir l e code −−***>**

**<p id=***” paragraphe Resultat ”***></p>**

**<s cript src=***” . / ex 03 −function . j s ”***></s cript>**

**<**

**<***!−− Inclusion du f i c h i e r Java Script pour l e code dynamique −−***>**

**</html>**

**/body>**

6

7

8

9

10

11

12

13

14

Code Source 1.7 : /bases-js/ex03-function.js

1. */\*\* @ description Calcule l e prix TTC d ’ un produit*
2. *\* @ function calcul Prix TTC*
3. *\* @param {number} prixHT Le prix hors taxe du produit*

−

1. *\* @param {number} tauxTVA Le taux de TVA à appliquer*

−

1. *\* @throws Si l e s paramètres prixHT et tauxTVA ne sont pas des nombres*

6 *\*/*

1. **var** calcul Prix TTC = **function** ( prixHT , tauxTVA) {
2. **i f** ( ! ( **typeof** prixHT == *” number”* ) | | ! ( **typeof** tauxTVA == *” number”* ) ) {
3. **throw new** Error ( *” Function calcul Prix TTC appel ée avec paramètre in correct . ”* ) 10 }

11 **return** prixHT \*(1 .0 +tauxTVA / 1 0 0 . 0 ) ; 12 } ;

13

14

1. *// Géné ration dynamique de code HTML :*
2. **var** codeHTML = *””* ;
3. *// Appel correct de l a fonction :*
4. **try** {
5. codeHTML += *” Prix TTC : ”* + calcul Prix TTC ( 1 8 0 . 0 , 1 9 . 6 ) ;
6. } **catch** ( e r r ) {
7. a l e r t ( e r r ) ; 22 }
8. *// Appel in correct de l a fonction dé c l enchant une exception :*
9. try {
10. codeHTML += *” Prix TTC : ”* + calcul Prix TTC ( *” coucou”* , 1 9 . 6 ) ;
11. } **catch** ( e r r ) {
12. a l e r t ( e r r ) ; 28 }
13. *// Mise à jour de l a vue*
14. document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

##### Notons que l’on peut aussi déclarer une fonction un peu comme en *PHP* de la manière suivante :

Code Source 1.8 : Ne jamais faire ça !

1

**function** myFunction (myParam) {

**return** (myParam **<** 0 ) ;

}

2

3

mais la fonction est alors globale (son nom existe dans tout le programme).

La **bonne pratique** consiste à déclarer les éléments d’un programme de sorte qu’ils aient la portée la plus locale possible, donc à déclarer la fonction avec le mot clé var comme dans le premier exemple de fonction ci-dessus.

* 1. **Objets**

Un *objet JavaScript* rassemble plusieurs propriétés, qui peuvent être des données, d’autres objets, ou encore des fonctions, alors appelées *méthodes*. Un objet n’est ni tout à fait une structure comme en *C*, ni tout à fait une classe comme dans un *langage objet classique*. Par

exemple, un objet *JavaScript* n’a pas de visibilité (privée, public) pour ses propriétés. Par ailleurs, le principal mécanisme d’héritage en *JavaScript* se fait par la notion de *prototype* et est très différent de l’héritage dans les langages objet classiques. Là encore, on peut mimer une

notion de visibilité via des patrons de conception.

Les noms de propriétés peuvent être

Soit une chaîne de caractère (comme ”nom de propriété !”) quelconque (respecter les doubles *quotes* dans un tel cas).

*•*

Soit des noms légaux (commençant par une lettre suivi par une suite de lettres, chiffres, et *underscores* (caractère \_) auquel cas les doubles *quotes* sont optionnelles pour désigner le nom.

*•*

* + 1. **Création d’un objet au moyen d’un littéral**

On peut créer un nouvel objet par un littéral, en définissant ses propriétés des accolades . On met alors chaque nom de propriété suivi d’un : suivi de la valeur de la propriété. Les propriétés ainsi construites sont séparées par des virgules.

*{}*

Code Source 1.9 : /bases-js/ex04-objectLitteral.js

1

*/\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ” \*/*

**var** produit = {

*” denomination ”* : *” Notebook sous Ubuntu 4 cores 2 . 0GB”* ,

*” prixHT”* : 1 8 0 . 0 ,

*”tauxTVA”* : 19 . 6

} ;

*/\*\**

* *@ description Calcule l e prix TTC d ’ un produit*
* *@ function calcul Prix TTC*

*\* @param { Object } prod − Le produit à t r a i t e r*

*\* @param {number} prod . prixHT − Le prix hors taxe du produit*

*\* @throws Si l e s propri é t é s du paramètre prod ne sont pas des nombres*

*\*/*

**var** calcul Prix TTC = **function** ( prod ) {

*// Test d ’ ex i s t en ce des propri é t é s de l ’ ob j e t :*

**i f** ( *” prixHT”* **in** prod && *”tauxTVA”* **in** prod ) {

**return** prod . prixHT \*(1 .0 + prod . tauxTVA / 1 0 0 . 0 ) ;

} **else** {

*// Rejet d ’ une exception personnalis ée :*

*// On r e j e t t e un ob j e t avec une prop . ”name” et une prop . ” message ” .*

**throw** {

*\* @param {number} prod . tauxTVA − Le taux de TVA à appliquer*

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

} ;

name : *”Bad Parameter”* ,

message : *” Mauvais type de paramètre pour l a fonction calcul Prix TTC”*

} ;

}

*// Essai d ’ appel de l a fonction*

**try** {

**var** codeHTML = *” Prix TTC du produit \””*+produit . denomination

+*”\” : ”*+calcul Prix TTC ( produit ) ;

} **catch** ( e ) { *// a f f i ch ag e de l ’ exception personnalis ée .*

a l e r t ( *”Une erreur \””* + e . name + *”\” s ’ es t produite :\n”* + e . message ) ;

}

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

24

25

26

27

28

29

30

31

32

33

34

35

36

37

* + 1. **Méthodes**

Un objet peut contenir des propriétés qui sont de type function. On parle alors de *méthode* de l’objet. Dans une méthode, on accède aux propriétés des l’objet grâce à l’identificateur this, désignant l’objet auquel appartien la méthode.

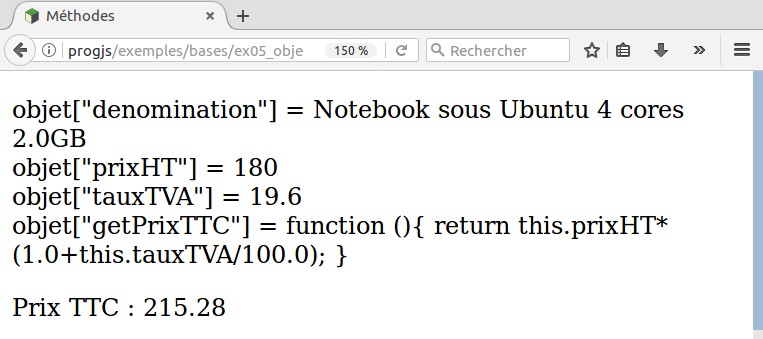


Figure 1.3 : Illustration du code source [1.10](#_bookmark17)

Code Source 1.10 : /bases-js/ex05-objectMethod.js (cf. Fig [1.3](#_bookmark16))

1

*/\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ” \*/*

**var** produit = {

*” denomination ”* : *” Notebook sous Ubuntu 4 cores 2 . 0GB”* ,

*” prixHT”* : 1 8 0 . 0 ,

*”tauxTVA”* : 1 9 . 6 ,

*/\*\**

* *@ description Calcule l e prix TTC d ’ un produit*
* *@method calcul Prix TTC*
* *@return Le prix TTC du produit*

*\*/*

getPrixTTC : **function** ( ) {

**return this** . prixHT \*(1 .0 + **this** . tauxTVA / 1 0 0 . 0 ) ;

}

} ;

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

*// Fonction dans l e contexte g l ob a l :*

*/\*\* @ description Gé nère l e code HTML pour a f f i c h e r un ob j e t quelconque*

*\* @ function get Html Objet*

*\* @return Le code HTML pour a f f i c h e r l ’ ob j e t*

*\*/*

**var** getHtml Objet = **function** ( o b je t ) {

**var** chaine = *””* ;

*// Parcours de t ou t es l e s propri é t é s de l ’ ob j e t ( s t y l e ” foreach ”) :*

**for** ( **var** nom **in** o b je t ) {

chaine += *” ob j e t [ \” ”*+nom+*” \”] = ”*

+ o b je t [ nom ] *// Appel de l a mé thode t o S t r in g par dé f aut*

+*”***<br** */***>***”* ;

}

**return** chaine ;

} ;

*// appel d ’ une fonction dé f i n i e dans l e contexte g l ob a l :*

**var** codeHTML = *”***<p>***”* + getHtml Objet ( produit ) + *”***</p>***”* ;

*// appel d ’ une mé thode :*

codeHTML += *”***<p>***P rix TTC : ”* + produit . getPrixTTC ( ) + *”***</p>***”* ;

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

*\* @param { Object } ob j e t − L ’ ob j e t à t r a i t e r*

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

Une méthode d’objet *JavaScript* n’est pas tout à fait comme une méthode d’un langage à objet classique, car la méthode *JavaScript* existe en autant d’exemplaires qu’il y a d’instance des objets. Nous verrons plus loin la notion de *prototype*, qui permet de crée des méthodes qui existent en un seul exemplaire pour toute une classe d’objets ayant les mêmes propriétés.

* + 1. **Objets Imbriqués (Composites et Composés)**

Il est possible, dans un littéral d’objet, de créer une propriété qui est elle-même un objet. On peut parler de *composite* pour l’objet qui contient un autre objet, qui est alors un *composé*.

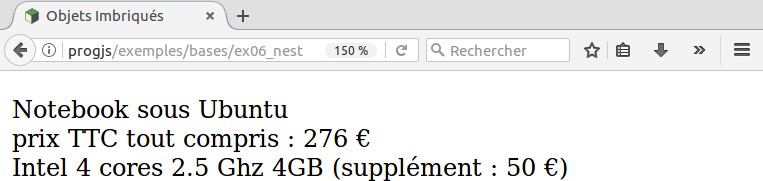


Figure 1.4 : Illustration du code source [1.11](#_bookmark20)

Code Source 1.11 : /bases-js/ex06-nestedObjects.js (cf. Fig [1.4](#_bookmark19))

1

*/\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ” \*/*

**var** produit = {

denomination : *” Notebook sous Ubuntu”* , prixHT\_base : 1 8 0 . 0 ,

tauxTVA : 2 0 . 0 ,

*/\*\* @ description Objet ” nich é” dans un sur−ob j e t ( Options du produit ) \*/*

o p t i o n s : {

p r o c e s s o r : *” I n t e l 4 cores 2 . 5 Ghz”* ,

2

3

4

5

6

7

8

|  |  |
| --- | --- |
| 9 | memory : *” 4GB”* , |
| 10 | *” prix suppl é mentaire HT”* : 5 0 . 0 , |
| 11 | */\*\* @ description Gé nère l e code HTML des options* |
| 12 | *\* @method getHTML* |
| 13 | *\*/* |
| 14  15  16  17 | getHtml : **function** () {  **return this** . p r o c e s s o r + *” ”* + **this** . memory +  *” ( suppl ément : ”* + **this** [ *” prix suppl é mentaire HT”* ] + *” &euro ;)”* ;  } |
| 18 | } , |
| 19 | */\*\* @ description Gé nère l e code HTML complet du produit* |
| 20 | *\* @method getHTML* |
| 21 | *\*/* |
| 22 | getHtml : **function** () { |
| 23 | **return this** . denomination + |
| 24 | *”***<br** */***>***p r i x TTC t out compris : ”* |
| 25 | + ( **this** . prixHT\_base + ( **this** . o p t i o n s [ *” prix suppl é mentaire HT”* ] | | 0 . 0 ) |
|  | ) |
| 26 | \*(1 .0 + **this** . tauxTVA / 1 0 0 . 0 ) |
| 27 | + *” &euro ;***<br** */***>***”* + **this** . o p t i o n s . getHtml ( ) + *”***<br** */***>***”* ; |
| 28 | } |
| 29 | } ; |
| 30 |  |
| 31 | *// appel d ’ une mé thode :* |
| 32 | **var** codeHTML = *”***<p>***”* + produit . getHtml ( ) + *”***</p>***”* ; |
| 33 | *// Mise à jour de l a vue* |
| 34 | document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ; |

On peut aussi choisir de définir l’objet contenant les options dans une méthode getOptions

du composite, qui retourne un objet. La méthode getOptions joue alors un rôle de “fabrique”.

Code Source 1.12 : /bases-js/ex06-objectReturnedInMethod.js

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | */\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ”* | *\*/* |  |
| 2 | **var** produit = { |  |
| 3 | denomination : *” Notebook sous Ubuntu”* , |  |
| 4 | prixHT\_base : 1 8 0 . 0 , |  |
| 5 | tauxTVA : 2 0 . 0 , |  |
| 6 | */\*\* @ description Construction d ’ un ob j e t dans une mé thode* |  |
| 7 | *\*/* |  |
| 8 | get Options : **function** ( ) { |  |
| 9 | *// Cré ation de l ’ ob j e t option dans l a fonction (” f ab r i q u e ”)* |  |
| 10 | **return** { |  |
| 11 | p r o c e s s o r : *” I n t e l 4 cores 2 . 5 Ghz”* , |  |
| 12 | memory : *” 4GB”* , |  |
| 13 | *” prix suppl é mentaire HT”* : 5 0 . 0 , |  |
| 14 | */\*\* @ description Gé nère l e code HTML des options* |  |
| 15  16 | *\* @method getHTML* |  |
| 17  18 | getHtml : **function** () { |  |
| 19 | *” ( suppl ément : ”* + **this** [ *” prix suppl é mentaire HT”* ] | + *”* | *&euro ;)”* ; |
| 20 | } |  |  |
| 21 | } ; |  |  |
| 22 | } , |  |  |
| 23 | */\*\* @ description Gé nère l e code HTML complet du produit* |  |  |
| 24 | *\* @method getHTML* |  |  |

\*/

**return this** . p r o c e s s o r + *” ”* + **this** . memory +

25

} ;

*\*/*

getHtml : **function** () {

**return this** . denomination +

*”***<br** */***>***p r i x TTC t out compris : ”*

+ ( **this** . prixHT\_base

+ ( **this** . get Options ( ) [ *” prix suppl é mentaire HT”* ] | | 0 . 0 ) )

\*(1 .0 + **this** . tauxTVA / 1 0 0 . 0 )

+ *” &euro ;***<br** */***>***”* + **this** . get Options ( ) . getHtml ( ) + *”***<br** */***>***”* ;

}

*// appel d ’ une mé thode :*

**var** codeHTML = *”***<p>***”* + produit . getHtml ( ) + *”***</p>***”* ;

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

26

27

28

29

30

31

32

33

34

35

36

37

38

39

* + 1. **Constructeur d’Object et mot réservé new**

On peut créer un objet via le constructeur Object. Voici un exemple où l’on crée un objet qui représente un produit. On crée ensuite une fonction qui calcule le prix *TTC* de ce produit après avoir testé l’existence d’attributs.

Code Source 1.13 : /bases-js/ex07-objectNew.js

1. */\*\* @ description In s t an t i a t i on d ’ un Objet ” produit ” \*/*
2. **var** produit = **new** Object ( ) ;
3. *// Ajout dynamique de propri é t é s*
4. produit . denomination = *” Notebook sous Ubuntu 4 cores 2 . 0GB”* ;
5. produit . prixHT = 1 8 0 . 0 ;
6. produit . tauxTVA = 2 0 . 0 ; 7

8 */\*\**

1. *\* @ description Calcule l e prix TTC d ’ un produit*
2. *\* @ function calcul Prix TTC*
3. *\* @param { Object } prod Le produit à t r a i t e r*

−

1. *\* @param {number} prod . prixHT Le prix hors taxe du produit*

−

1. *\* @param {number} prod . tauxTVA Le taux de TVA à appliquer*

−

1. *\* @throws Si l e s propri é t é s du paramètre prod ne sont pas des nombres*

15 *\*/*

1. **var** calcul Prix TTC = **function** ( prod ) {
2. **i f** ( *” prixHT”* **in** prod && *”tauxTVA”* **in** prod ) {
3. **return** prod . prixHT \*(1 .0 + prod . tauxTVA / 1 0 0 . 0 ) ;
4. } **else** {
5. **throw new** Error ( *” Mauvais type de paramètre pour l a fonction calcul Prix TTC”* ) ; 21 }

22 }

1. *// Géné ration de code HTML*
2. **var** codeHTML = *” Prix TTC du produit \””*+produit . denomination+*”\” : ”*+ calcul Prix TTC ( produit ) ;
3. *// Mise à jour de l a vue*
4. document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

##### Dans la mesure du possible, il est préférable de définir les objets *JavaScript* par des littéraux car ça peut être plus efficace que la construction dynamique avec le constructeur Object.

* 1. **Tableaux (le type Array)**
     1. **Notion d’Array et construction**

Dans les langages classique, un tableau est une séquence d’éléments, contigus en mémoire, avec un accès aux éléments par un indice entier. En *JavaScript*, les tableaux sont des objets dont les propriétés sont automatiquement nommées avec les chaînes '0', '1', '2'. Ces tableaux possèdent certains avantages des objets, comme par exemple la possibilité d’avoir des éléments de types différents, mais sont significativement plus lents que les tableaux classiques.

Un tableau peut être créé par un littéral, entre crochets [ ].

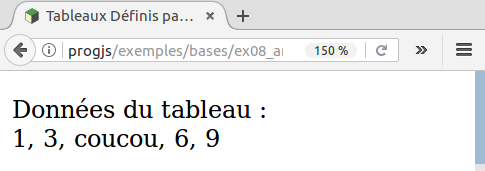


Figure 1.5 : Illustration du code source [1.14](#_bookmark25)

Code Source 1.14 : /bases-js/ex08-arrayLitterals.js (cf. Fig [1.5](#_bookmark24))

1

*/\*\* @ description Dé c l arat i on d ’ un Array sous forme de l i t t é ral \*/*

**var** tab = [ 1 , 3 , *” coucou”* , 6 ] ; tab [ 4 ]= 9 ; *// Ajout d ’ un é l ément*

*// Géné ration dynamique de code HTML :*

**var** codeHTML = *”Donné es du t ab l eau :***<br** */***>***”* ;

*// Parcours du t ab l eau avec un indice numé r i que*

**for** ( **var** i =0 ; i**<**tab . l e n g th ; i ++){

**i f** ( i **>** 0 ) { codeHTML += *” , ”*

}

codeHTML += tab [ i ] ;

}

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

2

3

4

5

6

7

8

9

10

11

12

13

14

Un tableau peut aussi être créé par le constructeur d’*Array*. Celui-ci peut prendre en argu- ment soit le nombre de cases du tableau à allouer, soit les éléments du tableau initialisés lors de la création du tableau. On peut toujours ajouter des éléments au tableau par une simple affectation de ces éléments et la mémoire évolue automatiquement.

Code Source 1.15 : /bases-js/ex09-arraysNew.js

1

*/\*\* @ description Dé c l arat i on d ’ un Array sous forme de l i t t é ral \*/*

**var** tab = **new** Array ( 1 , 3 , *” coucou”* , 6 ) ; tab [ 4 ]= 9 ; *// Ajout d ’ un é l ément*

*// Géné ration dynamique de code HTML :*

**var** codeHTML = *”Donné es du t ab l eau :***<br** */***>***”* ;

*// Parcours du t ab l eau avec un indice numé r i que*

**for** ( **var** i =0 ; i**<**tab . l e n g th ; i ++){

**i f** ( i **>** 0 ) { codeHTML += *” , ”*

}

2

3

4

5

6

7

8

9

10

11

}

codeHTML += tab [ i ] ;

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

*// Mise à jour de l a vue*

12

13

14

De même que pour les objets, il est préférable de définir les tableaux *JavaScript* par des littéraux car ça peut être plus efficace que la construction dynamique avec le constructeur Array.

* + 1. **Quelques méthodes prédéfinies sur le type Array**

Considérons un tableau array obtenu par invocation du contructeur d’Array :

Code Source 1.16 :

1

**var** array **= new** Array ( . . . ) ;

On peut manipuler le tableau avec des méthodes de collection de conception classique : Suppression du dernier élément (l’élément supprimé est retourné par la méthode) :

*•*

function array.pop();

Suppression du premier élément (l’élément supprimé est retourné par la méthode) :

*•*

function array.shift();

Suppression d’une partie des éléments :

*•*

function array.splice(firstElementKey, numberOfElementsToRemove);

Ajout d’un ou plusieurs élément(s) à la fin :

*•*

function array.push(element1, element2...);

Tri d’un tableau : function array.sort(compareFuntion);

*•*

où compareFuntion est une fonction permettant de comparer deux éléments du tableau qui a pour prototype : function compareFuntion(a, b);

et renvoie 0 si les éléments sont égaux, un nombre négatif si a est strictement inférieur à

b, et un nombre positif si a est strictement supérieur à b.

Pour les chaînes de caractère, on peut utiliser la méthode string.localCompare(that)

(similaire à strcmp).

* 1. **Exemple : traitement d’un formulaire avec *jQuery***
     1. **Qu’est-ce que *jQuery* ?**

##### La librairie *jQuery*, dont on peut obtenir le code et la documentation sur api.jquery.com, permet de simplifier la gestion de différents aspects d’une application côté client en *JavaScript* :

* + - * Gestion des événements utilisateur ;
      * Récupération des valeurs saisies dans un formulaire ;
      * Manipulation du document via le *DOM* ;

Requêtes asynchrones (transfert de données entre serveur et client en dehors du charge- ment initial de la page) ;

*•*

* + - * Codage des données pour la transfert (par exemple *JSON* ).

Pour utiliser *jQuery*, il suffit d’insérer son code dans un script, via une balise (remplacer

x.xx.x par le numéro de version utilisé sur jquery.com) :

<script src="https://code.jquery.com/jquery-x.xx.x.js"></script>

Pour travailler *offline*, on peut utiliser *jQuery* en local après téléchargement dans le répertoire courant :

<script src="./jquery-x.xx.x.js"></script>

Les méthodes de *jQuery* peuvent être appelées (avec des argument args) par l’abréviation

$(args).

* + 1. **Récupérer, filtrer et afficher les données d’un formulaire**

Le script suivant récupère les données d’un formulaire, les filtres par expressions régulières, et les affiche en modifiant le *DOM*.

Plus précisément, le script réalise les opération suivantes :

Déclaration d’un gestionnaire (fonction afficheDonneesForm) de l’événement submit du formulaire ayant formStudentData pour *ID* ;

*•*

* + - * Dans cette fonction afficheDonneesForm,
        + Récupération des valeurs saisies dans les éléments ayant nom et annee pour *ID*, qui sont respectivement un *input* et un *select*.
        + Test sur la forme (expression régulière, champs obligatoire,...) sur les valeurs des champs nom et année du formulaire (à l’aide d’un littéral de type expression régulière entre slashes /.../).
        + Ajout dans le <span> ayant spanResultat pour *ID* du code *HTML* du résultat de la saisie (affichage du nom et de l’année, ou le cas échéant un message d’erreur).
        + Empêcher le comportement par défaut de la soumission du formulaire (appel du script action côté serveur lors du *click* sur l’input de type submit).

Voici le fichier *HTML* :

##### Code Source 1.17 : /bases-js/ex10-jQueryFormTest.html (cf. Fig [1.6](#_bookmark31))

1

**<!doctype HTML> <***!−− Template de Document HTML −−***>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**Formulaires avec jQuery**</ t i t l e>**

**</head>**

**<body>**

**<form id=***” form Student Data”***>**

**<p>**

2

3

4

5

6

7

8

9



Figure 1.6 : Illustration du code source [1.17](#_bookmark30)

10

**<label for=***”nom”* **>**Nom**</ label>**

**<i nput name=***”nom”* **id=***”nom”* /**>**

**</p>**

**<p>**

**<s elect name=***” annee”* **id=***” annee”***>**

**<option value=***” Première”***>**Première ann é e**</option>**

**<option value=***” Deuxième”***>**Deuxième ann é e**</option>**

**</s elect>**

**</p>**

**<p>**

**<i nput type=***” submit ”* **value=***” Valider ”* /**>**

**</p>**

**</form>**

**<p>**

**<strong>**donn é es s a i s i e s&nbsp ; :**</s trong><br** /**>**

**<option value=***” c h o i s i s s e z ”* **selected disabled>***−−* c h o i s i s s e z *−−***</option>**

**<span id=***” span Resultat ”***></span>**

**</p>**

**<s cript src=***” j query . j s ”***></s cript>**

**<***!−− Les r é s u l t a t s de l a s a i s i e vont s ’ a f f i c h e r dans ce span −−***>**

**<s cript src=***” ex10−jQueryForm . j s ”***></s cript>**

**</body>**

**</html>**

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

Il est recommandé de mettre, dans la mesure du possible, le script à la fin du document, car cela limite le coût et le délai des chargements et *parsing* de la librairie *jQuery* lors d’un premier chargement (ou rafraîchissement) de la page. Voici le fichier *JavaScript* :

Code Source 1.18 : /bases-js/ex10-jQueryForm.js

1. */\*\* @ description Callback : Récupé re et a f f i c h e l e s inputs de formulaire .*
2. *\* L ’ impl é mentation u t i l i s e jQuery .*
3. *\* @ function afficheDonnees Form*
4. *\* @param { jQuery Event } event l ’ événement ( de type submit ) gér é par ce Handler*

−

5 *\*/*

1. **var** affiche Donnees Form = **function** ( event ) {
2. *// r écupé ration ( v ia jQuery ) de l a valeur de l ’ input d ’ ID ”nom”*
3. **var** nom = $ ( *”#nom”* ) . val () ;
4. *// r écupé ration ( v ia jQuery ) de l a valeur du s e l e c t d ’ ID ” annee”*

10

**var** annee = $ ( *”#annee”* ) . val ( ) ;

*// t e s t de champs o b l i g a t o i r e s et d ’ expressions r é g u l i è r e sur l e nom*

**i f** (( annee !== *” Première”* && annee !== *” Deuxième”* ) | |

*\ s ” ’ −]{1 ,}$/*

. t e

! */^ [ a−zA−ZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæ ¸ èéêö ì í î ïðñòóôõö÷øùúû*Āā*üýþÿ*

s t (nom) ) {

} ;

$ ( *”#span Resultat ”* ) . html ( *” Problème : forme d ’ un champs in correct . ”* ) ;

} **else** {

$ ( *”#span Resultat ”* ) . html ( *”***<em>***Nom :* **</em>***”* + nom + *”***<br** */***>***”* +

*”***<em>***Annee :* **</em>***”* + annee ) ;

}

*// Év i t e r d ’ appeler l ’” action ” par dé f aut ( requ ê t e sur un s c r i p t PHP, etc . )*

*// du formulaire l or s du c l i c k sur l e bouton submit*

event . prevent Default ( ) ;

*// Gestion de l ’ événement submit du formulaire .*

*// On dé f i n i t afficheDonnees Form comme g es t i on n a i re ( handler )*

*// de l ’ événement .*

$ ( *”#form Student Data”* ) . on ( *” submit ”* , affiche Donnees Form ) ;

11

12

13

14

15

16

17

18

19

20

21

22

23

24

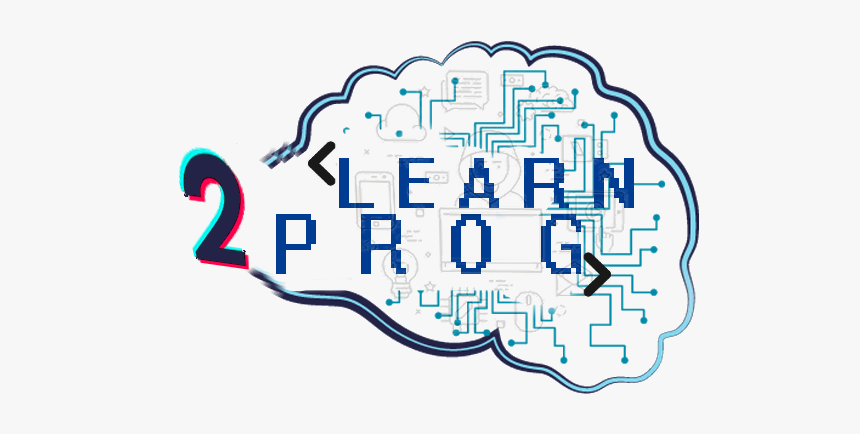
25

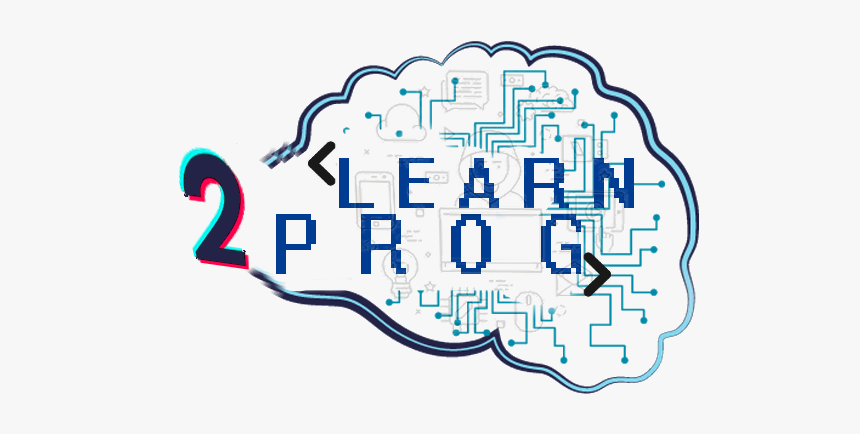
26

27

28

29





**Chapitre 2**

**Programmation Fonctionnelle et Objet en *JavaScript***

On distingue en *JavaScript* deux catégories de *patterns* (et éventuellement des *patterns* hy- brides) :

Les *patterns* dits *fonctionnels* s’appuient sur les aspects de *JavaScript* en tant que *langage fonctionnel*. Autrement dit, ces patterns exploitent les propriétés des fonctions *JavaScript* en tant que données, ainsi que la portée des variables dans ces fonctions.

*•*

##### Les *patterns* dits *prototypaux* s’appuient sur les aspects de *JavaScript* en tant que *langage prototypal*. Ceci est lié à une propriété que possèdent tous les objets *JavaScript*, appelée le prototype. Le prototype permet de partager des propriétés entre plusieurs objets, et il conduit naturellement à des notions d’héritage. Il permet aussi d’*augmenter* les objets pour leur rajouter des propriétés, bien après que ces objets aient été définis, y compris sur les types de base comme String.

*•*

Nous commencerons par voir un certain nombre de *patterns* fonctionnels, qui permettent de faire de la programmation objet avec des notions comme la visibilité, la structuration d’une application en modules (ou *packages*), des fabriques, ou encore des *patterns* permettant le découplage des composants d’une application à base d’événements, ou comme *subscriber/pu- blisher*.

Ces *patterns* peuvent paraître déconcertant au premier abord pour un développeur habitué aux langages objet classiques. Avec un peu d’habitude, on en vient à considérer que *JavaScript* est un excellent langage objet, très expressif et très souple. Cependant, certains problèmes de

conception du langage, qui n’ont pu être corrigés pour assurer la compatibilité ascendante, nécessitent quelques précautions, sous la forme de bonnes habitudes.

* 1. **Passages d’Arguments, Objets this et *Pattern* that**
     1. **Passage d’arguments par objets**

En *JavaScript*, il est souvent plus pratique, plutôt que de passer une série de paramètres, ce qui oblige à tenir compte de l’ordre de ces paramètres, de donner en argument à une fonction les données dans les propriétés d’un objet, soit construit à la volée, soit construit auparavant.

Ce *pattern* offre souvent plus de souplesse que la manière classique. Dans l’exemple suivant, la fonction génère le code *HTML* de l’objet passé en paramètre, sans savoir de quel type d’objet il s’agit. On l’utilise ensuite pour afficher une adresse.

Code Source 2.1 : /pattern-fonct/ex02-affichageObjetBasic.js

1

*/\*\* @summary Créee un cha î ne de caractère l i s i b l e qui repr é sente l ’ ob j e t .*

* *@ description On suppose que t ou t es l e s propri é t é s de l ’ ob j e t sont de type*
* *cha î ne ou nombre ( e l l e s peuvent ê t re automatiquement con v ert i es en cha î ne)*
* *@ function object To Html Table*

*\* @return Le code HTML pour a f f i c h e r l e s propri é t é s de l ’ ob j e t*

*\*/*

**var** objectTo Html Table = **function** ( spec ) {

**var** chaine = *”***<table><tbody>***”* ;

*// Parcours des propri é t é s de l ’ ob j e t spec pass é en argument*

**for** ( propertyName **in** spec ) {

*// La propri é t é es t dé f i n i e et non vide ,*

*// e l l e ne v i en t pas du prototype de l ’ ob j e t et ce n ’ es t pas une fonction*

**i f** ( spec [ propertyName ] && spec . hasOwnProperty ( propertyName ) && **typeof** spec [ propertyName ] != *” function ”* ) {

*// Concaté nation à une cha î ne . Les nombres sont con v ert i s .*

*\* @param { Object } spec − L ’ ob j e t dont l e s propri é t é s doivent ê t re a f f i c h é es*

chaine += *’***<tr><td** *s t y l e=”***text** *−***align** *:* **right** *;”***><em>***’* + propertyName + *” :***</**

**em></td>***”* +

*”***<td>***”* + spec [ propertyName ] + *”***</td></ tr>***”* ;

} ;

}

} ;

**return** chaine ;

chaine += *”***<tbody></table>***”* ;

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

Code Source 2.2 : /pattern-fonct/ex02-affichageObjetBasicTest.js

1

*// Invocation de l a fonction avec l e pattern*

**var** codeHTML = objectTo Html Table ({ id : *” 0 f 3 ea 759 b 1 ”* ,

numeroRue : *” 2 b i s ”* , rue : *”Rue de l a Paix ”* , complementAddr : *””* , code Postal : *” 63000 ”* ,

v i l l e : *” Clermont−Ferrand”* ,

*// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ;

pays : *” France”*

2

3

4

5

6

7

8

9

10

11

12

* + 1. **Invocation de Méthode avec le *Pattern “apply”***

##### Le *pattern apply* permet d’“appliquer” une méthode d’un certain objet à un autre objet. En d’autres termes, *pattern apply* permet d’exécuter une méthode d’un objet comme si elle était définie dans un autre objet. Plus précisément, lors de l’invocation de la méthode suivant le *pattern apply*, le code de la fonction est exécuté, mais chaque occurrence du mot réservé this, au lieu de faire référence à l’objet contenant la méthode, va faire référence à un autre objet, qui

est passé en premier argument lors de l’invocation. Les arguments ordinaires de la méthode, suivant la définition de ses paramètres, sont transmis, lors de l’invocation, dans un Array passé en second argument.

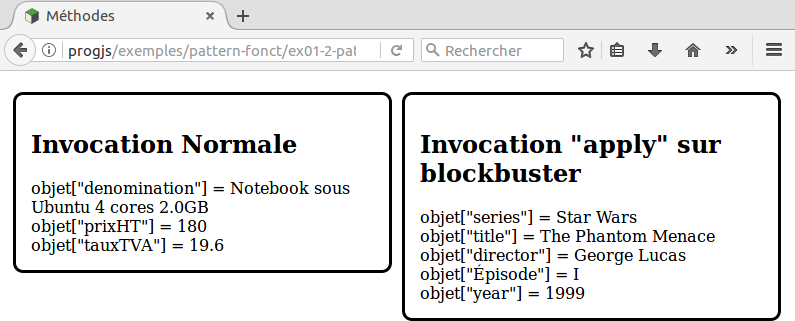


Figure 2.1 : Illustration du code source [2.3](#_bookmark37)

Code Source 2.3 : /pattern-fonct/ex01-2-patternApplyInvocationTest.js (cf. Fig [2.1](#_bookmark36))

1. */\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ” \*/*
2. **var** produit = {
3. *” denomination ”* : *” Notebook sous Ubuntu 4 cores 2 . 0GB”* , 4 *” prixHT”* : 1 8 0 . 0 ,

5 *”tauxTVA”* : 1 9 . 6 ,

6

1. */\*\* @ description Gé nère l e code HTML pour a f f i c h e r un ob j e t quelconque*
2. *\* @method get Html Objet*
3. *\* @param { s t r i n g } t i t r e t ex t e du t i t r e* **<h2>**

−

1. *\* @param { s t r i n g } b a l i s e type de b a l i s e HTML contenant l e code génér é*

−

11 \* (” p” , ”em” , ” d iv ” , ” span ” . . . )

12 *\*/*

1. getHtml : **function** ( t i t r e , b a l i s e ) {
2. **var** chaine = *”***<***”* + b a l i s e + *”***>***” // On ouvre l a b a l i s e ;*

15 chaine += *”***<h2>***”* + t i t r e + *”***</h2>***”* ;

1. *// Parcours de t ou t es l e s propri é t é s de l ’ ob j e t ( s t y l e ” foreach ”) :*
2. **for** (nom **in this** ) {
3. **i f** ( **this** . hasOwnProperty (nom) && **typeof this** [ nom ] !== *” function ”* ) { 19 chaine += *” ob j e t [ \” ”*+nom+*” \”] = ”*

20 + **this** [ nom ] + *”***</br>***”* ; *// Appel de l a mé thode t o S t r in g par dé f aut*

21 }

22 }

23

24

25 }

26 } ;

27

chaine += *”***</***”* + b a l i s e + *”***>***”* ; *// On ferme l a b a l i s e*

**return** chaine ;

1. */\*\* @ description Repré sente un f i lm de l a s é r i e ” Star Wars” \*/*
2. **var** b l o c k b u s te r = {
3. *” s e r i e s ”* : *” Star Wars”* ,
4. *” t i t l e ”* : *”The Phantom Menace”* ,
5. *” d i r ect or ”* : *” George Lucas”* ,
6. *”Épisode ”* : *” I ”* , 34 *” year ”* : *” 1999 ”*

35

} ;

*// Invocation c l as s i q u e de l a mé thode a f f i ch an t l e produit :*

codeHTML = produit . getHtml ( *” Invocation Normale”* , *” span”* ) ;

*// Invocation avec pattern apply de l a mé thode de ” produit ”*

*// pour a f f i c h e r ” b l ock b u s t er ” :*

codeHTML += produit . getHtml . apply ( blockbuster , *// Objet prenant p l ace de t h i s*

*// List e des arguments de l a mé thode :*

[ *” Invocation \” apply \” sur b l ock b u s t er ”* , *// t i t r e ” span”* ] ) ; *// b a l i s e*

*// Mise à jour de l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

36

37

38

39

40

41

42

43

44

45

46

* + 1. **Accès au Composite à partir du Composé (*pattern* that)**

Une méthode d’un objet imbriqué, tel qu’un composé de la partie [1.4.3](#_bookmark18) obtenus par une fa- brique, peut accéder aux propriétés du composite en utilisant une propriété ou une variable, traditionnellement appelée that, qui contient la référence du composite :

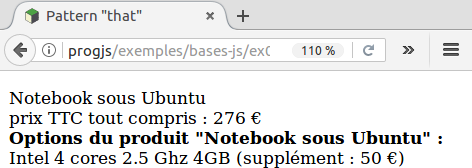


Figure 2.2 : Illustration du code source [2.4](#_bookmark40)

Code Source 2.4 : /pattern-fonct/ex06-returnedObjectCompositeAccess.js (cf. Fig [2.2](#_bookmark39))

1

*/\*\* @ description Li t t é ral dé f i n i s s a n t un ob j e t appel é ” produit ” \*/*

**var** produit = {

denomination : *” Notebook sous Ubuntu”* , prixHT\_base : 1 8 0 . 0 ,

tauxTVA : 2 0 . 0 ,

*/\*\* @ description Construction d ’ un ob j e t dans une mé thode*

*\*/*

get Options : **function** ( ) {

*// Mé morisation du composite pour accès par l e composé*

**var** that = **this** ; *// pattern t hat ( p art i e 1 )*

*// Cré ation de l ’ ob j e t option dans l a fonction (” f ab r i q u e ”)*

**return** {

p r o c e s s o r : *” I n t e l 4 cores 2 . 5 Ghz”* , memory : *” 4GB”* ,

*” prix suppl é mentaire HT”* : 5 0 . 0 ,

*/\*\* @ description Gé nère l e code HTML des options*

*\* @method getHTML*

*\*/*

getHtml : **function** () {

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

|  |  |
| --- | --- |
| 20 | *// Accès au composite à p ar t i r d ’ une mé thode du composé* |
| 21 | **var** codeHTML = *”***<b>***O ptions du produit \””* |
| 22 | + that . denomination *// pattern t hat ( p art i e 2 )* |
| 23 | + *”\” :* **</b>***”* ; |
| 24 | codeHTML += *”***<br** */***>***”* + **this** . p r o c e s s o r + *” ”* + **this** . memory + |
| 25  26  27  28 | *” ( suppl ément : ”* + **this** [ *” prix suppl é mentaire HT”* ] + *” &euro ;)”* ;  **return** codeHTML ;  }  } ; |
| 29 | } , |
| 30 | */\*\* @ description Gé nère l e code HTML complet du produit* |
| 31 | *\* @method getHTML* |
| 32 | *\*/* |
| 33 | getHtml : **function** () { |
| 34 | **return this** . denomination + |
| 35 | *”***<br** */***>***p r i x TTC t out compris : ”* |
| 36 | + ( **this** . prixHT\_base |
| 37 | + ( **this** . get Options ( ) [ *” prix suppl é mentaire HT”* ] | | 0 . 0 ) ) |
| 38 | \*(1 .0 + **this** . tauxTVA / 1 0 0 . 0 ) |
| 39 | + *” &euro ;***<br** */***>***”* + **this** . get Options ( ) . getHtml ( ) + *”***<br** */***>***”* ; |
| 40 | } |
| 41 | } ; |
| 42 |  |
| 43 | *// appel d ’ une mé thode :* |
| 44 | codeHTML = *”***<p>***”* + produit . getHtml ( ) + *”***</p>***”* ; |
| 45 | *// Mise à jour de l a vue* |
| 46 | document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ; |

* 1. **Le *Pattern* Module**
     1. **Cycle de Vie d’une Variable Locale à une Fonction**

##### Une variable locale à une fonction, déclarée avec le mot clé var, n’est plus accessible à l’extérieur de la fonction. Cependant, contrairement à ce qui se passe dans un langage comme le *C*, la variable locale peut continuer à exister en mémoire après la fin de l’exécution de la fonction.

* + - 1. **Rappel sur les variables locales en *C***

Dans le langage *C*, les variables locales à une fonction sont créées dans une zone mémoire appelée la *pile*, et sont automatiquement détruites lors du retour de la fonction après son invocation.

Dans l’exemple suivant, une fonction *C* retourne l’adresse d’une variable locale, puis la fonction main tente d’accéder à la zone mémoire correspondante.

Code Source 2.5 : /pattern-fonct/fonction-C/variableLocale.c

1

**#include <s** td i o . h**>**

*/\*\* Fonction bugg ée qui retourne l ’ adresse d ’ une v ar i ab l e l oc a l e \*/*

**int** \* maFonction ( ) {

**int** x = 2 ; *// v ar i ab l e l oc a l e x*

**int** \* pRetour = &x ; *// Pointeur sur l a v ar i ab l e x*

p r i n t f ( *”%d\n”* , \* pRetour ) ; *// Affiche x , donc 2*

2

3

4

5

6

7

8

}

**return** pRetour ; *// Retourne l ’ adresse d ’ une v ar i ab l e l oc a l e ! ! ! ! !*

**int** main () {

**int** \* pointeur Sur X = maFonction ( ) ; *// Invocation de l a fonction*

p r i n t f ( *”%d\n”* , \* pointeur Sur X ) ; *// Erreur mémoire ! ! ! !*

**return** 0 ;

}

9

10

11

12

13

14

15

Même si l’exécution peut éventuellement fonctionner sur certain compilateurs, le code n’est pas défini dans le standard du *C ANSI*, et le comportement peut donner n’importe quoi ! On peut mettre en évidence le *bug* avec l’outil *valgrind*, qui détecte une erreur mémoire (ici sur la ligne 13) :

==3894== Use of uninitialised value of size 8

==3894== at 0x4E8476B: \_itoa\_word (\_itoa.c:179)

==3894== by 0x4E8812C: vfprintf (vfprintf.c:1631)

==3894== by 0x4E8F898: printf (printf.c:33)

==3894== by 0x400619: main (variableLocale.c:13)

* + - 1. **Survivance des variables locales en *JavaScript***

##### En *JavaScript*, le cycle de vie d’une variable locale peut aller au delà du retour de la fonction après son invocation, si on retourne une entité (objet, fonction, tableau, etc.) qui fait référence à cette variable locale.Ceci est possible du fait que les variables locales ne sont pas gérés en mémoire via une pile, mais comme de la mémoire dynamique libérée par le *garbage collector*. La mémoire pour la variable locale ne peut donc pas être détruite tant qu’il existe une référence à cette variable !

Dans l’exemple suivant, une fonction maFonction retourne une autre fonction locale accesVariableX, qui elle même retourne la variable locale x de maFonction. On peut alors accéder, via la fonction

locale retournée, à la variable locale x, bien après que l’exécution de la fonction maFonction

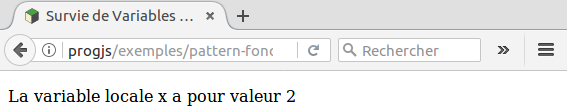


Figure 2.3 : Illustration du code source [2.6](#_bookmark44)

Code Source 2.6 : /pattern-fonct/ex01-survieVariableLocaleJS.js (cf. Fig [2.3](#_bookmark43))

1

**var** maFonction = **function** () {

**var** x = 2 ; *// Val r i ab l e l oc a l e x de type ” number” , valeur é gale à 2*

*// Fonction l oc a l e qui accède à l a v ar i ab l e l oc a l e x et l a retourne*

**var** acces Variable X = **function** () {

**return** x ; *// U t i l i s a t i o n de l a v ar i ab l e l oc a l e x*

} ;

2

3

4

5

6

7

8

9

10

} ;

**return** acces Variable X ; *// On retourne l a fonction l oc a l e*

codeHTML = *”La v ar i ab l e l oc a l e x a pour*

**var** getX = maFonction () ; *// On r é cupère l a fonction retourn ée par maFonction*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

*valeur ”* + getX () ;

11

12

13

14

15

* + 1. **Principe Général du *Pattern* Module**

##### Le *pattern Module* permet de créer des composants qui peuvent jour le rôle que jouent les classes dans les langages objet classiques. Il permet, entre autre, de créer des données et méthodes privées, et une interface publique avec d’autres données et méthodes, qui sont accessibles de l’extérieur, et qui peuvent, elles, accéder aux propriétés privées.

Le *pattern* consiste à créer une fonction. Les données et méthodes privées sont simplement

des variables locales de la fonction. Elles ne sont donc pas visibles du monde extérieur à la fonction. La fonction renvoie un objet, qui constitue l’interface publique du module, dont les propriétés (données, objets ou fonctions) accèdent aux variables privées. Lorsque l’objet est retourné, on ne peut plus accéder directement aux variables locales de la fonction, mais celles- ci restent vivantes (leur cycle de vie ne se termine pas) tant que l’objet retourné qui s’y réfère n’est pas lui-même détruit. Autrement dit, on peut continuer à manipuler ces variables locales au travers des méthodes de l’interface publique.

Code Source 2.7 : /pattern-fonct/ex01-modulePattern.js

* 1. */\*\* @ description I l l u s t r a t i o n du pattern Module*
  2. *\* permettant d ’ impl émenter l a v i s i b i l i t é en Java Script .*

3 *\*/*

1. **var** mySecretModule = **function** ( d e f a u l t S e c r e t V a l u e ) {
2. */\*\* @ description Donnée priv ée avec une valeur par dé f aut*
3. *\* ( v ar i ab l e l oc a l e )*
4. *\* @ private { s t r i n g } Private Secret = ( default Secret Value | ””)*

8 *\*/*

9 **var** my Private Secret = ( ( d e f a u l t S e c r e t V a l u e && *// s ’ i l y a une valeur en paramètre*

1. **typeof** d e f a u l t S e c r e t V a l u e === *” s t r i n g ”* )
2. && d e f a u l t S e c r e t V a l u e )
3. | | *””* ; *// s i pas de valeur en paramètre*

13

1. */\*\* @ description Teste une expression r é g u l i è r e ( f i x ée) sur une cha î ne*
2. *\* @ function myRegexTestMethod*
3. *\* @ private*
4. *\* @param { s t r i n g } chaine La cha î ne à t e s t e r*

−

18 *\*/*

1. **var** myRegexTestMethod = **function** ( chaine ) {
2. **return** ( **typeof** chaine === *” s t r i n g ”* ) && */^ [ a z ] \* $/* i . t e s t ( chaine ) ; 21 } ;

−

22

1. */\*\* @summary In t er f ace publique du module , suivant l e ” pattern module ” .*
2. *\* @ description On cr ée un ob j e t qui va ê t re rendu p u b l i c*
3. *\* Cet ob j e t va ê t re retourn é , mais pas l e s donné es priv é es .*
4. *\* Les mé thodes de cet ob j e t con s t i t u eron t l e s mé thodes pu b l i q u es du module ,*
5. *\* qui u t i l i s e n t l e s v ar i ab l e s ( et f onct i ons ) priv é es ( v ar i ab l e s l o c a l e s ) .*

28 *\*/*

1. **var** p u b l i c I n t e r f a c e = {
2. */\*\* @ description Donnée publique avec une valeur par dé f aut*
3. *\* @ public { s t r i n g } donnee Publique = ’ donnée publique par dé f aut ’*

32 *\*/*

33 donnee Publique : *’ donnée publique par dé f aut ’* , 34

1. */\*\* @ description S e t t er pour modifier l a donnée priv ée my Private Secret*
2. *\* @method s e t S ecret*
3. *\* @ public*
4. *\* @param { s t r i n g } secret Value La valeur à t e s t e r puis a f f e c t e r*

−

1. *\* @throws Si secret Value ne s u i t pas l ’ expression r é g u l i è r e myRegexTestMethod*

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64 }

*\*/*

s e t S e c r e t : **function** ( s e c r e t V a l u e ) {

// Test d ’ expression r é g u l i è r e v ia l a f on c t in priv ée :

**i f** ( myRegexTestMethod ( s e c r e t V a l u e ) ) { my Private Secret = s e c r e t V a l u e ;

} **else** {

**throw** {

name : ” I l l egal Argument Exception ” ,

message : *” Le s ecret ”* + s e c r e t V a l u e + *” es t i n v a l i d e . ”*

} ;

}

} ,

/\*\* @ description Accesseur pour acc é der à l a donnée priv ée my Private Secret

* *@method g e t S ecret*
* *@ public*
* *return La valeur de l a donnée priv ée my Private Secret*

\*/

g e t S e c r e t : **function** ( ) {

**return** my Private Secret ;

} ,

} ; // Fin de p u b l i c In t er f ace

// L ’ i n t er f ace publique du module es t retourn ée pour u t i l i s a t i o n hors de l a fonction

**return** p u b l i c I n t e r f a c e ;

Le fichier suivant teste les accès aux données et méthodes publiques du module suivant le

*pattern module* :

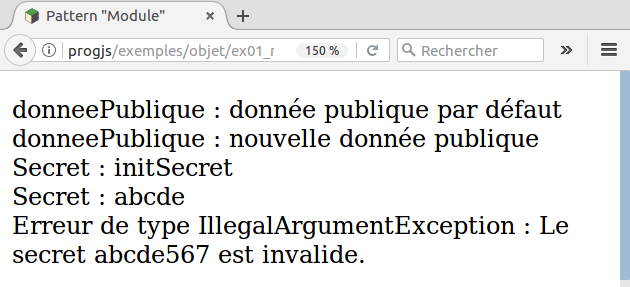


Figure 2.4 : Illustration du code source [2.8](#_bookmark47)

Code Source 2.8 : /pattern-fonct/ex01-modulePatternTest.js (cf. Fig [2.4](#_bookmark46))

1

*// Cré ation du module par invocation du ” pattern module” :*

* 1. **var** secret Module = mySecretModule ( *” i n i t S ec r e t ”* ) ;
  2. *// Code HTML à ajouter au paragraphe d ’ ID paragraphe Resultat*
  3. **var** codeHTML = *””* ;
  4. *// Accès à l a donnée publique de l ’ i n t er f ace :*
  5. codeHTML += *” donnee Publique : ”* + secret Module . donnee Publique + *”***<br** */***>***”* ;
  6. *// Modification de l a donnée publique de l ’ i n t er f ace :*
  7. secret Module . donnee Publique = *” nou v e l l e donnée publique ”* ;
  8. codeHTML += *” donnee Publique : ”* + secret Module . donnee Publique + *”***<br** */***>***”* ; 10

1. *// Accesseur de s ecret ( v ar i ab l e l oc a l e priv ée du ” pattern module ”) :*
2. codeHTML += *” Secret : ”* + secret Module . g e t S e c r e t () + *”***<br** */***>***”* ;
3. *// Tentative de modifier l e s ecret ( l e s e t t e r p u b l i c f a i t des t e s t s ) :*
4. try {
5. secret Module . s e t S e c r e t ( *” abcde”* ) ; *// La cha î ne n ’ entra î ne pas d ’ erreur*
6. codeHTML += *” Secret : ”* + secret Module . g e t S e c r e t ( ) + *”***<br** */***>***”* ;
7. } **catch** ( e ) {
8. codeHTML += *” Erreur de type ”* + e . name + *”***<br** */***>***Message : ”* + e . message + *”***<br** */*

**>***”* ;

19 }

1. *// Tentative de modifier l e s ecret ( l e s e t t e r p u b l i c f a i t des t e s t s ) :*
2. try {
3. secret Module . s e t S e c r e t ( *” abcde567 ”* ) ; *// La cha î ne gé nère une erreur*
4. codeHTML += *” Secret : ”* + secret Module . g e t S e c r e t ( ) + *”***<br** */***>***”* ;
5. } **catch** ( e ) {
6. codeHTML += *” Erreur de type ”* + e . name + *” : ”* + e . message + *”***<br** */***>***”* ; 26 }

27 document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

##### Le mécanisme du langage essentiel pour ce *pattern* est la portée (*scope*) des variables locales à une fonction, qui s’étend aux sous-fonctions de la fonction, et à leurs sous-fonctions...

* 1. **Exemple de fabrique sommaire**

Dans l’exemple suivant, une fabrique, suivant un *pattern* module très sommaire, construit un objet de type adresse (éventuellement partiellement rempli), en sélectionnant les propriétés que l’objet en paramètre qui sont dans une liste.

Cet exemple est plutôt un *exemple d’école* et nous verrons plus loin des exemple plus

complets, les propriété de l’objet retourné manipulant des données (attributs) privées.

Code Source 2.9 : /pattern-fonct/ex03-methodLitteralParam.js

1

*/\*\* @ description Fabrique cr éant un ob j e t de type ” Adresse”*

* *avec des ” a t t r i b u t s ” ( des propri é t é s attendues dé termin é es ) f i x é s .*
* *On vé r i f i e que l e s propri é t é s sont b ien sp é c i f i é espour l e s ” a t t r i b u t s ” .*

*\* @param { Object } spec − Objet avec l e s donné es d ’ une adresse*

*\* @param { s t r i n g } spec . id − l ’ i d e n t i f i a n t unique de l ’ adresse*

*\* @param { s t r i n g } spec . numeroRue − l e numéro de l a rue/ p l ace*

*\* @param { s t r i n g } spec . rue − l a rue/ p l ace*

*\* @param { s t r i n g } spec . complementAddr − Le bâtiment , l i eu d i t . . .*

*\* @param { s t r i n g } spec . code Postal − Le code p os t a l*

*\* @param { s t r i n g } spec . v i l l e − La v i l l e*

*\* @param { s t r i n g } spec . pays − Le pays*

* *@return Une instance d ’ adresse ” v a l i d ée” ( ob j e t avec l e s propri é t é s )*
* *@throws Si ” spec ” ne contient pas t ou t es l e s propri é t é s attendues .*

*\*/*

2

3

4

5

6

7

8

9

10

11

12

13

14

15

**var** f a b r i q u e A d r e s s e Ve r s i o n 1 = **function** ( spec ) {

*// Objet à retourner i n i t i a l em en t v ide*

**var** a d r e s s e = {} ;

*// List e des ” a t t r i b u t s ” d ’ une adresse ( propri é t é s attendues )*

**var** l i s t e P r o p r i e t e s = [ *” id ”* , *” numeroRue”* , *” rue”* , *” complementAddr”* ,

*” code Postal ”* , *” v i l l e ”* , *” pays ”* ] ;

*// Parcours des propri é t é s de l ’ ob j e t spec pass é en argument*

**for** ( propertyName **in** spec ) {

**i f** ( spec . hasOwnProperty ( propertyName ) ) {

*// Si l a propri é t é ex i s t e dans l e type adresse :*

**i f** ( l i s t e P r o p r i e t e s . index Of ( propertyName ) **>**= 0 ) { a d r e s s e [ propertyName ] = spec [ propertyName ] ;

} **else** {

**throw** {

name : *” Unknown Property Exception”* ,

message : *” Propri é t é de l ’ adresse inconnue . ”*

} ;

} Les

}

}

**return** a d r e s s e ;

} ;

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

Code Source 2.10 : /pattern-fonct/ex03-methodLitteralParamTest.js

1

*// cr é ation d ’ une instance avec l a f ab r i q u e*

**var** a d r e s s e = f a b r i q u e A d r e s s e Ve r s i o n 1 ({ id : *” 0 f 3 ea 759 b 1 ”* ,

numeroRue : *” 2 b i s ”* , rue : *”Rue de l a Paix ”* , complementAddr : *””* , code Postal : *” 63000 ”* ,

v i l l e : *” Clermont−Ferrand”* ,

*// Géné ration d ’HTML par i n t ros p ect i on :*

codeHTML = objectTo Html Table ( a d r e s s e ) ;

*// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ;

pays : *” France”*

2

3

4

5

6

7

8

9

10

11

12

13

14

* 1. **Structuration d’une application**

L’un des principaux défauts de *JavaScript* est sa tendance à créer, parfois sans faire exprès, des variables globales, ce qui a tendance à créer des interactions involontaires entre des parties du code qui n’ont rien à voir, ce qui génère des *bugs* difficiles à débusquer...

Nous allons voir maintenant comment rédiore les nombres de variables globales de notre programme à une seule variable, ici appelée myApp, qui contient toute notre application.

L’objet myApp, initialement, ne contient que deux méthodes :

Une méthode addModule qui permet d’ajouter un objet quelconque (de type pattern-fonct, Function, Array, etc.) sous la forme de propriété de l’application.

*•*

Une méthode init, qui permet de rajouter un ensemble de propriétés prédéfinies, sans avoir à les créer une par une.

*•*

Code Source 2.11 : /pattern-fonct/ex04-structureApplication.js

1

*/\*\* Dé f i n i t i o n d ’ une v ar i ab l e ap p l i cat i on .*

* *L ’ ap p l i cat i on es t i n i t i a l em en t v ide et ne comporte que l a f on c t i on n a l i t é*
* *permettant d ’ ajouter des modules .*
* *Une mé thode i n i t () permet d ’ i n i t i a l i s e r p l u s i eu r s modules .*

*\**

*\*/*

**var** myApp = {

*/\*\* Mé thode qui ajoute un module à notre ap p l i cat i on*

* + *Un module peut ê t re n ’ importe quel ob j e t qui contient*
  + *des donné es ou des mé thodes . . .*
  + *@method addModule*
  + *@param { Object | function | s t r i n g | regex | number | Array | . . . } module Object*

*\**

addModule : **function** ( moduleName , module Object ) {

**i f** ( **typeof** moduleName === *” s t r i n g ”* &&

*\*/*

*− un ob j e t ou valeur quelconque à ajouter à notre ap p l i cat i on .*

} **else** {

*/^ [ a−z ] { 1 ,}[ a−z0 −9\\_]\* $/* i . t e s t ( moduleName ) ) {

**this**

[ moduleName ] = module Object ;

} ,

}

**throw** {

name : *” I l l ageal Argument Exception ”* ,

message : *” Impossible de cr é er l e s module : nom ”* + moduleName

+ *” i l l é gal ”*

}

*/\*\* Ajoute t ou t es l e s propri é t é s d ’ un ob j e t à notre ap p l i cat i on .*

*\* @method i n i t*

} ;

*\*/*

i n i t : **function** ( spec ) {

**for** ( **var** attribute Name **in** spec ) {

**i f** ( spec . hasOwnProperty ( attribute Name ) ) {

**this** . addModule ( attribute Name , spec [ attribute Name ] ) ;

}

}

}

*\* @param { Object } spec − ob j e t contenant l e s propri é t é s à ajouter .*

*// I n i t i a l i s a t i o n de l ’ ap p l i cat i on avec un module metier i n i t i a l em en t v ide .*

myApp. i n i t ({ metier : {}

}) ;

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

Nous utilisons maintenant ce squelette d’application et nous créons dans notre application un module metier.

Nous utilisons ensuite le *pattern apply* qui nous permet d’utiliser la méthode myApp.addModule

en prenant comme ”this” un autre objet que myApp.

En appliquant donc la méthode myApp.addModule en prenant myApp.metier comme ”this”,

nous créons un sous-module de myApp.metier, appelé myApp.metier.sousModule. Ce sous- module contient une propriété essai.

Code Source 2.12 : /pattern-fonct/ex04-structureApplicationTest.js

1. *// ajout d ’ une propri é t é au mé t i e r :*
2. myApp. metier . coucou = *” t e s t ”* ; 3
3. *// Ajout d ’ un sous module au module myApp. metier*

−

1. *// On applique ( pattern apply ) addModule en prenant t h i s = myApp. metier*
2. myApp. addModule . apply (myApp. metier ,
3. [ *” sousModule”* ,
4. { e s s a i : *” Je su i s l a propri é t é \” es s a i \” du sous module”* }

9 ] ) ;

10

1. */\*\* @ description Module myApp. view avec une propri é t é ”codeHTML”*
2. *\* @module myApp. view*
3. *\* La propri é t é ”codeHTML” contienda l e code HTML de l a vue*
4. *\* ( ins ér é dans l e paragraphe d ’ ID ” paragraphe Resultat ”)*

15 *\*/*

16 myApp. addModule ( *” view ”* , {codeHTML : *””* }) ; 17

1. */\*\* @ description Programme Principal de l ’ ap p l i cat i on*
2. *\* @method mainFunction*
3. *\* @ public*

21 *\*/*

1. myApp. addModule ( *” mainFunction”* , **function** ( ) {
2. myApp. view . codeHTML += *”Exé cution de myApp. mainFunction . . .***<br** */***>***”* ;
3. myApp. view . codeHTML += *”myApp. metier . coucou : ”*

25 + myApp. metier . coucou + *”***<br** */***>***”* ;

26 myApp. view . codeHTML += *”myApp. metier . sousModule . es s a i : ”*

27 + myApp. metier . sousModule . e s s a i ; 28 }) ;

29

1. *// Exé cution de l a mé thode mainFunction*
2. myApp. mainFunction ( ) ;
3. *// U t i l i s a t i o n de ”myApp. view . codeHTML” pour géné rer l a vue*
4. document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML

34 = myApp. view . codeHTML ;

”

Code Source 2.13 : Fichiers *JS* inclus dans ex04-structureApplicationTest.html

**<p id=***” paragraphe Resultat ”***></p>**

**<s cript src=***” ex 04 −structure Application . j s ”***></s cript>**

**<s cript src=***” ex 04 −structure Application Test . j s ”***></s cript>**

1

2

3

* 1. **Exemple : un module metier.regexUtil**

L’exemple suivant montre l’utilisation du *pattern* Module pour créer un sous-module métier utilitaire pour tester des expressions régulières courantes :

Expressions formées avec les caractères du langage courant dans une des langues dont les accents sont normalisés dans la norme ISO 8859 1 (*Latin-1*, Europe occidentale), admettant aussi les guillemets, apostrophes et traits d’union (tiret haut).

*−*

*•*

* Mêmes caractère que la précédente mais admettant en outre les chiffres.

Mêmes caractère que la précédente mais admettant en outre les caractères de ponctuation (; . , ! ? :) et les parenthses.

*•*

Trois expressions régulières constances (donc pré-compilées) sont définies comme données statiques (en un seul exemplaire) privées. L’interface fournit trois méthodes pour tester ces expressions régulière sur une chaîne, avec éventuellement des conditions de longueur minimale ou maximale sur la chaîne (exemple : champs obligatoire...).

Code Source 2.14 : /pattern-fonct/ex05-modulePatternRegex.js

1. */\*\* @ description Sous module du mé t i e r : u t i l i t a i r e pour expressions r é g u l i è r e s*

−

1. *\* L ’ ob j e t cr éé es t l ’ i n t er f ace publique retourn ée par une fonction*
2. *\* qui s u i t l e ” pattern module ” .*
3. *\* La fonction retourne son i n t er f ace : un ob j e t avec ses mé thodes .*
4. *\* Cet ob j e t es t ajout é comme sous module au module ” metier ” par ” addModule ” .*

−

1. *\* @module r eg ex U t i l*
2. *\* @augments myApp. metier*

8 *\*/*

9 myApp. addModule . apply (myApp. metier , [ *” r eg ex U t i l ”* , **function** () { 10

11 ////////////////////////////////////////////////

12 // Variables et mé thodes ” s t a t i q u e s ” priv é es

13

1. */\*\* @ description Expression r é g u l i è r e constante pour l a langue n a t u r e l l e*
2. *\* Admet l e s caractères d ’ europe occid en t a l e ( jeu l a t i n 1 ) et espaces .*
3. *\* @constant*
4. *\* @ private*

18 *\*/*

19 **var** regex Latin 1

20 = /^ [ a−zA−ZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæ ¸ èéêö ì í î ïðñòóôõö÷øùúû

21 Āā*üýþÿ\ s ” ’\ −]\*$/* i ;

1. */\*\* @ description Expression r é g u l i è r e constante pour l a langue et c h i f f r e s*
2. *\* @constant*
3. *\* @ private*

25 *\*/*

1. **var** regex Latin 1 With Digits =
2. */^ [ a−zA−ZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæ ¸ èéêö ì í î ïðñòóôõö÷øùúû*

28 Āā*üýþÿ\ s ”’\ −0 −9]\*$/* i ;

1. */\*\* @ description Expression regex constante langue + c h i f f r e s + ponctuation*
2. *\* @constant*
3. *\* @ private*

32 *\*/*

1. **var** regex Latin 1 With Digits Punctuation =
2. */^ [ a−zA−ZÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖ×ØÙÚÛÜÝÞßàáâãäåæ ¸ èéêö ì í î ïðñòóôõö÷øùúû*

35 Āāüýþÿ\ s ” ’\ −0 −9\ ;\.\ ,\ !\ ?\ :\(\) ] \* $/ i ;

1. */\*\* @ description Valide une expression r é g u l i è r e sur une chaine*
2. *\* avec conditions de longueur ( minimale ou maximale) op t i on n e l l es*
3. *\* @method validate Regex*
4. *\* @ private*
5. *\* @param { Object } spec − ob j e t contenant l e s donné es du t e s t à e f f ec t u e r*
6. *\* @param { function } spec . regex Test −*
7. *\* fonction de t e s t qui renvoie t rue en cas de succès*
8. *\* et un message d ’ erreur en cas d ’ é chec*
9. *\* @param { s t r i n g } spec . chaine cha î ne de caractères à t e s t e r*

−

1. *\* @param {number} [ spec . minLength=0] longueur minimale pour l a cha î ne*

−

1. *\* @param {number} [ spec . maxLength=undefined ] longueur maximale pour l a cha î ne*

−

1. *\* @return { boolean | s t r i n g } t rue s i l e s conditions sont s a t i s f a i t e s ,*
2. *\* un message d ’ erreur pour un u t i l i s a t e u r sinon .*

49 *\*/*

1. **var** validate Regex = **function** ( spec ) {
2. **i f** ( **typeof** spec . chaine === *” s t r i n g ”*
3. && ( ! spec . minLength | | spec . chaine . l e n g th **>**= spec . minLength )
4. && ( spec . maxLength === undefined | |
5. spec . chaine . l e n g th **<**= spec . maxLength ) 55 ) {

56 **return** spec . regex . t e s t ( spec . chaine ) ; 57 }

58 **return** ” Erreur : longueur de l ’ entr ée ( champ ob l i g a t o i r e , t rop l ong . . . ) ” ; 59 } ;

60

1. */\*\* @ description In t er f ace publique du ” pattern module” pour regex Utils ,*
2. *\* retourn é par l a fonction , contient l e s mé thodes pu b l i q u es du module . \*/*
3. **var** p u b l i c I n te r f a c e Re g e x = {
4. */\*\* @ description t e s t e l ’ expression du l angage naturel avec espaces*
5. *\* @method t est Regex Latin 1*
6. *\* @ public*
7. *\* @param { Object } spec ob j e t contenant l e s donné es du t e s t à e f f ec t u e r*

−

1. *\* @param { s t r i n g } spec . chaine cha î ne de caractères à t e s t e r*

−

1. *\* @param {number} [ spec . minLength=0] longueur minimale de l a cha î ne*

−

1. *\* @param {number} [ spec . maxLength=undefined ] longueur max de l a cha î ne*

−

1. *\* @return { boolean | s t r i n g } t rue s i l e s conditions sont s a t i s f a i t e s ,*
2. *\* un message d ’ erreur pour un u t i l i s a t e u r sinon .*

73 *\*/*

1. test Regex Latin 1 : **function** ( spec ) {
2. *// Ajout d ’ une propri é t é à spec ( augmentation )*
3. spec . regex = regex Latin 1 ;
4. **return** validate Regex ( spec ) ; 78 } ,

79

1. */\*\* @ description t e s t e l ’ expression du l angage naturel , espaces , c h i f f r e s*
2. *\* @method t est Regex Latin 1 With Digits*
3. *\* @ public*
4. *\* @param { Object } spec ob j e t contenant l e s donné es du t e s t à e f f ec t u e r*

−

1. *\* @param { s t r i n g } spec . chaine cha î ne de caractères à t e s t e r*

−

1. *\* @param {number} [ spec . minLength=0] longueur minimale de l a cha î ne*

−

1. *\* @param {number} [ spec . maxLength=undefined ] longueur max de l a cha î ne*

−

1. *\* @return { boolean | s t r i n g } t rue s i l e s conditions sont s a t i s f a i t e s ,*
2. *\* un message d ’ erreur pour un u t i l i s a t e u r sinon .*

89 *\*/*

1. test Regex Latin 1 With Digits : **function** ( spec ) {
2. *// Ajout d ’ une propri é t é à spec ( augmentation )*
3. spec . regex = regex Latin 1 With Digits ;
4. **return** validate Regex ( spec ) ; 94 } ,

95

96 /\*\* @ description t e s t e l e l angage naturel , espaces , c h i f f r e s et

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

Le fichier *HTML* réalise des tests des méthodes du module regexUtil sur un jeu de chaînes, et affiche les résultats dans une table.

*ponctuation*

* *@method t est Regex Latin 1 With Digits Punctuation*
* *@ public*

*\* @param { Object } spec − ob j e t contenant l e s donné es du t e s t à e f f ec t u e r*

*\* @param { s t r i n g } spec . chaine − cha î ne de caractères à t e s t e r*

*\* @param {number} [ spec . minLength=0] − longueur minimale de l a cha î ne*

*\* @param {number} [ spec . maxLength=undefined ] − longueur max de l a cha î ne*

*\* @return { boolean | s t r i n g } t rue s i l e s conditions sont s a t i s f a i t e s ,*

*\**

*un message d ’ erreur pour un u t i l i s a t e u r sinon .*

test Regex Latin 1 With Digits Punctuation : **function** ( spec ) {

*// Ajout d ’ une propri é t é à spec ( augmentation )* spec . regex = regex Latin 1 With Digits Punctuation ; **return** validate Regex ( spec ) ;

}

} ; *// f i n du l i t t é ral dé f i n i s s a n t public Interface Regex*

*// On retourne l ’ ob j e t contenant l ’ i n t er f ace publique ( pattern ” module ”) .*

**return** p u b l i c I n te r f a c e Re g e x ;

}() ] *// f i n ET APPEL de l a fonction qui cr ée l ’ ob j e t ” r eg ex U t i l ”*

) ; *// f i n de l ’ appel de l a mé thode myApp. addModule avec l e pattern ” apply ”*

*// ( ajout de l ’ ob j e t public Interface Regex au metier , sous l e nom r eg ex U t i l )*

*\*/*

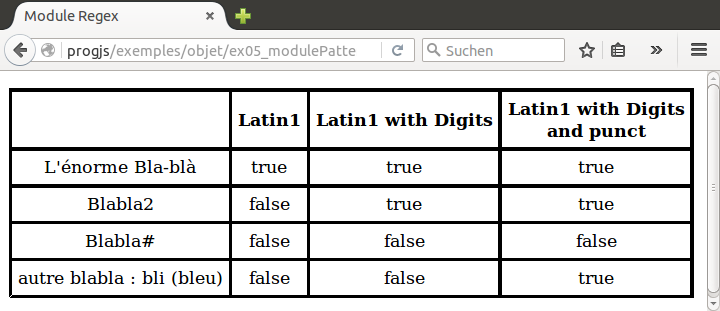


Figure 2.5 : Illustration du code source [2.15](#_bookmark52)

Code Source 2.15 : /pattern-fonct/ex05-modulePatternRegexTest.js (cf. Fig [2.5](#_bookmark51))

1

myApp. addModule ( *” mainFunction”* ,

*/\*\* @ description Programme Principal : t e s t du module myApp. metier . r eg ex U t i l*

* *@method mainFunction*
* *@ public*

*\*/*

**var** tab Chaines = [ *”L ’ énorme Bla−b l à”* , *” Blabla 2 ”* , *” Blabla#”* , *” autre b l ab l a :*

*// Chaî nes pour l e s t e s t s d ’ expressions r é g u l i è r e s :*

**function** () {

*b l i ( b l eu )”* ] ;

**var** i ;

*// Raccourci par copie de r é f é rence :* **var** r e g e x U t i l = myApp. metier . r e g e x U t i l ; **var** codeHTML = *”***<table><thead>***”* +

2

3

4

5

6

7

8

9

10

11

12

*”***<tr><th></th><th>***L atin 1***</th><th>***L atin 1 with D i g i t s***</th>***”* +

*”***<th>***L atin 1 with D i g i t s***<br** */***>***and punct***</th></ tr></thead><tbody>***”* ;

**for** ( i = 0 ; i**<** tab Chaines . l e n g th ; i++){

codeHTML += *”***<tr><td>***”* + tab Chaines [ i ] + *”***</td>***”*

+ *”***<td>***”* + r e g e x U t i l . test Regex Latin 1 ({

chaine : tab Chaines [ i ]

}) + *”***</td>***”*

+ *”***<td>***”* + r e g e x U t i l . test Regex Latin 1 With Digits ({

chaine : tab Chaines [ i ]

}) + *”***</td>***”*

+ *”***<td>***”* + r e g e x U t i l . test Regex Latin 1 With Digits Punctuation ({

chaine : tab Chaines [ i ]

}) + *”***</td>***”*

}

codeHTML += *”***</tbody></table>***”* ;

*// U t i l i s a t i o n de ”myApp. view . codeHTML” pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ; *// f i n de myApp. mainFunction ()*

*// Lancement de l ’ ap p l i cat i on :*

myApp. mainFunction ( ) ;

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

Code Source 2.16 : Fichiers *JS* inclus dans ex05-modulePatternRegexTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” ex 04 \_ s t ructure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . /ex05−modulePattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de l a mé thode de t e s t myApp. metier . mainFunction −−***>**

**<s cript src=***” . /ex 05 −modulePattern Regex Test . j s ”***></s cript>**

1

2

3

4

5

6

* 1. **Modélisation de Modules Métier (version 1)**

Nous voyons maintenant un utilitaire général permettant d’ajouter dans myApp.metier un module pour représenter ce qu’on appellerait dans les langages objets classiques une *classe métier*. Nous utiliserons dans la suite de ce cours les conventions et le vocabulaire suivant :

Un module métier permet de représenter des entités relevant du métier qui possèdent (regroupent) des attributs.

*•*

**Exemple.** Un module métier myApp.metier.adresse regroupera les attributs id, numeroRue, rue, complementAddr, codePostal, ville et pays...

Chaque module métier comprendra des *méthodes d’instances*, qui s’appliqueront aux instances, et dont l’implémentation s’appuiera sur les valeurs des attributs.

*•*

**Exemple.** La méthode d’instance getAttribute, appelée *accesseur*, permet d’obtenir la valeur d’un attribut dans une instance à partir du nom de l’attribut.

Chaque module métier comprendra des *méthodes de classe*, dont l’implémentation et le résultat seront indépendant des valeurs des attributs, donc indépendante des instances. **Exemple.** La méthode de classe getAttributeList renvoie, sous forme d’Array, la liste des noms d’attributs du module métier (liste commune à toutes les instances d’un même module métier).

*•*

Chaque module métier est construit à partir d’un objet attributesPatterns, qui définit la structure d’une instance. Ici, attributesPatterns définit, pour chaque attribut, une mé- thode de test par expression régulière de validité d’une valeur pour l’attribut, et une propriété labelText à afficher pour indiquer à l’utilisateur de quelle donnée il s’agit (typiquement : texte de l’élément *HTML* <label> associé à un input dans un formulaire). On pourrait facilement adapter le code pour permettre des propriétés calculées, ou encore différents types d’éléments d’interface utilisateur (divers *inputs* (couleur, nombre, date,...), de *textarea* ou de *select*) dans les formulaires, etc.

* + 1. **Attributs et méthodes statiques (Version 1)**

L’interface publique du module métier propose quelques méthodes statiques utilitaires, comme l’accès à la liste des noms d’attributs, aux données labelText, ou le test d’expression régulière d’une valeur pour un attribut.

Code Source 2.17 : /pattern-fonct/ex06-createModuleMetier.js

1. */\*\* @ description Dé f i n i t l e s propri é t é s géné ral e des ob j e t s mé t i e r s*
2. *\* à p ar t i r d ’ une sp é c i f i c a t i o n des a t t r i b u t s .*
3. *\* On ajoute au mé t i e r un ob j e t qui es t l ’ i n t er f ace publique d ’ une fonction qui s u i t l e pattern ” module ” .*
4. *\* La fonction retourne son i n t er f ace publique qui es t un ob j e t .*
5. *\* Cet ob j e t es t ajout é comme sous module au module ” metier ” .*

−

6 *\**

1. *\* Dans cet objet , on ne t rouve pas pour l e moment l e s propri é t é s d ’ instance .*
2. *\* Celles −ci seront ajout é es par ” augmentation ” .*
3. *\* @module create Module Metier*
4. *\* @augments myApp. metier*
5. *\* @param { s t r i n g } moduleName nom du module mé t i e r*

−

1. *\* @param { Object } a t t r i b u t es Pat t er n s ob j e t contenant l e s a t t r i b u t s d ’ un module mé t i e r*

−

13 *\**

du module mé t i e−r

Chaque propri é t é de a t t r i b u t es Pat t er n s dé f i n i t un a t t r i b u t

1. *\* chaque a t t r i b u t possède*

−

1. *\* + une fonction de t e s t sur l e s valeurs nommée regex Test*
2. *\* + un t ex t e de l a b e l permettant de dé s i gner l a donnée pour l ’ u t i l i s a t e u r*

17 *\*/*

1. myApp. addModule . apply (myApp. metier , [ *” create Module Metier ”* ,
2. **function** ( moduleName , a t t r i b u te s P a t te r n s ) { 20

21 ////////////////////////////////////////////////

22 // Propri é t é s et mé thodes ” s t a t i q u e s ” priv é es

23

24 */\*\**

1. *\* Tableau contenant l a l i s t e des a t t r i b u t s d ’ une instance .*
2. *\* Le t ableau es t pr é cal cu l é l or s de l ’ i n i t i a l i s a t i o n .*
3. *\* @member*
4. *\* @ private*

29 *\*/*

1. **var** a t t r i b u t e L i s t = **function** () {
2. **var** l i s t e = [ ] ; 32
3. *// Parcours des propri é t é s de l ’ ob j e t a t t r i b u t es Pat t er n s . regex Test*
4. *// qui correspondent aux a t t r i b u t s de l ’ instance*
5. **for** ( **var** attribute Name **in** a t t r i b u te s P a t te r n s ) {
6. *// Ne pasconsid é rer l e s propri é t é s ” hé r i t é es ” du prototype .*
7. **i f** ( a t t r i b u te s P a t te r n s . hasOwnProperty ( attribute Name ) ) {
8. l i s t e . push ( attribute Name ) ; 39 }

40 }

41

1. **return** l i s t e ;
2. }() ; *// appel immé d i at de l a fonction anonyme .*

44

45

46 ////////////////////////////////////////////////

47 // In t er f ace publique du module

48

1. */\*\* @ description Objet contenant l e s donné es et mé thodes pu b l i q u es*
2. *\* ( l e s propri é t é s pu b l i q u es sont retourn é es par l a fonction ” module ”) .*

51 *\*/*

52 **var** p u b l i c In te r f a c e M o d u l e s M e t i e r = { 53

1. */\*\* @ description Renvoie l a l i s t e des noms d ’ a t t r i b u t s s des instances .*
2. *\* @method g e t A t t r i b u t e L i s t*

56 *\*/*

1. g e t A ttr i b u te L i s t : **function** () {
2. **return** a t t r i b u t e L i s t ; 59 } ,

60

1. */\*\* @ description Renvoie l e t ex t e de d es cr i p t i on d ’ un a t t r i b u t des instances*

.

1. *\* @method get Label Text*
2. *\* @param { s t r i n g } attribute Name nom de propri é t é*

−

1. *\* @return { s t r i n g } l e t ex t e de d es cr i p t i on courte du champs*

65 *\*/*

1. get Label Text : **function** ( attribute Name ) {
2. **return** a t t r i b u te s P a t te r n s [ attribute Name ] . l a b e l Te xt ; 68 } ,

69

1. */\*\* @ description Expose l e t e s t d ’ expression r é g u l i è r e des a t t r i b u t s*
2. *\* des instances .*
3. *\* Peut ê t re u t i l i s ée pour l e f i l t r a g e des donné es d ’ un formulaire .*
4. *\* @method t est Regex*
5. *\* @param { s t r i n g } attribute Name nom de propri é t é*

−

1. *\* @param { s t r i n g } value valeur pour i n i t i a l i s e r l ’ a t t r i b u t*

−

1. *\* @return { boolean | s t r i n g } t rue s i l a chaine es t un code p os t a l valide ,*
2. *\* un message d ’ erreur sinon .*

78 *\*/*

1. test Regex : **function** ( attribute Name , value ) {
2. **i f** ( a t t r i b u te s P a t te r n s [ attribute Name ] === undefined ) {
3. **return** *”La propri é t é ”* + attribute Name + *” n ’ ex i s t e pas”* ;
4. } **else** {
5. **return** a t t r i b u te s P a t te r n s [ attribute Name ] . regex Test ( value ) ; 84 }

85 }

86

87 } ; // f i n de l ’ ob j e t public Interface Modules Metier

88

89

myApp. metier [ moduleName ] = p u b l i c In te r f a c e M o d u l e s M e t i e r ;

}] *// f i n de*

) ; *// f i n de*

*// ( ajout*

*l a fonction qui cr ée l ’ ob j e t myApp. metier [ moduleName ]*

*l ’ appel ” apply ” de l a mé thode myApp. addModule*

*create Module Metier )*

*de l ’ ob j e t public Interface Modules Metier au metier , sous l e nom*

90

91

92

93

94

* + 1. **Exemple : Module Métier adresse**

Nous créons (nous instancions) maintenant un sous-module myApp.metier.adresse.

Code Source 2.18 : /pattern-fonct/ex06-moduleMetierAdresse.js

1. */\*\* @ descripion Module mé t i e r myApp. metier . adresse*
2. *\* permettant de repr é senter des adresses p os t a l es .*

3 *\**

1. *\* @module*
2. *\* @ public*

6 *\**

1. *\* @ property { function ( s t r i n g )} g e t A t t r i b u t e permet d ’ obtenir l a valeur d ’ un a t t r i b u t à p ar t i r du nom de l ’ a t t r i b u t*

−

1. *\* @ property { function ( s t r i n g )} get Label Text permet d ’ obtenir l e t ex t e de l a b e l d ’ un a t t r i b u t à p ar t i r du nom de l ’ a t t r i b u t*

−

1. *\* @ property { function ( s t r ing , value )} t est Regex permet de t e s t e r l a v a l i d i t é d ’ une valeur pour un a t t r i b u t*

−

10 *\*/*

1. myApp. metier . create Module Metier ( *” adresse ”* ,
2. */\*\* @ descripion Spé c i f i e l e s a t t r i b u t s des ob j e t s de type adresse :*
3. *\* propri é t é s attendues , forme de ces donné es . . .*

14 *\**

15 *\* @ private*

16 *\**

1. *\* @ property { Object } id Propri é t é s de l ’ i d e n t i f i a n t unique de l ’ instance*

−

1. *\* @ property { Object } numéroRue Propri é t é s du numéro de l a rue*

−

1. *\* @ property { Object } rue Propri é t é s du nom de l a rue/ p l ace*

−

1. *\* @ property { Object } complementAddr Propri é t é s du compl ément Lieu d i t /Bâ t iment . . .*

−

1. *\* @ property { Object } code Postal Propri é t é s du code p os t a l*

−

1. *\* @ property { Object } v i l l e Propri é t é s du nom de l a v i l l e*

−

1. *\* @ property { Object } numéroRue Propri é t é s du nom du pays*

−

24 *\*/*

25 {

1. id : {
2. regex Test : **function** ( chaine ) {

28 **i f** ( */^[0 9a f ]{ 10 }$/* i . t e s t ( chaine ) === **true** ) {

− −

1. **return true** ;
2. } **else** {
3. **return** *”L ’ i d e n t i f i a n t d o i t comporter 10 c h i f f r e s hexa . ”* ; 32 }

33 } ,

34 l a b e l Te xt : *” I d en t i f i a n t ”*

35 } ,

1. numeroRue : {
2. regex Test : **function** ( chaine ) {
3. **i f** (myApp. metier . r e g e x U t i l . test Regex Latin 1 With Digits ({

|  |  |
| --- | --- |
| 39 | chaine : chaine , |
| 40 | maxLength : 15 |
| 41 | }) === **true** ) |
| 42 | { |
| 43 | **return true** ; |
| 44  45  46  47 | } **else** {  **return** *” Le numéro de l a rue contient au p l us 15 caractères , ”*  +*” l e t t r e s , t i r e t s et g u i l l em e t s ou c h i f f r e s . ”* ;  } |
| 48 | } , |
| 49 | l a b e l Te xt : *”Numéro”* |
| 50 | } , |
| 51 | rue : { |
| 52 | regex Test : **function** ( chaine ) { |
| 53 | **i f** (myApp. metier . r e g e x U t i l . test Regex Latin 1 With Digits ({ |
| 54 | chaine : chaine , |
| 55 | minLength : 1 , |
| 56 | maxLength : 255 |
| 57 | }) === **true** ) |
| 58 | { |
| 59 | **return true** ; |
| 60 | } **else** { |
| 61 | **return** *” l e nom de l a rue/ place , o b l i g a t o i r e ne contient que”* |
| 62 | + *” des l e t t r e s , t i r e t s et g u i l l em e t s ou c h i f f r e s . ”* ; |
| 63 | } |
| 64 | } , |
| 65 | l a b e l Te xt : *” rue/ p l ace ”* |
| 66 | } , |
| 67  68  69  70 | complementAddr : {  regex Test : **function** ( chaine ) {  **i f** (myApp. metier . r e g e x U t i l . test Regex Latin 1 With Digits Punctuation ({ chaine : chaine , |
| 71 | maxLength : 255 |
| 72 | }) === **true** ) |
| 73 | { |
| 74 | **return true** ; |
| 75 | } **else** { |
| 76 | **return** *” l e compl ément d ’ adresse ne contient que des l e t t r e s , ”* |
| 77 | + *” t i r e t s et g u i l l em e t s ou c h i f f r e s . ”* ; |
| 78 | } |
| 79 | } , |
| 80 | l a b e l Te xt : *” Lieu d i t , Bâtiment , BP”* |
| 81 | } , |
| 82 | code Postal : { |
| 83 | regex Test : **function** ( chaine ) { |
| 84  85 | **i f r** ( */^[ 0 −9]{5}$/* . t e s t ( chaine ) === **true** ) {  **eturn true** ; |
| 86 | } **else** { |
| 87 | **return** *” Le code p os t a l d o i t comporter 5 c h i f f r e s décimaux . ”* ; |
| 88 | } |
| 89 | } , |
| 90  91 | l a b e l Te xt : *” Code Postal ”*  } , |
| 92 | v i l l e : { |
| 93 | regex Test : **function** ( chaine ) { |
| 94 | **i f** (myApp. metier . r e g e x U t i l . test Regex Latin 1 ({ |

95

{

} **else** {

chaine : chaine , minLength : 1 , maxLength : 255

}) === **true** )

**return true** ;

**return** *”La v i l l e , ob l i g a t o i r e , ne contient que des l e t t r e s , ”*

+ *” t i r e t s et g u i l l em e t s . ”* ;

} ,

}

pays : {

regex Test : **function** ( chaine ) {

**i f** (myApp. metier . r e g e x U t i l . test Regex Latin 1 ({ chaine : chaine ,

minLength : 1 , maxLength : 255

}) === **true** )

{

**return true** ;

} **else** {

**return** *” Le pays , ob l i g a t o i r e , ne contient que des l e t t r e s , ”*

+ *” t i r e t s et g u i l l em e t s . ”* ;

}

} ,

l a b e l Te xt : *” Pays”*

}

} *// f i n de l ’ ob j e t a t t r i b u t es Pat t er n s*

) ; *// f i n de l ’ invocation de myApp. metier . create Module Metier*

} ,

l a b e l Te xt : *” V i l l e ”*

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

* + 1. **Fabrique Générique d’Instances Métier (Version 1)**

Nous créons ensuite, via un *pattern* Module, une fabrique *générique* (ou encore *fabrique abs- traite*) d’instances d’objets métier. Cette fabrique est générique en ce sens qu’elle peut servir de fabrique pour n’importe quel module métier qui implémente les méthodes getAttributeList(), getLabelText() et testRegex(), comme les modules métier généré en utilisant le code source [2.17](#_bookmark55), comme illustré dans le code source [2.18](#_bookmark57). Nous verrons par exemple, dans la partie [2.6.4](#_bookmark59), com- ment utiliser cette méthode pour créer très simplement une *fabrique concrète* d’adresses en s’appuyant sur le module myApp.metier.adresse (voir la partie [2.6.2](#_bookmark56)).

La fabrique prend comme paramètre un objet contenant des valeurs pour initialiser les attributs, effectue les tests d’expressions régulières, et crée deux objets privés :

L’objet privé dataAttributes contient comme propriétés les attributs de l’instance d’ob- jets métier ;

*•*

L’objet privé dataError contient comme propriétés les éventuels messages d’erreur suite aux tests (typiquement d’expression régulière) sur la validité des valeurs des attributs de l’instance d’objets métier ;

*•*

Des méthodes publiques, dans l’interface du module, permettent d’accéder à, ou de modifier les données de l’instance. Notons que les méthodes de l’interface publique du module créé

peuvent utiliser les méthodes du module parent en utilisant le *pattern* that (voir la partie [2.1.3](#_bookmark38)). L’interface publique des instances expose aussi les méthodes statiques (méthodes de classes), dont le code ne dépend pas des instances, mais est factorisé au niveau du module métier (ici

donné par l’objet that suivant le *pattern* that).

Nous avons aussi ajouté la possibilité, en passant un argument inputObj égal à null, de réer une instance par défaut (id aléatoire et autres attributs vides) Ceci permet par exemple d’initialiser un formulaire vide pour créer une nouvelle instance.

Code Source 2.19 : /pattern-fonct/ex07-fabriqueObjetMetier.js

* 1. */\*\* @ description Fabrique qui cr ée des ob j e t s d ’ un module mé t i er , suivant l e ” pattern module ” .*
  2. *\* Le paramètre spec de notre fonction es t un ob j e t contenant l e s val eurs des a t t r i b u t s*
  3. *\* de l ’ instance à cr é er .*

4 *\**

1. *\* Pour ê t re u t i l i s ée , cet t e f ab r i q u e d o i t ê t re ajout ée en t ant que mé thode*
2. *\* à un module mé t i e r comportant dé j à dans son i n t er f ace publique l e s mé thodes*
3. *\* g e t A t t r i b u t e Lis t , get Label Text et t est Regex comme l e s modules*
4. *\* génér é s par l a mé thode myApp. Metier . create Module Metier ( . , . ) .*

9 *\**

1. *\* @method create Instance Generique*
2. *\* @augments myApp. metier*
3. *\* @param { Object } input Obj sp é c i f i c a t i o n des valeurs des a t t r i b u t s d ’ une instance de module mé t i e r*

−

1. *\* @param { s t r i n g } input Obj . id i d e n t i f i a n t unique de l ’ instance*

−

1. *\* @param { s t r i n g | number} input Obj . attribute Name valeur à a f f e c t e r à l ’ a t t r i b u t attribute Name*

−

1. *\* ( exemple pour une adresse : input Obj . numeroRue , input Obj . code Postal , etc .*

16 *\*/*

17 myApp. addModule . apply (myApp. metier , [ *” create Instance Generique ”* , **function** ( inputObj ) {

18

19 ////////////////////////////////////////////////

20 // Propri é t é s et mé thodes ” s t a t i q u e s ” priv é es

21

1. */\*\* @ description contient l e s valeurs des a t t r i b u t s de l ’ instance , i n i t i a l em en t v ide*
2. *\* @member*
3. *\* @ private*

25 *\*/*

1. **var** d a ta A ttr i b u te s = {} ;
2. */\*\* @ description contient l e s messages d ’ erreur associ é s aux a t t r i b u t s de l ’ instances*
3. *\* @member*
4. *\* @ private*

30 *\*/*

31 **var** data Error = {} ; 32

1. *// Application du pattern ” t hat ”*
2. *// Si une fonction l oc a l e (comme set Attribute And Error p l us l o i n )*
3. *// veut u t i l i s e r une mé thode du module contenant concrètement notre fabrique ,*
4. *// ( par exemple myApp. metier . adresse . t est Regex )*
5. *// I l d o i t appeler ( par exemple ) t h i s . set Attribute And Error .*
6. *// Mais s i l a mé thode set Attribute And Error es t appel ée dans*
7. *// l ’ i n t er f ace publique des instances ( p u b l i c In t er f ace In s t an ce ) ci dessous ,*

−

1. *// ” t h i s ” r é f é rera à cet autre ob j e t ( i c i p u b l i c In t er f ace In s t an ce ) .*
2. *// Une s o l u t i on con s i s t e à mé moriser l ’ ob j e t t h i s dans une v ar i ab l e that ,*
3. *// et d ’ appeler l a mé thode t hat . t est Regex et non pas t h i s . t est Regex*
4. *// dans l a fonction l oc a l e set Attribute And Error ci dessous .*

−

1. **var** that = **this** ; *// exemple : t h i s = myApp. metier . adresse*

45

1. */\*\* @ description Ajoute un message d ’ erreur associ é à un a t t r i b u t*
2. *\* @method add Error*
3. *\* @ private*

49 *\*/*

1. **var** add Error = **function** ( attribute Name , message ) {
2. *// Ajout d ’ une propri é t é*
3. data Error [ attribute Name ] = message ; 53 }

54

1. */\*\* @ description S e t t er : i n i t i a l i s e l a valeur pour un a t t r i b u t d ’ une instance .*
2. *\* En cas d ’ erreur un message pour cet a t t r i b u t es t es t ajout é dans data Error .*
3. *\* En l ’ absence d ’ erreur , une é v en t u e l l e erreur pr écé dente es t e f f ac ée .*
4. *\* @method input Obj .*
5. *\* @ private*

60 *\*/*

1. **var** set Attribute And Error = **function** ( attribute Name , value ) {
2. **var** result Test Regex = that . test Regex ( attribute Name , value ) ;
3. *// On i n t i a l i s e l ’ a t t r i b u t de l ’ instance*
4. d a ta A ttr i b u te s [ attribute Name ] = value ;
5. *// Si l a v a l i d a t i on par expression r é g u l i è r e es t pass ée*
6. **i f** ( result Test Regex === **true** ) {
7. *// On e f f ace une v i e i l l e erreur é v en t u e l l e*
8. **delete** data Error [ attribute Name ] ;
9. } **else** {
10. *// On i n i t a l i s e l ’ a t t r i b u t de l ’ ob j e t des erreurs .*
11. *// avec l e message d ’ erreur .*
12. add Error ( attribute Name , *” A t t r i b u t ”* + value +
13. *” i n v a l i d e : ”* + result Test Regex ) ; 74 }

75 }

76

1. *// I n i t i a l i s a t i o n des a t t r i b u t s de l ’ instance .*
2. *// Si l ’ ob j e t en argument es t null , on con s t ru i t une instance par dé f aut*
3. **i f** ( inputObj !== **null** ) {
4. *// Parcours des propri é t é s de g e t A t t r i b u t e L i s t ()*
5. *// qui correspondent aux a t t r i b u t s de l ’ instance à cr é er*
6. **for** ( **var** i = 0 ; i **< this** . g e t A ttr i b u te L i s t () . l e n g th ; ++i ) {
7. **var** attribute Name = **this** . g e t A ttr i b u te L i s t ( ) [ i ] ;
8. *// U t i l i s a t i o n du s e t t e r pour i n i t i a l i s e r l ’ a t t r i b u t*
9. set Attribute And Error ( attribute Name , inputObj [ attribute Name ] ) ; 86 }
10. } **else** {
11. *// On i n i t i a l i s e l e s valeu r s des a t t r i b u t s à ””*
12. **for** ( **var** i = 0 ; i **< this** . g e t A ttr i b u te L i s t () . l e n g th ; ++i ) {
13. **var** attribute Name = **this** . g e t A ttr i b u te L i s t ( ) [ i ] ;
14. d a ta A ttr i b u te s [ attribute Name ] = *””* ; 92 }

93 }

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

////////////////////////////////////////////////

// In t er f ace publique du module

/\*\* @ description In t er f ace publique des instances du module mé t i e r .

\* Fournit l e s mé thodes pour manipuler l ’ instance ( accesseurs , s e t t e r s . . . )

\*/

**var** p u b l i c I n t e r f a c e I n s t a n c e = {

/\*\* @ description Retourne l e module avec l e s mé thodes ” s t a t i q u e s ”

* *(comme l ’ accès d i r ect à l a l i s t e des propri é t é s ou l e s t e s t s regex )*
* *@return { Object } l e module myApp. metier . moduleName*

\*/

getModule : **function** ( ) {

**return** that ;

} ,

/\*\* @ description Accesseur pour tous l e s membres priv é s d ’ instance .

* *@method g e t A t t r i b u t e*
* *@ public*
* *@param { s t r i n g } attribute Name nom de l ’ a t t r i b u t attendue d ’ une instance*

−

* *@return { s t r i n g } l a valeur de l ’ a t t r i b u t ou undefined en cas de nom d ’ a t t r i b u t inconnu .*

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

*\*/*

g e t A ttr i b u te : **function** ( attribute Name ) {

**return** d a ta A ttr i b u te s [ attribute Name ] ;

} ,

/\*\* @ description S e t t er :

* *i n i t i a l i s e l a valeur pour un a t t r i b u t d ’ une instance après un t e s t .*
* *En cas d ’ erreur , un message pour cet a t t r i b u t es t ajout é dans data Error .*
* *En l ’ absence d ’ erreur , une é v en t u e l l e erreur pr écé dente es t e f f ac ée .*
* *I l s ’ ag i t d ’ un s imple a l i a s vers l a fonction priv ée set Attribute And Error .*
* *@method s e t A t t r i b u t e*
* *@ public*
* *@param { s t r i n g } attribute Name nom de l ’ a t t r i b u t attendue d ’ une instance*

−

* *@param { s t r i n g } value valeur à prendre pour l ’ a t t r i b u t attendu d ’ une instance*

−

* *@return { boolean } t rue s ’ i l y a au moins une erreur , f a l s e sinon*

\*/

s e t A ttr i b u te : set Attribute And Error ,

/\*\* @ description permet de savoir s i un des a t t r i b u t s au moins comporte une erreur .

\* @return { boolean } t rue s ’ i l y a au moins une erreur , f a l s e sinon

\*/

has Error : **function** () {

**for** ( **var** attribute Name **in** data Error ) {

**i f** ( data Error . hasOwnProperty ( attribute Name ) ) {

**return true** ;

}

}

**return f al s e** ;

} ,

/\*\* @ description Donne l ’ accès au message d ’ erreur d ’ un a t t r i b u t ( s ’ i l ex i s t e

) .

|  |  |  |
| --- | --- | --- |
| 146 | *\* @param { s t r i n g } attribute Name* | *− nom d ’ a t t r i b u t d ’ une instance de module mé* |
| 147 | *\* @return { s t r i n g | undefined } l e* | *message d ’ erreur pour un a t t r i b u t s ’ i l* |
| 148 | *\* ou undefined en l ’ absence d ’ erreur* | |
| 149 | *\*/* | |
| 150  151  152 | get Error Message : **function** ( attribute Name ) {  **return** data Error [ attribute Name ] ;  } , | |
| 153 |  | |
| 154 | */\*\* @ description Ré cupère l a l i s t e des noms d ’ a t t r i b u t s qui ont une erreur* | |
| 155 | *\* @return { s t r i n g [ ] } t ab l eau des noms d ’ a t t r i b u t s qui comportent une erreur* | |
| 156 | *\*/* | |
| 157 | g e t E r r o r L i s t : **function** () { | |
| 158 | **var** e r r o r L i s t = [ ] ; | |
| 159 | **for** ( **var** attribute Name **in** data Error ) { | |
| 160 | **i f** ( data Error . hasOwnProperty ( attribute Name ) ) { | |
| 161 | e r r o r L i s t . push ( attribute Name ) ; | |
| 162 | } | |
| 163 | } | |
| 164 | **return** e r r o r L i s t ; | |
| 165 | } , | |
| 166  167  168  169  171  172  173  174  175  176  178  179  180  181  182  183  184  185  187  188  189  190  191  192  193  194  195  196  197 | *////////////////////////////////////////////////////////////////////////*  *// Ajout des mé thodes s t a t i q u e s du module dans l ’ i n t er f ace des instances*  *////////////////////////////////////////////////////////////////////////*  */\*\* @ description Renvoie l a l i s t e des noms d ’ a t t r i b u t s des instances .*  *\* @method g e t A t t r i b u t e Li s t*  *\*/*  g e t A ttr i b u te L i s t : **function** () {  **return** that . g e t A ttr i b u te L i s t () ;  } ,  */\*\* @ description Renvoie l e t ex t e de d es cr i p t i on d ’ un a t t r i b u t des instances*  *.*   * *@method get Label Text* * *@param { s t r i n g } attribute Name − nom d ’ un a t t r i b u t* * *@return { s t r i n g } l e t ex t e de d es cr i p t i on courte du champ*   *\*/*  get Label Text : **function** ( attribute Name ) {  **return** that . get Label Text ( attribute Name ) ;  } ,  */\*\* @ description Expose l e t e s t d ’ expression r é g u l i è r e des a t t r i b u t s*   * *des instances .* * *Peut ê t re u t i l i s ée pour l e f i l t r a g e des donné es d ’ un formulaire .* * *@method t est Regex* * *@param { s t r i n g } attribute Name − nom d ’ un a t t r i b u t* * *@param { s t r i n g } value − valeur pour i n i t i a l i s e r l ’ a t t r i b u t* * *@return { boolean | s t r i n g } t rue s i l a chaine es t un code p os t a l valide ,* * *un message d ’ erreur sinon .*   *\*/*  test Regex : **function** ( attribute Name , value ) {  **return** that . test Regex ( attribute Name , value ) ; | |

t i e r

ex i s t e

.

170

177

186

198

199

200

201

202

203

204

205

206

} ,

/\*\* @ description Fabrique qui cr ée des ob j e t s d ’ un module mé t i er , suivant l e ” pattern module ” .

* *Le paramètre spec de notre fonction es t un ob j e t contenant l e s valeurs des a t t r i b u t s*
* *de l ’ instance à cr é er .*
* *@param { Object } input Objet sp é c i f i c a t i o n des valeurs des a t t r i b u t s d ’ une instance de module mé t i e r*

−

* *@param { s t r i n g } input Objet . id i d e n t i f i a n t unique de l ’ instance*

−

* *@param { s t r i n g } input Objet . attribute Name valeur à a f f e c t e r à l ’ a t t r i b u t attribute Name*

−

* *( exemple pour une adresse : input Obj . numeroRue , input Obj*

. code Postal , etc .

207

208

209

210

211

212

213

*\*/*

c r e a te I n s ta n c e : **function** ( input Objet ) {

**return** that . c r e a te I n s ta n c e ( input Objet ) ;

} ,

/\*\* @ description Crée une copie de l ’ ob j e t à l ’ i d en t i q u e .

\* @ returns { Object } une instance avec des valeurs d ’ a t t r i b u t s i den t i q u e à t h i s .

214

215

216

217

218

219

220

221

222

223

*\*/*

c l o n e : **function** () {

**return this** . c r e a te I n s ta n c e ( d a ta A ttr i b u te s ) ;

}

} ; // f i n de p u b l i c In t er f ace In s t an ce

**return** p u b l i c I n t e r f a c e I n s t a n c e ;

} // f i n de l a mé thode create Instance

] ) ; // f i n de l ’ appel ” apply ” de l a mé thode myApp. addModule

### Exemple : La Fabrique du Module adresse

##### La fabrique d’instances d’adresse, comme toutes les autres fabriques concrètes pour les instances d’objets métier, est simplement une copie de la fabrique générique d’objets métier vue dans la partie [2.6.3](#_bookmark58), qui est simplement placée dans le sous-module myApp.metier.adresse.

Ainsi, mors de l’invocation de cette fabrique, les méthodes getAttributeList(), getLabelText()

et that.testRegex() font référence aux méthodes éponymes du module myApp.metier.adresse, ce qui permet à la fabrique, dont le code est générique, de travailler cincrètement sur des adresses.

Code Source 2.20 : /pattern-fonct/ex08-fabriqueAdresse.js

1

*/\*\**

* *Fabrique qui cr ée des ob j e t s repr é sentant des adresse , suivant l e ” pattern module ” .*
* *Le paramètre spec de notre fonction es t un ob j e t contenant l e s propri é t é s d ’ une adresse à cr é er .*

*\**

* *@method create Instance*
* *@augments myApp. metier . adresse*

*\* @param { Object } input Obj − sp é c i f i c a t i o n des propri é t é s d ’ une instance d ’*

*adresse*

2

3

4

5

6

7

8

*\* @param { s t r i n g } input Obj . id − i d e n t i f i a n t unique de l ’ instance*

*\* @param { s t r i n g } input Obj . numeroRue − numero de rue*

*\* @param { s t r i n g } input Obj . rue − nom de rue*

*\* @param { s t r i n g } input Obj . complementAddr − compl ément d ’ adresse ( l i eu dut , bâ*

*\* @param { s t r i n g } input Obj . code Postal − code p os t a l*

*timent , r é sidence , etc . )*

*\* @param { s t r i n g } input Obj . v i l l e − nom de v i l l e*

*\* @param { s t r i n g } input Obj . pays − nom de pays*

myApp. addModule . apply (myApp. metier . adresse , [ *” create Instance ”* ,

myApp. metier .

c r e a te In s ta n c e G e n e r i q u e ] ) ; *//*

*f i n de l ’ appel ” apply ” de l a mé thode myApp. addModule*

*\*/*

9

10

11

12

13

14

15

16

17

Le fichier *HTML* réalise le test de création d’une instance et d’utilisation de *setters*, et affiche les données et les erreurs obtenues.

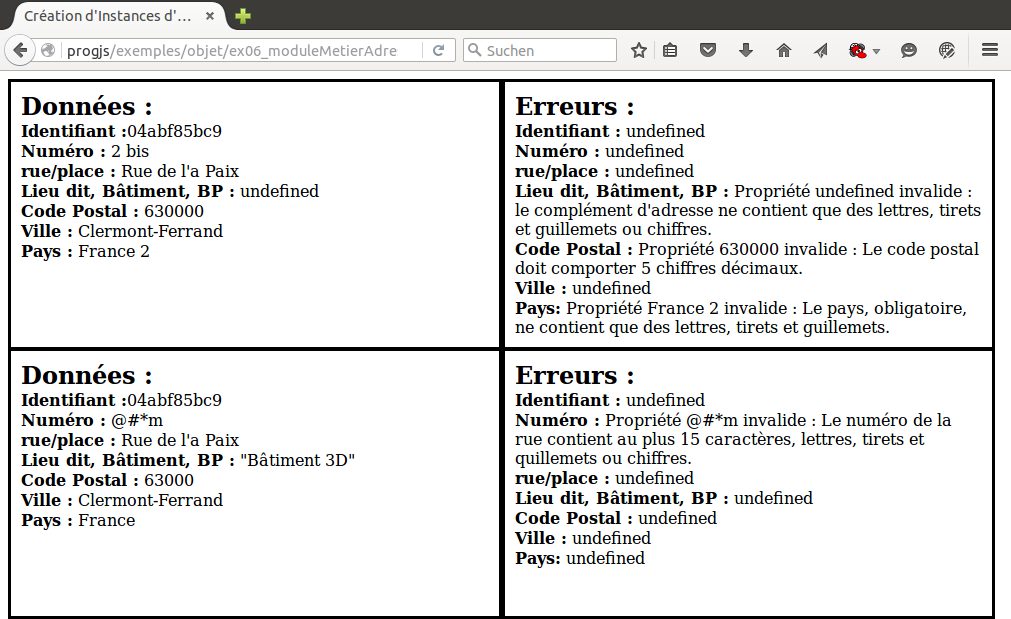


Figure 2.6 : Illustration du code source [2.21](#_bookmark61)

Code Source 2.21 : /pattern-fonct/ex06-moduleMetierAdresseTest.js (cf. Fig [2.6](#_bookmark60))

1

*/\*\* Module de t e s t avec une mé thode d ’ a f f i ch ag e d ’ une instance*

* *d ’ adresse avec ses é v en t u e l l e s erreurs .*
* *@module myApp. t e s t*
* *@method myApp. t e s t . t est Affiche Adresse*
* *@ public*

*\* @param { Object } adresse − instance d ’ ob j e t mé t i e r du module ” adresse ”*

myApp. addModule ( *” t e s t ”* , {

te s t A f f i c h e A d r e s s e : **function** ( a d r e s s e ) {

*// Code HTML pour l ’ a f f i ch ag e de l ’ instance :*

**var** codeHTML = *”***<div><span><h2>***Donné es :***</h2>***”* +

*”***<s trong>***”* + a d r e s s e . get Label Text ( *’ id ’* ) +

*” :***</s trong>***”* + a d r e s s e . g e t A ttr i b u te ( *’ id ’* ) + *”***<br** */***>***”* +

*”***<s trong>***”* + a d r e s s e . get Label Text ( *’ numeroRue ’* ) +

*” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ numeroRue ’* ) + *”***<br** */***>***”* +

*”***<s trong>***”* + a d r e s s e . get Label Text ( *’ rue ’* ) +

*\*/*

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

1. *” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ rue ’* ) + *”***<br** */***>***”* +
2. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ complementAddr ’* ) +
3. *” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) + *”***<br** */***>***”* +
4. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ code Postal ’* ) +
5. *” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ code Postal ’* ) + *”***<br** */***>***”* +
6. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ v i l l e ’* ) +
7. *” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ v i l l e ’* ) + *”***<br** */***>***”* +
8. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ pays ’* ) +
9. *” :***</s trong>** *”* + a d r e s s e . g e t A ttr i b u te ( *’ pays ’* ) + 26 *”***</span>***”* ;

27

1. *// variante en énumé rant automatiquement l e s propri é t é s*
2. codeHTML += *”***<span><h2>***E rreurs :***</h2>***”* +
3. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ id ’* ) +
4. *” :***</s trong>***”* + a d r e s s e . get Error Message ( *’ id ’* ) + *”***<br** */***>***”* +
5. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ numeroRue ’* ) +
6. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ numeroRue ’* ) + *”***<br** */***>***”* +
7. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ rue ’* ) +
8. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ rue ’* ) + *”***<br** */***>***”* +
9. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ complementAddr ’* ) +
10. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ complementAddr ’* ) + *”***<br** */***>***”* +
11. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ code Postal ’* ) +
12. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ code Postal ’* ) + *”***<br** */***>***”* +
13. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ v i l l e ’* ) +
14. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ v i l l e ’* ) + *”***<br** */***>***”* +
15. *”***<s trong>***”* + a d r e s s e . get Label Text ( *’ pays ’* ) +
16. *” :***</s trong>** *”* + a d r e s s e . get Error Message ( *’ pays ’* ) + 44 *”***</span></div>***”* ;
17. **return** codeHTML ;
18. } *// f i n de l a mé thode t est Affiche Adresse*
19. } *// f i n du module myApp. t e s t*
20. ) ; *// f i n de l ’ invocation de myApp. addModule*

49

1. */\*\* @ description Programme p r i n c i p a l qui con s t ru i t l e s donné es*
2. *\* et gé nère l a vue .*
3. *\* @method myApp. mainFunction*
4. *\* @ public*

54 *\*/*

1. myApp. addModule ( *” mainFunction”* , **function** () {
2. *// cr é ation d ’ une instance*
3. **var** a d r e s s e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 58 id : *” 04 abf 85 bc 9 ”* ,
4. numeroRue : *” 2 b i s ”* ,
5. rue : *”Rue de l ’ a Paix ”* ,
6. *// ou b l i du champs complementAddr*
7. code Postal : *” 630000 ”* ,
8. v i l l e : *” Clermont Ferrand”* ,

−

1. pays : *” France 2 ”*

65 }) ;

66

67 **var** codeHTML = **this** . t e s t . te s t A f f i c h e A d r e s s e ( a d r e s s e ) ; 68

1. *// Test de s e t t e r*
2. a d r e s s e . s e t A ttr i b u te ( *” complementAddr”* , *”\”Bâtiment 3D\””* ) ;
3. a d r e s s e . s e t A ttr i b u te ( *” code Postal ”* , *” 63000 ”* ) ;
4. a d r e s s e . s e t A ttr i b u te ( *” pays ”* , *” France”* ) ;

73

a d r e s s e . s e t A ttr i b u te ( *” numeroRue”* , *”@#\*m”* ) ; codeHTML += **this** . t e s t . te s t A f f i c h e A d r e s s e ( a d r e s s e ) ;

*// Test de clonage :*

codeHTML += **this** . t e s t . te s t A f f i c h e A d r e s s e ( a d r e s s e . c l o n e ( ) ) ;

*// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ;

*// Exé cution de l a mé thode mainFunction*

myApp. mainFunction ( ) ;

74

75

76

77

78

79

80

81

82

83

84

85

Code Source 2.22 : Fichiers *JS* inclus dans ex06-moduleMetierAdresseTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . /ex05−modulePattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −createModuleMetier . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . / ex 07 −fabrique Objet Metier . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −moduleMetierAdresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**<s cript src=***” . / ex 08 −fabrique Adresse . j s ”***></s cript>**

**<s cript src=***” . /ex 06 −moduleMetier AdresseTest . j s ”***></s cript>**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

* + 1. **Utilisation : Création d’un Module myApp.view.adresse**

Nous ajoutons, dans un module myApp.view.adresse, des méthodes pour générer le code *HTML* d’une adresse, au format compact (sur une ligne) ou au format développé (avec le détail des labels des attribut).

Code Source 2.23 : /pattern-fonct/ex09-adresseView.js

1

myApp. addModule ( *” view ”* , { a d r e s s e : {}}) ;

*// Cré ation d ’ un module myApp. view et d ’ un sous−module myApp. view . adresse*

*/\*\**

* *Mé thode de géné ration de code HTML pour une instance d ’ adresse .*
* *Pour chaque propri é t é attendue d ’ une adresse , l a d es cr i p t i on de l a propri é t é et sa valeur sont a f f i c h é es .*

*\**

* *@method get Html Developped*
* *@augments myApp. view . adresse*

*\* @param { Object } adresse − sp é c i f i c a t i o n des propri é t é s d ’ une instance d ’*

*\* @param { s t r i n g } adresse . id − i d e n t i f i a n t unique de l ’ instance*

*adresse*

*\* @param { s t r i n g } adresse . numeroRue − numero de rue*

*\* @param { s t r i n g } adresse . rue − nom de rue*

2

3

4

5

6

7

8

9

10

11

12

13

1. *\* @param { s t r i n g } adresse . complementAddr compl ément d ’ adresse ( l i eu dut , bâ timent , r é sidence , etc . )*

−

1. *\* @param { s t r i n g } adresse . code Postal code p os t a l*

−

1. *\* @param { s t r i n g } adresse . v i l l e nom de v i l l e*

−

1. *\* @param { s t r i n g } adresse . pays nom de pays*

−

18 *\*/*

1. myApp. addModule . apply (myApp. view . adresse , [ *” get Html Developped ”* , **function** ( a d r e s s e

) {

1. **var** htmlCode = *””* ; 21

22 **var** module Adresse = a d r e s s e . getModule ( ) ; 23

1. **i f** ( a d r e s s e . g e t A ttr i b u te ( *’ numeroRue ’* ) ) {
2. htmlCode += *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ numeroRue ’* ) + *”&nbsp ; :***</span>** *”* +
3. a d r e s s e . g e t A ttr i b u te ( *’ numeroRue ’* ) + *” ,***<br** */***>***”* ; 27 }

28

1. htmlCode += *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ rue ’* )

+ *”&nbsp ; :***</span>** *”* +

1. a d r e s s e . g e t A ttr i b u te ( *’ rue ’* ) + *” ,***<br** */***>***”* ; 31
2. **i f** ( **typeof** a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) === *” s t r i n g ”* &&
3. a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) !== *””* ) {
4. htmlCode += *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ complementAddr ’* ) + *”&nbsp ; :***</span>** *”* +
5. a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) + *” ,***<br** */***>***”* ; 36 }

37

1. htmlCode += *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ code Postal ’* ) + *”&nbsp ; :***</span>** *”* +
2. a d r e s s e . g e t A ttr i b u te ( *’ code Postal ’* ) + *”***<br** */***>***”* +
3. *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ v i l l e ’* ) + *”& nbsp ; :***</span>** *”* +
4. a d r e s s e . g e t A ttr i b u te ( *’ v i l l e ’* ) + *”***<br** */***>***”* +
5. *”***<span** *c l as s =\” adresse Item \”***>***”* + module Adresse . get Label Text ( *’ pays ’* ) + *”& nbsp ; :***</span>** *”* +
6. a d r e s s e . g e t A ttr i b u te ( *’ pays ’* ) + *”***<br** */***>***”* ; 44
7. **i f** ( a d r e s s e . has Error ( ) ) {
8. **var** e r r o r L i s t = a d r e s s e . g e t E r r o r L i s t () ;
9. htmlCode += *”***<s trong>***C ertains champs ont une erreur :***</s trong><br** */***>***”* ;
10. **for** ( **var** i =0 ; i**<**e r r o r L i s t . l e n g th ; i ++){ 49 **i f** ( i **>** 0 ) {

50 htmlCode += *” , ”* ; 51 }

52 htmlCode += e r r o r L i s t [ i ] ; 53 }

54 }

55

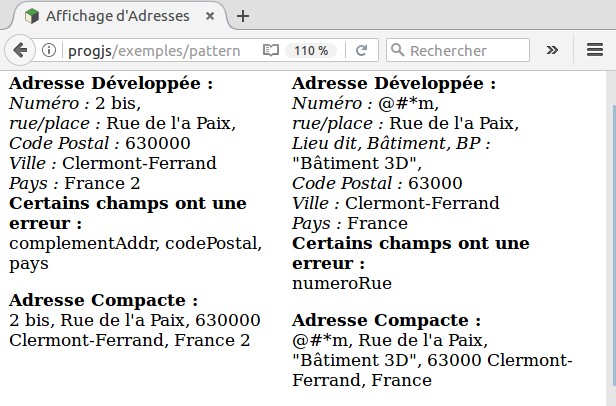
56 **return** htmlCode ; 57 } ] ) ;

58

59 */\*\**

1. *\* Mé thode de géné ration de code HTML pour une instance d ’ adresse .*
2. *\* L ’ adresse es t a f f i c h ée sur une l i gne , sans mention des erreurs .*

|  |  |
| --- | --- |
| 62 | *\** |
| 63 | *\* @method get Html Developped* |
| 64 | *\* @augments myApp. view . adresse* |
| 65 | *\* @param { Object } adresse − sp é c i f i c a t i o n des propri é t é s d ’ une instance d ’ adresse* |
| 66  67  68 | * *@param { s t r i n g } adresse . id − i d e n t i f i a n t unique de l ’ instance* * *@param { s t r i n g } adresse . numeroRue − numero de rue* * *@param { s t r i n g } adresse . rue − nom de rue* |
| 69 | *\* @param { s t r i n g } adresse . complementAddr − compl ément d ’ adresse ( l i eu dut , bâ*  *timent , r é sidence , etc . )* |
| 70  71  72  73 | * *@param { s t r i n g } adresse . code Postal − code p os t a l* * *@param { s t r i n g } adresse . v i l l e − nom de v i l l e* * *@param { s t r i n g } adresse . pays − nom de pays*   *\*/* |
| 74 | myApp. addModule . apply (myApp. view . adresse , [ *” getHtmlCompact”* , **function** ( a d r e s s e ) { |
| 75 | **var** htmlCode = *””* ; |
| 76 |  |
| 77 | **i f** ( a d r e s s e . g e t A ttr i b u te ( *’ numeroRue ’* ) ) { |
| 78 | htmlCode += a d r e s s e . g e t A ttr i b u te ( *’ numeroRue ’* ) + *” , ”* ; |
| 79 | } |
| 80 |  |
| 81 | htmlCode += a d r e s s e . g e t A ttr i b u te ( *’ rue ’* ) + *” , ”* ; |
| 82 | **i f** ( a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) ) { |
| 83 | htmlCode += a d r e s s e . g e t A ttr i b u te ( *’ complementAddr ’* ) + *” , ”* ; |
| 84 | } |
| 85 | htmlCode += a d r e s s e . g e t A ttr i b u te ( *’ code Postal ’* ) + *” ”* + |
| 86 | a d r e s s e . g e t A ttr i b u te ( *’ v i l l e ’* ) + *” , ”* + |
| 87  88  89 | a d r e s s e . g e t A ttr i b u te ( *’ pays ’* ) ;  **return** htmlCode ;  } ] ) ; |



##### Figure 2.7 : Illustration du code source [2.24](#_bookmark64)

Code Source 2.24 : /pattern-fonct/ex09-adresseViewTest.js (cf. Fig [2.7](#_bookmark63))

1

*/\*\* Module de t e s t avec une mé thode d ’ a f f i ch ag e d ’ une instance*

* *d ’ adresse avec l e s mé thodes du package myApp. view . adresse*
* *@module myApp. t e s t*
* *@method myApp. t e s t . t est Affiche Adresse*
* *@ public*

*\* @param { Object } adresse − instance d ’ ob j e t mé t i e r du module ” adresse ”*

*\*/*

2

3

4

5

6

7

* 1. myApp. addModule ( *” t e s t ”* , {
  2. te s t A f f i c h e A d r e s s e : **function** ( a d r e s s e ) {
  3. **return** *”***<span** *s t y l e =\”***width** *:260 px ;* **display** *: in l ine b l ock ;* **vertical align** *:*

− −

**top** *;\”***>***”* +

* 1. *”***<p><s trong>***A dresse Dé velopp ée :***</s trong><br** */***>***”* +
  2. myApp. view . a d r e s s e . getHtml Developped ( a d r e s s e ) + *”***</p>***”* +
  3. *”***<p><s trong>***A dresse Compacte :***</s trong><br** */***>***”* +
  4. myApp. view . a d r e s s e . getHtmlCompact ( a d r e s s e ) + *”***</p>***”* + 15 *”***</span>***”* ;

1. } *// f i n de l a mé thode t est Affiche Adresse*
2. } *// f i n du module myApp. t e s t*
3. ) ; *// f i n de l ’ invocation de myApp. addModule*

19

20

1. */\*\* @ description Programme p r i n c i p a l qui con s t ru i t l e s donné es*
2. *\* et gé nère l a vue .*
3. *\* @method myApp. mainFunction*
4. *\* @ public*

25 *\*/*

1. myApp. addModule ( *” mainFunction”* , **function** () {
2. *// cr é ation d ’ une instance*
3. **var** a d r e s s e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 29 id : *” 04 abf 85 bc 9 ”* ,
4. numeroRue : *” 2 b i s ”* ,
5. rue : *”Rue de l ’ a Paix ”* ,
6. *// ou b l i du champs complementAddr*
7. code Postal : *” 630000 ”* ,
8. v i l l e : *” Clermont Ferrand”* ,

−

1. pays : *” France 2 ”*

36 }) ;

37

38 **var** codeHTML = **this** . t e s t . te s t A f f i c h e A d r e s s e ( a d r e s s e ) ; 39

1. a d r e s s e . s e t A ttr i b u te ( *” complementAddr”* , *”\”Bâtiment 3D\””* ) ;
2. a d r e s s e . s e t A ttr i b u te ( *” code Postal ”* , *” 63000 ”* ) ;
3. a d r e s s e . s e t A ttr i b u te ( *” pays ”* , *” France”* ) ;
4. a d r e s s e . s e t A ttr i b u te ( *” numeroRue”* , *”@#\*m”* ) ; 44
5. codeHTML += **this** . t e s t . te s t A f f i c h e A d r e s s e ( a d r e s s e ) ;
6. *// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*
7. document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ; 48 }) ;

49

1. *// Exé cution de l a mé thode mainFunction*
2. myApp. mainFunction ( ) ;

Code Source 2.25 : Fichiers *JS* inclus dans ex09-adresseViewTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . /ex05−modulePattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −createModuleMetier . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . / ex 07 −fabrique Objet Metier . j s ”***></s cript>**

1

2

3

4

5

6

7

8

9

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −moduleMetierAdresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

**<s cript src=***” . / ex 08 −fabrique Adresse . j s ”***></s cript>**

**<***!−− Cré ation de f onct i ons d ’ a f f i ch ag e dans myApp. metier . view . adresse −−***>**

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**<s cript src=***” . /ex09−adresseView . j s ”***></s cript>**

**<s cript src=***” . /ex 09 −adresseView Test . j s ”***></s cript>**

10

11

12

13

14

15

16

17

* 1. **Interface Générique de Objets métier**

Il est possible de tester l’affichage des objets métier de manière générique, sans supposer qu’il s’agit particuièrement d’une adresse, sur la base de l’implémentation d’une interface. Nous devons d’abord créer un outil pour vérifier l’implémentation d’une interface par des objets *JavaScript*.

* + 1. **Implémentation d’interfaces en *JavaScript***

##### Voici une classe Interface, qui possède comme attribut un Array de noms de méthodes, et qui permet de vérifier qu’un objet possède bien des méthodes avec les noms correspondants. Notons que nous ne vérifions pas que lés méthodes correspondent bien à un prototype déterminé, mais seulement que les noms de méthodes sont présents.

Code Source 2.26 : /pattern-fonct/ex11-interfaceImplementation.js

1 */\*\**

* 1. *\* Dé f i n i t une ” i n t er f ace ” , avec un nom et un ensemble de mé thodes .*
  2. *\* Ceci nous permettra de vé r i f i e r qu ’ un certain nombre d ’ opé rations*
  3. *\* sont pr é sentes dans un ob j e t Java Script .*

5 *\**

1. *\* @ constructor In t er f ace*
2. *\* @param { s t r i n g } name nom de l ’ i n t er f ace*

−

1. *\* @param { s t r i n g [ ] } methods t ab l eau contenant l e s noms des mé thodes de l ’ i n t er f ace .*

−

9 *\*/*

10 **var** I n t e r f a c e = **function** ( methods ) { 11

1. **i f** ( methods . l e n g th === undefined ) {
2. **throw** {
3. name : *” I l l egal Argument ”* ,
4. message : *”Une i n t er f ace né c e s s i t e un array ( ou array l i k e ) de noms de mé thodes . ”*

−

16 } ;

17 }

18

1. *// Cré ation d ’ une propri é t é pour s t ocker l e nom de l ’ i n t er f ac*
2. *// Cré ation d ’ un t ab l eau pour socker l e s noms de mé thoded*
3. **this** . methods = [ ] ;
4. *// pour chaque nom de mé thode*
5. **for** ( **var** i = 0 ; i **<** methods . l e n g th ; ++i ) {
6. *// Vé r i f i c a t i o n de type*
7. **i f** ( **typeof** methods [ i ] !== *’ s t r i n g ’* ) {

26

27

28

29 } ;

30

**throw** {

name : *” I l l egal Argument ”* ,

message : ” Les noms de mé thodes d ’ une i n t er f ace doivent ê t re de type s t r i n g . ”

}

31

32

33

34 } ;

35

// Ajout du nom de mé thode

**this** . methods . push ( methods [ i ] ) ;

}

36 */\*\**

1. *\* Vé r i f i e qu ’ un ob j e t ” impl émente une i n t er f ace ” , en ce sens qu ’ i l comporte*
2. *\* un cert ain nombre de mé thodes ( propri é t é s de type fonction ) qui ont l e s*
3. *\* mêmes noms que l e s mé thodes de l ’ i n t er f ace .*

40 *\**

1. *\* @method is Implemented By*
2. *\* @param { Object } ob j e t ob j e t qui d o i t impl émenter l ’ i n t er f ace .*

−

1. *\* @return { boolean | s t r i n g } t rue s i l ’ ob j e t comporte t ou t es l e s mé thodes de l ’ in t erface ,*
2. *\* un message d ’ erreur indiquant une mé thode qui n ’ es t pas pr é sente dans l ’ ob j e t sinon .*

45 *\*/*

1. I n t e r f a c e . prototype . is Implemented By = **function** ( o b je t ) {
2. *// Pour chaque nom de mé thode*
3. **for** ( **var** i = 0 ; i **< this** . methods . l e n g th ; ++i ) {
4. **var** methodName = **this** . methods [ i ] ;
5. *// Si l ’ ob j e t n ’ a pas de propri é t é de ce nom qui s o i t de type fonction*
6. **i f** ( ! o b je t [ methodName ] | | **typeof** o b je t [ methodName ] !== *’ function ’* ) {
7. **return** *”L ’ ob j e t n ’ impl émente pas l a mé thode ”* + methodName ; 53 }

54 }

55 **return true** ; 56 } ;

##### Voici un exemple dans lequel nous définissions une interface attendues de nos modules métier. Cette interface contient deux types de méthodes :

1. L’interface attendue des instances du module métier permet de tester la présence d’un certain nombre de méthodes communes à toutes les instances (similaires à des méthodes de classe), qui sont définies dans le prototype des instances ;
2. L’interface attendue des instances permet de s’assurer de la présence d’un certain nombre de méthodes sur les instances, qui sont définies soit au niveau du prototype (si le code source est indépendant de l’instance) ou au niveau de l’instance (si besoin de personnaliser le code).

### Interface des instances de modules métier

Code Source 2.27 : /pattern-fonct/ex11-interfaceImplementationMetier.js

1

*/\*\* @ description Test s i un ob j e t impl émente l e s i n t e r f aces géné r i q ues*

*\* pour l e s ob j e t s mé t i e r s suivant notre modlisation .*

*\**

2

3

1. *\* @method t est Interface Implementation*
2. *\* @augments myApp. metier*
3. *\* @ public*
4. *\* @param { Object } instance Instance d ’ ob j e t mé t i e r suppos ée*

−

1. *\* @param { function } instance . get Module retourne l e module mé t i e r (mé thodes s t a t i q u e s )*

−

1. *\* @param { function } instance . g e t A t t r i b u t e*
2. *\* @param { function } instance . s e t A t t r i b u t e*
3. *\* @param { function } instance . has Error*
4. *\* @param { function } instance . get Error Message*
5. *\* @param { function } instance . get Error List*

14 *\**

1. *\* @param { function } instance . get Module () . g e t A t t r i b u t e L i s t*
2. *\* @param { function } instance . get Module () . get Label Text*
3. *\* @param { function } instance . get Module () . t est Regex*
4. *\* @param { function } instance . get Module () . create Instance*

19 *\**

20 *\* @return*

21 *\*/*

22 myApp. addModule . apply (myApp. metier , [ *” t est Interface Implementation ”* , **function** ( i n s ta n c e ) {

23

1. *// Dé f i n i t i o n de l ’ i n t er f ace commune aux modules mé t i e r ( adresse , personne , etc . )*
2. **var** metierCommonMethods = **new** I n t e r f a c e ( [
3. *// 1 ) Mé thodes s t a t i q u e s ( r é s u l t a t s ind épendant de l ’ instance )*
4. *” g e t A t t r i b u t e L i s t ”* , *” get Label Text ”* , *” t est Regex ”* , *” create Instance ”* ,
5. *// 2 ) Mé thodes d ’ instance ( r é s u l t a t s dépendant de l ’ instance )*
6. *” get Module”* , *” g e t A t t r i b u t e ”* , *” s e t A t t r i b u t e ”* , *” has Error ”* , *” get Error Message”* ,

” get Error List ”

30 ] ) ;

31

1. *// cr é ation d ’ une instance ( en l ’ occurrence une adresse )*
2. **var** monObjet = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 34 id : *” 04 abf 85 bc 9 ”* ,
3. numeroRue : *” 2 b i s ”* ,
4. rue : *”Rue de l ’ a Paix ”* ,
5. complementAdresse : *”Bâtiment 3D”* ,
6. code Postal : *” 63000 ”* ,
7. v i l l e : *” Clermont Ferrand”* ,

−

1. pays : *” France”*

41 }) ;

42

43 **return** metierCommonMethods . is Implemented By ( monObjet ) ; 44 } ] ) ;

Code Source 2.28 : /pattern-fonct/ex11-interfaceImplementationTest.js

1

myApp. addModule ( *” mainFunction”* ,

*/\*\* @ description Programme p r i n c i p a l qui con s t ru i t l e s donné es*

* *et gé nère l a vue .*
* *@method myApp. mainFunction*
* *@ public*

*\*/*

**var** monObjet = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({

*// cr é ation d ’ une instance*

**function** () {

2

3

4

5

6

7

8

9

id : *” 04 abf 85 bc 9 ”* , numeroRue : *” 2 b i s ”* ,

rue : *”Rue de l ’ a Paix ”* , complementAdresse : *”Bâtiment 3D”* , code Postal : *” 63000 ”* ,

pays : *” France”*

}) ;

v i l l e : *” Clermont−Ferrand”* ,

**var** t e s t I n t e r f a c e s M e t i e r = myApp. metier . te s t I n te r f a c e I m p l e m e n ta t i o n ( monObjet ) ;

**var** codeHTML ;

**i f** ( t e s t I n t e r f a c e s M e t i e r === **true** ) {

codeHTML = *”***<p>***L ’ ob j e t semble b ien impl émenter l e s mé thodes req u i s es .***</p>***”* ;

} **else** {

codeHTML = *”***<p>***”* + t e s t I n t e r f a c e s M e t i e r + *”***</p>***”*

}

*// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ;

*// Exé cution de l a mé thode mainFunction*

myApp. mainFunction ( ) ;

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

”

Code Source 2.29 : Fichiers *JS* inclus dans ex11-interfaceImplementationTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . /ex05−modulePattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −createModuleMetier . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . / ex 07 −fabrique Objet Metier . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −moduleMetierAdresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

**<s cript src=***” . / ex 08 −fabrique Adresse . j s ”***></s cript>**

**<***!−− Cré ation de f onct i ons d ’ a f f i ch ag e dans myApp. metier . view . adresse −−***>**

**<s cript src=***” . /ex09−adresseView . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . / ex 11 −interface Implementation . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**<s cript src=***” . / ex 11 −interface Implementation Metier . j s ”***></s cript>**

**<s cript src=***” . / ex 11 −interface Implementation Test . j s ”***></s cript>**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

### Exemple d’utilisation : Méthode d’affichage générique

Code Source 2.30 : /pattern-fonct/ex12-objetMetierView.js

1

myApp. addModule ( *” view ”* , { a d r e s s e : {}}) ;

*// Cré ation d ’ un module myApp. view et d ’ un sous−module myApp. view . adresse*

2

3

4 */\*\**

1. *\* Mé thode de géné ration de code HTML pour une instance d ’ ob j e t mé t i e r .*
2. *\* Pour chaque a t t r i b u t d ’ une instance , l a d escrip t i on de l a propri é t é*
3. *\* et sa valeur sont a f f i c h é es .*

8 *\**

1. *\* @method get Html Developped*
2. *\* @augments myApp. view*
3. *\* @ public*
4. *\* @param { Object } instance instance d ’ ob j e t mé t i e r qui impl émente :*

−

1. *\* ” get Module ” , ” g e t A t t r i b u t e ” , ” s e t A t t r i b u t e ” ,*
2. *\* ” has Error ” , ” get Error Message ” , ” get Error List ”*
3. *\* a i n s i que l e s mé thodes s t a t i q u e s :*
4. *\* ” g e t A t t r i b u t e Li s t ” , ” get Label Text ” , ” t est Regex ” ,*
5. *\* ” create Instance ” , ” get Module ” .*

18 *\*/*

1. myApp. addModule . apply (myApp. view , [ *” get Html Generic”* , **function** ( i n s ta n c e ) {
2. **var** t e s t I n t e r f a c e s M e t i e r = myApp. metier . te s t I n te r f a c e I m p l e m e n ta t i o n ( i n s ta n c e ) ;
3. **i f** ( t e s t I n t e r f a c e s M e t i e r !== **true** ) {
4. *// Retourner un message d ’ erreur au l i eu du code HTML de l ’ instance*
5. **return** t e s t I n t e r f a c e s M e t i e r ; 24 }
6. *// Code HTML pour l ’ a f f i ch ag e de l ’ instance :*
7. **var** codeHTML = *”***<div><span><h2>***Donné es de l ’ instance :***</h2>***”* ;
8. **for** ( **var** i =0 ; i **<** i n s ta n c e . getModule () . g e t A ttr i b u te L i s t () . l e n g th ; ++i ) {
9. **var** attribute Name = i n s ta n c e . getModule ( ) . g e t A t tr i b u te L i s t ( ) [ i ] ;
10. codeHTML += *”***<s trong>***”*

30 + i n s ta n c e . getModule () . get Label Text ( attribute Name ) + *” : ”*

31 + *”***</s trong>** *”*

32 + i n s ta n c e . g e t A ttr i b u te ( attribute Name ) + *”***<br** */***>***”* ; 33 }

34 codeHTML += *”***</span>***”* ; 35

1. *// variante en énumé rant automatiquement l e s propri é t é s*
2. codeHTML += *”***<span><h2>***E rreurs :***</h2>***”* ;
3. **for** ( **var** i =0 ; i **<** i n s ta n c e . getModule () . g e t A ttr i b u te L i s t () . l e n g th ; ++i ) {
4. **var** attribute Name = i n s ta n c e . getModule ( ) . g e t A t tr i b u te L i s t ( ) [ i ] ;
5. codeHTML += *”***<s trong>***”*

41 + i n s ta n c e . getModule () . get Label Text ( attribute Name ) + *” : ”*

42 + *”***</s trong>** *”* +

43 i n s ta n c e . get Error Message ( attribute Name ) + *”***<br** */***>***”* ; 44 }

1. codeHTML += *”***</span></div>***”* ;
2. **return** codeHTML ; 47 } ] ) ;

##### Code Source 2.31 : /pattern-fonct/ex12-objetMetierViewTest.js

1

*/\*\* Module de t e s t avec une mé thode d ’ a f f i ch ag e d ’ une instance*

* *d ’ adresse avec l e s mé thodes du package myApp. view . adresse*
* *@module myApp. t e s t*
* *@method myApp. t e s t . t est Affiche Adresse*
* *@ public*

*\* @param { Object } adresse − instance d ’ ob j e t mé t i e r du module ” adresse ”*

myApp. addModule ( *” t e s t ”* , {

t e s t A f f i c h e I n s t a n c e : **function** ( i n s ta n c e ) {

*\*/*

2

3

4

5

6

7

8

9

1. **return** myApp. view . getHtml Generic ( i n s ta n c e ) ;
2. } *// f i n de l a mé thode t es t A f f i ch e In s t an ce*
3. } *// f i n du module myApp. t e s t*
4. ) ; *// f i n de l ’ invocation de myApp. addModule*

14

15

1. */\*\* @ description Programme p r i n c i p a l qui con s t ru i t l e s donné es*
2. *\* et gé nère l a vue .*
3. *\* @method myApp. mainFunction*
4. *\* @ public*

20 *\*/*

1. myApp. addModule ( *” mainFunction”* , **function** () {
2. *// cr é ation d ’ une instance*
3. **var** a d r e s s e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 24 id : *” 04 abf 85 bc 9 ”* ,
4. numeroRue : *” 2 b i s ”* ,
5. rue : *”Rue de l ’ a Paix ”* ,
6. complementAddr : *” aa”* ,
7. *// ou b l i du champs complementAddr*
8. code Postal : *” 630000 ”* ,
9. v i l l e : *” Clermont Fernand”* ,

−

1. pays : *” France 2 ”*

32 }) ;

33

34 **var** codeHTML = **this** . t e s t . t e s t A f f i c h e I n s t a n c e ( a d r e s s e ) ; 35

1. a d r e s s e . s e t A ttr i b u te ( *” complementAddr”* , *”\”Bâtiment 3D\””* ) ;
2. a d r e s s e . s e t A ttr i b u te ( *” code Postal ”* , *” 63000 ”* ) ;
3. a d r e s s e . s e t A ttr i b u te ( *” pays ”* , *” France”* ) ;
4. a d r e s s e . s e t A ttr i b u te ( *” numeroRue”* , *”@#\*m”* ) ; 40

41 codeHTML += **this** . t e s t . t e s t A f f i c h e I n s t a n c e ( a d r e s s e ) ; 42

1. *// a l e r t (codeHTML) ;*
2. *// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*
3. document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ; 46 }) ;

47

1. *// Exé cution de l a mé thode mainFunction*
2. myApp. mainFunction ( ) ;

”

Code Source 2.32 : Fichiers *JS* inclus dans ex12-objetMetierViewTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . /ex05−modulePattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −createModuleMetier . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . / ex 07 −fabrique Objet Metier . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 06 −moduleMetierAdresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

**<s cript src=***” . / ex 08 −fabrique Adresse . j s ”***></s cript>**

1

2

3

4

5

6

7

8

9

10

11

12

13

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . / ex 11 −interface Implementation . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . / ex 11 −interface Implementation Metier . j s ”***></s cript>**

**<***!−− Cré ation de f onct i ons d ’ a f f i ch ag e géné r i que dans myApp. metier . view −−***>**

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**<s cript src=***” . /ex 12 −objet Metier View . j s ”***></s cript>**

**<s cript src=***” . / ex 12 −objet Metier View Test . j s ”***></s cript>**

14

15

16

17

18

19

20

21

**Chapitre 3**

**Constructeurs, Prototype et *Patterns***

# Associés

## Constructeurs

##### Un *classe* en *Javascript* se crée à partir d’un constructeur, qui est une fonction dont le nom est le nom de la classe à créer. À l’intérieur du constructeur, les propriétés de la classe sont créées et initialisées à l’aide de l’identificateur this. Le constructeur retourne un unique objet dont les propriétés correspondent à celles qui ont été initialisées à l’aide de l’identificateur this. En d’autres termes, le constructeur retourne une instance de la classe. Par convention, **les noms de constructeurs commencent par une majuscule**.

Code Source 3.1 : /pattern-proto/ex01-classeTelephone.js

1 */\*\**

* 1. *\* Constructeur de t é l éphone . Notez l a majuscule sur l e nom.*
  2. *\* @ constructor*
  3. *\* @param { s t r i n g } t e l 1 l e numéro de t é l éphone .*

−

* 1. *\* @param { s t r i n g } [ t e l 2 ] un second numéro de t é l éphone .*

−

6 *\*/*

7 **var** Telephone = **function** ( te l 1 , */\* argument optionnel \*/* t e l 2 ) { 8

9 **var** checkPhone = **function** ( t e l ) {

1. *// Test d ’ expression r é g u l i è r e après suppression des espaces et t ab u l a t i on s*
2. **i f** ( **typeof** t e l . l i b e l l e !== *” s t r i n g ”* | | **typeof** t e l . numero !== *” s t r i n g ”* | | 12 */^((\+33) | 0 ) [ 0 9]{9}$/* . t e s t ( t e l . numero . r e p l a c e (/\ s /g , *’ ’* ) ) !== **true** ) {

−

1. throw {
2. name : *” I l l egal Argument Exception ”* ,
3. message : *”Numéro de t é l éphone \””* + t e l . l i b e l l e + *” : ”* + t e l . numero + *”*

\” i n v a l i d e ”

16 }

17 }

18 } ;

19

1. checkPhone ( t e l 1 ) ;
2. *// Cré ation d ’ un a t t r i b u t de l a c l as s e*
3. **this** . t e l 1=t e l 1 ; 23
4. **i f** ( t e l 2 !== undefined ) {
5. checkPhone ( t e l 2 ) ;
6. *// Cré ation d ’ un a t t r i b u t de l a c l as s e*

27

}

**this** . t e l 2=t e l 2 ;

}

*/\*\**

* *@method get Html*
* *@return { s t r i n g } l e code HTML pour a f f i c h e r une instance .*

*\*/*

**this** . getHtml = **function** () {

**var** htmlCode = **this** . t e l 1 . l i b e l l e + *” : ”* + **this** . t e l 1 . numero + *”***<br** */***>***”* ;

**i f** ( **this** . t e l 2 !== undefined ) {

htmlCode += **this** . t e l 2 . l i b e l l e + *” : ”* + **this** . t e l 2 . numero + *”***<br** */***>***”* ;

}

**return** htmlCode ;

} ;

28

29

30

31

32

33

34

35

36

37

38

39

40

41

Code Source 3.2 : /pattern-proto/ex01-classeTelephoneTest.js

1

**try** {

*// Appel du constructeur avec l e mot c l é ”new” :*

**var** t e l **= new** Telephone ({ l i b e l l e : *” Maison”* , numero : *”+33 1 23 45 67 89 ”* } ,

{ l i b e l l e : *” Mobile”* , numero : *” 09 87 65 43 21 ”* }) ;

*// U t i l i s a t i o n de l a mé thode get Html ()*

**var** codeHTML **=** *”***<p>***”* + t e l . getHtml ( ) + *”***</p>***”* ;

} **catch** ( e r r ) {

a l e r t ( e r r . message ) ;

}

*// U t i l i s a t i o n de ”myApp. view . codeHTML” pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML **=** codeHTML ;

2

3

4

5

6

7

8

9

10

11

Un constructeur doit systématiquement être employé avec le mot clé new. En effet, l’emploi d’un constructeur sans le mot clé new, qui ne génère, en soi, aucune exception ni *warning* conduit à un comportement imprévisible, généralement catastrophique. D’où l’importance de **respecter la convention que les noms de constructeurs commencent par une majuscule**, contrairement à toutes les autres fonctions ou variables.

## Prototypes

### Notion de prototype

##### Les méthodes de classes telles que vues jusqu’à présent ont l’inconvénient que ces méthodes sont des propriétés des objets, qui existent en autant d’exemplaires qu’il y a d’instance des objets, alors qu’elles sont constantes.

Pour éviter cela, on peut mettre les méthodes non pas directement dans l’objet, mais dans son *prototype*. Le prototype est lui-même une propriété de l’objet, mais qui est partagée entre tous les objets de la classe (il s’agit d’une variable de classe). Toutes les variables de classes doivent être crées au niveau du prototype.

Code Source 3.3 : /pattern-proto/ex02-prototype.js

1

2

*/\*\**

*\*/*

Telephone . prototype . getNumero = **function** ( l i b e l l e ) {

**i f** ( **this** . t e l 1 . l i b e l l e . toLowerCase ( ) === l i b e l l e . toLowerCase ( ) ) {

**return this** . t e l 1 . numero ;

}

**i f** ( **this** . t e l 2 !== undefined &&

**this** . t e l 2 . l i b e l l e . toLowerCase () === l i b e l l e . toLowerCase () ) {

**return this** . t e l 2 . numero ;

}

**return** *”Numéro i n ex i s t an t ”* ;

} ;

* *Augmente l a c l as s e Telephone en ejoutant une mé thode au prototype de Telephone*
* *@method get Html By Libelle*
* *@return { s t r i n g } l e code HTML pour a f f i c h e r une instance .*

3

4

5

6

7

8

9

10

11

12

13

14

15

16

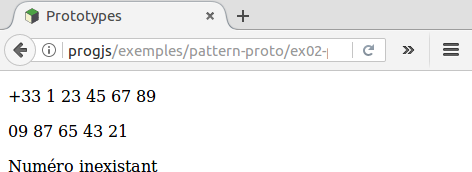


Figure 3.1 : Illustration du code source [3.4](#_bookmark75)

Code Source 3.4 : /pattern-proto/ex02-prototypeTest.js (cf. Fig [3.1](#_bookmark74))

1

**try** {

*// Appel du constructeur avec l e mot c l é ”new” :*

**var** t e l = **new** Telephone ({ l i b e l l e : *” Maison”* , numero : *”+33 1 23 45 67 89 ”* } ,

{ l i b e l l e : *” Mobile”* , numero : *” 09 87 65 43 21 ”* }) ;

*// U t i l i s a t i o n de l a mé thode getNumero () du prototype* **var** codeHTML = *”***<p>***”* + t e l . getNumero ( *” maison”* ) + *”***</p>***”* ; codeHTML += *”***<p>***”* + t e l . getNumero ( *” mobile”* ) + *”***</p>***”* ; codeHTML += *”***<p>***”* + t e l . getNumero ( *” t r a v a i l ”* ) + *”***</p>***”* ;

} **catch** ( e r r ) {

a l e r t ( e r r . message ) ;

}

*// U t i l i s a t i o n de ”myApp. view . codeHTML” pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

2

3

4

5

6

7

8

9

10

11

12

13

La méthode Object.prototype.hasOwnProperty() permet de tester si une propriété d’un objet existe au niveau de l’objet lui-même, ou au niveau de son prototype, ou encore du prototype de son prototype.

Le fonctionnement de la notion de prototype est le suivant : Lors d’une tentative d’accès à un propriété de l’objet, la propriété est tout d’abord recherchée au niveau des propriétés propres. Seulement si la propriété n’existe pas dans les propriétés propres, elle est ensuite recherchée dans le prototype de l’objet. Si elle n’existe pas non plus à ce niveau, la propriété est recherchée dans le prototype du prototype, et ainsi de suite...

Ce processus s’appelle la *délégation* et il permet de spécialiser les objets, en les faisant hériter

des propriétés d’un prototype, tout en leur permettant de surcharger (redéfinir) les données ou

méthodes. Ceci constitue un mécanisme très souple d’héritage, entièrement dynamique.

### Surcharge des méthodes du prototype : l’exemple de toString

La méthode toString, qui permet de convertir un objet en chaîne de caractères (par exemple pour l’afficher) a une implémentation par défaut définie dans le prototype de la classe Object. On peut la surcharger dans le prototype de notre classe Telephone pour changer le compor- tement par défaut de la méthode toString et mettre en forme à notre guise les numéros de téléphone.

Code Source 3.5 : /pattern-proto/ex03-toString.js

1

*/\*\**

*\* @return { s t r i n g } une cha î ne de caractère repr é s en t at n t l ’ instance de Telephone*

*\*/*

Telephone . prototype . to S tr i n g = **function** () {

**var** te x te = **this** . t e l 1 . l i b e l l e + *” : ”* + **this** . t e l 1 . numero ;

**i f** ( **this** . t e l 2 !== undefined ) {

te x te += *” et ”* + **this** . t e l 2 . l i b e l l e + *” : ”* + **this** . t e l 2 . numero ;

}

**return** te x te ;

}

*\* @ override t o S t r in g*

2

3

4

5

6

7

8

9

10

11

12

Code Source 3.6 : /pattern-proto/ex03-toStringTest.js

1

**try** {

*// Appel du constructeur avec l e mot c l é ”new” :*

**var** t e l = **new** Telephone ({ l i b e l l e : *” Maison”* , numero : *”+33 1 23 45 67 89 ”* } ,

{ l i b e l l e : *” Mobile”* , numero : *” 09 87 65 43 21 ”* }) ;

*// U t i l i s a t i o n i m p l i c i t e de l a mé thode t o S t r in g () ( conversion )*

**var** codeHTML = *”***<p>***”* + t e l + *”***</p>***”* ;

} **catch** ( e r r ) {

a l e r t ( e r r . message ) ;

}

*// U t i l i s a t i o n de ”myApp. view . codeHTML” pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

2

3

4

5

6

7

8

9

10

11

”

Code Source 3.7 : Fichiers *JS* inclus dans ex03-toStringTest.html

**<***!−− Dé f i n i t i o n du constructeur de Telephone −−***>**

**<s cript src=***” . / ex 01 −classe Telephone . j s ”***></s cript>**

**<***!−− Redé f i n i t i o n de l a mé thode t o S t r in g du prototype de Telephone −−***>**

*−−***<***!−− Impl é mentation du t e s t et de l a vue −−***>**

**<s cript src=***” . / ex 03 −to String . j s ”***></s cript>**

**<s cript src=***” . / ex 03 −to String Test . j s ”***></s cript>**

1

2

3

4

5

6

7

## Modélisation de Modules Métier (version 2)

Le but de cette partie est de proposer une nouvelle modélisation pour les modules métier définis dans la partie [2.6](#_bookmark53). L’objectif est d’utiliser la notion de prototype pour éviter d’avoir autant de copies des méthodes statiques et des méthodes de classe des modules métier qu’il y a d’instance. Constatant que le code source de toutes ces méthodes est le même pour toutes les méthodes, ce code sera factorisé dans le prototype du module, qui sera lui-même accessible via le prototype des instances.

### Attributs et méthodes statiques (version 2)

Code Source 3.8 : /pattern-proto/ex05-createModuleMetierProto.js

* 1. */\*\* @ description Dé f i n i t l e s propri é t é s géné ral e des ob j e t s mé t i e r s*
  2. *\* à p ar t i r d ’ une sp é c i f i c a t i o n des a t t r i b u t s .*
  3. *\* On ajoute au mé t i e r un ob j e t qui es t l ’ i n t er f ace publique d ’ une fonction qui s u i t l e pattern ” module ” .*
  4. *\* La fonction retourne son i n t er f ace publique qui es t un ob j e t .*
  5. *\* Cet ob j e t es t ajout é comme sous module au module ” metier ” .*

−

6 *\**

1. *\* Dans cet objet , on ne t rouve pas pour l e moment l e s propri é t é s d ’ instance .*
2. *\* Celles −ci seront ajout é es par ” augmentation ” .*
3. *\* @module create Module Metier*
4. *\* @augments myApp. metier*
5. *\* @param { s t r i n g } moduleName nom du module mé t i e r*

−

1. *\* @param { Object } a t t r i b u t es Pat t er n s ob j e t contenant l e s a t t r i b u t s d ’ un module mé t i e r*

−

13 *\**

du module mé t i e−r

Chaque propri é t é de a t t r i b u t es Pat t er n s dé f i n i t un a t t r i b u t

1. *\* chaque a t t r i b u t possède*

−

1. *\* + une fonction de t e s t sur l e s valeurs nommée regex Test*
2. *\* + un t ex t e de l a b e l permettant de dé s i gner l a donnée pour l ’ u t i l i s a t e u r*

17 *\*/*

1. myApp. addModule . apply (myApp. metier , [ *” create Module Metier ”* ,
2. **function** ( moduleName ,
3. a t t r i b u te s P a t te r n s ) {

21 //////////////////////////////////////////////////////////

1. *// Si l e modèle de t ou t es l e s mé thodes s t a t i q u e s communes*
2. *// à tous l e s ob j e t s mé t i e r n ’ a pas encore é t é cr éé*
3. *// ( Cet ob j e t ex i s t e en un exemplaire unique . )*
4. **i f** (myApp. metier . methodes Statiques Modules Metier === undefined ) {
5. *//On cr ée l e modèle de t ou t es l e s mé thodes s t a t i q u e s communes*
6. *// à tous l e s ob j e t s mé t i e r .*
7. myApp. metier . methodes Statiques Modules Metier = {
8. */\*\* @ description Renvoie l a l i s t e des propri é t é s attendues des instances*

.

1. *\* @method g e t A t t r i b u t e Li s t*

31 *\*/*

1. g e t A ttr i b u te L i s t : **function** () {
2. **return** a t t r i b u t e L i s t ; 34 } ,

35

1. */\*\* @ description Renvoie l e t ex t e de d es cr i p t i on de l ’ a t t r i b u t attendue des instances .*
2. *\* Renvoie undefined en cas d ’ erreur ( propri é t é inconnue )*
3. *\* @method get Label Text*
4. *\* @param { s t r i n g } attribute Name nom de propri é t é*

−

1. *\* @return { s t r i n g } l e t ex t e de d es cr i p t i on courte du champs*

41 *\*/*

1. get Label Text : **function** ( attribute Name ) {
2. **return** a t t r i b u te s P a t te r n s [ attribute Name ] . l a b e l Te xt ; 44 } ,

45

46

1. */\*\* @ description Expose l e t e s t d ’ expression r é g u l i è r e des a t t r i b u t s des instances .*
2. *\* Peut ê t re u t i l i s ée pour l e f i l t r a g e des donné es d ’ un formulaire .*
3. *\* @method t est Regex*
4. *\* @param { s t r i n g } attribute Name nom de propri é t é*

−

1. *\* @param { s t r i n g } value valeur pour i n i t i a l i s e r l ’ a t t r i b u t*

−

1. *\* @return { boolean | s t r i n g } t rue s i l a chaine es t un a t t r i b u t valide ,*
2. *\* un message d ’ erreur sinon .*

54 *\*/*

1. test Regex : **function** ( attribute Name , value ) {
2. **i f** ( a t t r i b u te s P a t te r n s [ attribute Name ] === undefined ) {
3. **return** *”La propri é t é ”* + attribute Name + *” n ’ ex i s t e pas”* ;
4. } **else** {
5. **return** a t t r i b u te s P a t te r n s [ attribute Name ] . regex Test ( value ) ; 60 }

61 }

1. } ; *// Fin du l i t t é ral myApp. metier . methodes Statiques Modules Metier*
2. } *// f i n du ” i f (myApp. metier . methodes Statiques Modules Metier === undefined )”*

64

65 **var** ModuleMetier = **function** () {} ; 66

1. *// Le modèle de t ou t es l e s mé thodes s t a t i q u e s communes à tous l e s ob j e t s*
2. *// mé t i e r s er t de prototype à tous l e s modules mé t i e r .*
3. *// Cela permet de rendre ces mé thodes ( exemplaire unique ) a c c e s s i b l e s*
4. *// dans tous l e s modules mé t i e r*
5. ModuleMetier . prototype

72 = myApp. metier . methodes Statiques Modules Metier ; 73

1. *// Cré ation du module avec l e constructeur de modules mé t i e r s*
2. *// pout pouvoir dé f i n i r un prototype commun pour tous*
3. *// l e s modules mé t i e r .*
4. myApp. metier [ moduleName ] = **new** ModuleMetier ( ) ; 78 */\*\**
5. *\* Tableau contenant l a l i s t e des a t t r i b u t s d ’ une instance .*
6. *\* Le t ab l eau es t pr é cal cu l é l or s de l ’ i n i t i a l i s a t i o n .*
7. *\* @member*
8. *\* @ private*

83 *\*/*

1. **var** a t t r i b u t e L i s t = **function** () {
2. **var** l i s t e = [ ] ; 86
3. *// Parcours des propri é t é s de l ’ ob j e t a t t r i b u t es Pat t er n s . regex Test*
4. *// qui correspondent aux a t t r i v u t s de l ’ instance*
5. **for** ( **var** attribute Name **in** a t t r i b u te s P a t te r n s ) {

90

}

*// Ne pasconsid é rer l e s propri é t é s ” hé r i t é es ” du prototype .*

**i f** ( a t t r i b u te s P a t te r n s . hasOwnProperty ( attribute Name ) ) { l i s t e . push ( attribute Name ) ;

}

}() ; *// appel immé d i at de l a fonction anonyme .*

*/\*\* @ description Propri é t é contenant l e s sp é c i f i c a t i o n s*

*\* de chaque a t t r i b u t t e l l e s que donné es en paramètres*

*\*/*

*//myApp. metier [ moduleName ] . a t t r i b u t es Pat t er n s = a t t r i b u t es Pat t er n s ;*

}] *// f i n de l a fonction qui cr ée l ’ ob j e t myApp. metier [ moduleName ]*

) ; *// f i n de l ’ appel ” apply ” de l a mé thode myApp. addModule*

*// ( ajout de l ’ ob j e t t h i s au metier , sous l e nom create Module Metier )*

**return** l i s t e ;

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

### Fabrique générique d’instances métier (version 2)

Pour les méthodes d’instance, le code source des méthodes étant indépendant de l’instance (même si le résultats de ces méthodes dépend des valeurs attributs, et donc de l’instance), il sera factorisé dans un prototype unique, qui est créé une seule fois. Le constructeur correspondant est mémorisé dans la propriété privateInstanceConstructor du module myApp.metier.

Comme les attributs d’un constructeur sont toujours publics, nous instantions avec notre privateInstanceConstructor une instance qui sera elle même privée (en suivant le *pattern module* comme dans la partie [2.2](#_bookmark41)).

Le module métier (comme par exemple myApp.metier.adresse), dont l’interface contient

les méthodes statiques (méthodes de classe), sera utilisé comme prototype des instances (PublicInstanceI Une instance du constructeur PublicInstanceInterface est retournée comme interface du

*pattern module* exposant à la fois les méthodes statiques et les méthodes d’instance.

Code Source 3.9 : /pattern-proto/ex05-fabriqueObjetMetierProto.js

1

*/\*\**

* *Fabrique qui cr ée des ob j e t s repr é sentant des instance d ’ ob j e t s mé t i er , suivant l e ” pattern module ” .*
* *Le paramètre spec de notre fonction es t un ob j e t contenant l e s a t t r i b u t s de l ’ instance à cr é er .*

*\**

* *@method create Instance Generique*
* *@augments myApp. metier*

*\* @param { Object | n u l l } input Obj − sp é c i f i c a t i o n des a t t r i b u t s d ’ une instance d ’*

*ob j e t mé t i e r .*

*\* Si input Obj es t null , on cr ée une instance par dé f aut ( id a l é atoire , autres a t t r i b u t s v ide ) .*

*\* @param { s t r i n g | | undefined } input Obj . id − i d e n t i f i a n t unique de l ’ instance*

*\* @param { s t r i n g | number} input Obj . attribute Name − valeur à a f f e c t e r à l ’*

*\**

*a t t r i b u t attribute Name*

*( exemple pour une adresse : input Obj . numeroRue , input Obj .*

*code Postal , etc .*

*\*/*

myApp. addModule . apply (myApp. metier , [ *” create Instance Generique ”* , **function** ( inputObj ) {

2

3

4

5

6

7

8

9

10

11

12

13

14

15 that = **this** ; 16

17 //////////////////////////////////////////////////////////

1. *// Si l e modèle de t ou t es l e s mé thodes priv é es communes*
2. *// à t ou t es l e s instances d ’ ob j e t s mé t i e r n ’ a pas encore é t é cr éé*
3. *// ( Cet ob j e t ex i s t e en un exemplaire unique . )*
4. **i f** (myApp. metier . methodes Instances Metier === undefined ) {
5. *// Constructeur d ’ un ob j e t unique avec un prototype*
6. **var** Pr i v a te In s ta n c e s C o n s tr u c to r = **function** () {
7. */\*\* @ description contient l e s valeurs des a t t r i b u t s de l ’ instance , i n i t i a l em en t v ide*
8. *\* @member*
9. *\* @ private*

27 *\*/*

1. **this** . d a ta A ttr i b u te s = {} ;
2. */\*\* @ description contient l e s messages d ’ erreur associ é s aux a t t r i b u t s de l ’ instances*
3. *\* @member*
4. *\* @ private*

32 *\*/*

33 **this** . data Error = {} ; 34 } ;

35

36 Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . add Error

37 = **function** ( attribute Name , message ) {

1. *// Ajout d ’ une propri é t é*
2. **this** . data Error [ attribute Name ] = message ; 40 } ;

41

42 Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . s e t A ttr i b u te

43 = **function** ( attribute Name , value ) {

1. **var** result Test Regex = that . test Regex ( attribute Name , value ) ;
2. *// On i n t i a l i s e l ’ a t t r i b u t de l ’ instance*
3. **this** . d a ta A ttr i b u te s [ attribute Name ] = value ;
4. *// Si l a v a l i d a t i on par expression r é g u l i è r e es t pass ée*
5. **i f** ( result Test Regex === **true** ) {
6. *// On e f f ace une v i e i l l e erreur é v en t u e l l e*
7. **delete this** . data Error [ attribute Name ] ;
8. } **else** {
9. *// On i n i t a l i s e l a propri é t é de l ’ ob j e t des erreurs .*
10. *// avec l e message d ’ erreur .*
11. **this** . add Error ( attribute Name , *” A t t r i b u t ”* + value +
12. *” i n v a l i d e : ”* + result Test Regex ) ; 56 }

57 } ;

58

59 Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . g e t A ttr i b u te

60 = **function** ( attribute Name ) {

61 **return this** . d a ta A ttr i b u te s [ attribute Name ] ; 62 } ;

63

64 Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . has Error

65 = **function** () {

1. **for** ( **var** attribute Name **in this** . data Error ) {
2. **i f** ( **this** . data Error . hasOwnProperty ( attribute Name ) ) {
3. **return true** ;

|  |  |
| --- | --- |
| 69 | } |
| 70 | } |
| 71 | **return f al s e** ; |
| 72 | } ; |
| 73 |  |
| 74  75  76 | Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . g e t E r r o r L i s t  = **function** () {  **var** e r r o r L i s t = [ ] ; |
| 77 | **for** ( **var** attribute Name **in this** . data Error ) { |
| 78 | **i f** ( **this** . data Error . hasOwnProperty ( attribute Name ) ) { |
| 79 | e r r o r L i s t . push ( attribute Name ) ; |
| 80 | } |
| 81 | } |
| 82 | **return** e r r o r L i s t ; |
| 83 | } ; |
| 84 |  |
| 85 | Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . get Error Message |
| 86 | = **function** ( attribute Name ) { |
| 87 | **return this** . data Error [ attribute Name ] ; |
| 88 | } ; |
| 89 | */\*\* @ description Géné ration d ’ un ID a l é a t o i r e en cas de cr é ation d ’ une* |
|  | *nou v e l l e instance* |
| 90 | *\* ( cas où l e s s p e c i f i c a t i o n s input Obj sont n u l l )* |
| 91 | *\* @ private* |
| 92 | *\*/* |
| 93 | Pr i v a te In s ta n c e s C o n s tr u c to r . prototype . generateRandomId |
| 94 | = **function** () { |
| 95 | **var** id Length = 10 ; |
| 96 | **var** r e s u l t a t = *””* ; |
| 97  98 | **var** hexa D i g i ts = Array ( *” 0 ”* , *” 1 ”* , *” 2 ”* , *” 3 ”* , *” 4 ”* , *” 5 ”* , *” 6 ”* , *” 7 ”* , *” 8 ”* , *” 9 ”* , *” a”* , *” b”* , *” c”* , *” d”* , *” e”* , *” f ”* ) ;  **var** i ; |
| 99 | **for** ( **var** i =0 ; i**<**10 ; ++i ) { |
| 100 | r e s u l t a t += he xa Di g i ts [ Math . f l o o r ( Math . random ( ) \*16 ) ] ; |
| 101 | } |
| 102 | **return** r e s u l t a t ; |
| 103 | } |
| 104 |  |
| 105 | *// On mémorise l e constructeur d ’ instances priv é es pour n ’ en avoir qu ’ un* |
| 106 | myApp. metier . p r i v a te I n s ta n c e C o n s tr u c to r = Pr i v a te In s ta n c e s C o n s tr u c to r ; |
| 107 | } *// f i n du ” i f (myApp. metier . methodes Instances Metier === undefined )”* |
| 108 |  |
| 109 | *// On cr ée une instance priv ée ( v ar i ab l e l ocal e , voir ” pattern module ”)* |
| 110 | **var** p r i v a te I n s ta n c e = **new** myApp. metier . p r i v a te I n s ta n c e C o n s tr u c to r ( ) ; |
| 111 |  |
| 112 |  |
| 113 | */\*\* @ description Cré ation d ’ un constructeur priv é , cr é ant une c l as s e qui* |
| 114 | *\* contiendra l e s mé thodes permettant de manipuler l ’ instance cr éée ,* |
| 115 | *\* qui cont i t u e l ’ i n t er f ace du pattern module .* |
| 116 | *\** |
| 117 | *\* L ’ i n t ér ê t d ’ un constructeur es t de dé f i n i r l e s mé thode au niveau du prototype ,* |
| 118 | *\* v ia l ’ ob j e t unique myApp. metier . methodes Instances Metier , cr éé une s eu l e f o i s .* |
| 119 | *\** |
| 120 | *\* De plus , pe prototype des instances hé r i t e l u i −même des mé thodes* |

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

s t a t i q u e s

* *du module mé t i e r .*

\*/

**var** P u b l i c I n s ta n c e I n te r f a c e = **function** () {} ;

// MISE À DISPOSITION DES MÉTHODES STATIQUES DU MODULE

// VIA LE PROTOTYPE

P u b l i c I n s ta n c e I n te r f a c e . prototype = **this** ; *// exemple : t h i s = myApp. metier . adresse*

/\*\* @ description Retourne l e module avec l e s mé thodes ” s t a t i q u e s ”

* *(comme l ’ accès d i r ect à l a l i s t e des a t t r i b u t s ou l e s t e s t s regex )*
* *@return { Object } l e module myApp. metier . moduleName*

\*/

P u b l i c I n s ta n c e I n te r f a c e . prototype . getModule = **function** () {

**return this** ;

} ;

// In t er f ace publique du module , retourn ée par l a f ab r i q u e

**var** i n s t a n c e I n t e r f a c e = **new** P u b l i c I n s ta n c e I n te r f a c e ( ) ;

i n s t a n c e I n t e r f a c e . getModule = **function** () {

**return this** ;

} ;

/////////////////////////////////////////////////

// On expose l e s mé thodes du prototype de l ’ instance priv ée

// v ia l ’ i n t er f ace des instances :

/\*\* @ description Ajoute une propri é t é ( message d ’ erreur ) dans t h i s . data Error

* *correspondant à un a t t r i b u t .*
* *@method add Error*
* *@ public*

\*/

i n s t a n c e I n t e r f a c e . add Error = **function** () { p r i v a te I n s ta n c e . add Error ( ) ;

} ;

/\*\* @ description S e t t er : i n i t i a l i s e l a valeur des a t t i b u t s d ’ une instance .

* *En cas d ’ erreur un message pour cet t e propri é t é es t es t ajout é dans t h i s . data Error .*
* *En l ’ absence d ’ erreur , une é v en t u e l l e erreur pr écé dente es t e f f ac ée .*
* *@method s e t A t t r i b u t e*
* *@ public*

\*/

i n s t a n c e I n t e r f a c e . s e t A ttr i b u te = **function** ( attribute Name , value ) { p r i v a te I n s ta n c e . s e t A ttr i b u te ( attribute Name , value ) ;

} ;

/\*\* @ description Accesseur pour tous l e s membres priv é s d ’ instance .

* *@method g e t A t t r i b u t e*
* *@ public*
* *@param { s t r i n g } attribute Name nom de l ’ a t t r i b u t attendue d ’ une instance*

−

* *@return { s t r i n g } l a valeur de l ’ a t t r i b u t ou undefined en cas de nom d ’*

|  |  |
| --- | --- |
| 173 | *a t t r i b u t inconnu .*  *\*/* |
| 174 | i n s t a n c e I n t e r f a c e . g e t A ttr i b u te = **function** ( attribute Name ) { |
| 175 | **return** p r i v a te I n s ta n c e . g e t A ttr i b u te ( attribute Name ) ; |
| 176 | } ; |
| 177 |  |
| 178  179 | */\*\* @ description Dé termine s i l ’ instance comprend au moins une erreur .*  *\* @ public* |
| 180 | *\* @return { boolean } t rue s ’ i l y a ( au moins ) une erreur , f a l s e sinon* |
| 181 | *\*/* |
| 182 | i n s t a n c e I n t e r f a c e . has Error = **function** ( attribute Name ) { |
| 183 | **return** p r i v a te I n s ta n c e . has Error ( attribute Name ) ; |
| 184 | } ; |
| 185 |  |
| 186 | */\*\* @ description Ré cupère l a l i s t e des champs qui ont une erreur* |
| 187 | *\* @ public* |
| 188 | *\* @return { s t r i n g [ ] } t ab l eau des noms de propri é t é s qui comportent une* |
|  | *erreur .* |
| 189 | *\*/* |
| 190 | i n s t a n c e I n t e r f a c e . g e t E r r o r L i s t = **function** () { |
| 191 | **return** p r i v a te I n s ta n c e . g e t E r r o r L i s t ( ) ; |
| 192 | } ; |
| 193 |  |
| 194 | */\*\* @ description Donne l ’ accès au message d ’ erreur d ’ un a t t r i b u t ( s ’ i l ex i s t e* |
|  | *) .* |
| 195 | *\* @method get Error Message* |
| 196 | *\* @ public* |
| 197 | *\* @param { s t r i n g } attribute Name − nom d ’ a t t r i b u t d ’ une instance de module mé t i e r* |
| 198  199 | * *@return { s t r i n g | undefined } l e message d ’ erreur pour un a t t r i b u t s ’ i l ex i s t e* * *ou undefined en l ’ absence d ’ erreur* |
| 200 | *\*/* |
| 201 | i n s t a n c e I n t e r f a c e . get Error Message = **function** ( attribute Name ) { |
| 202 | **return** p r i v a te I n s ta n c e . get Error Message ( attribute Name ) ; |
| 203 | } ; |
| 204 |  |
| 205 | *// I n i t i a l i s a t i o n des a t t r i b u t s de l ’ instance* |
| 206 | *// Si l ’ ob j e t en argument es t null , on con s t ru i t une instance par dé f aut* |
| 207 | **i f** ( inputObj !== **null** ) { |
| 208 | *// Parcours des propri é t é s de g e t A t t r i b u t e Li s t ()* |
| 209 | *// qui correspondent aux a t t r i b u t s de l ’ instance à cr é er* |
| 210 | **for** ( **var** i = 0 ; i **< this** . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++i ) { |
| 211 | **var** attribute Name = **this** . g e t A t tr i b u te L i s t ( ) [ i ] ; |
| 212 | *// U t i l i s a t i o n du s e t t e r pour i n i t i a l i s e r l ’ a t t r i b u t* |
| 213 | p r i v a te I n s ta n c e . s e t A ttr i b u te ( attribute Name , inputObj [ attribute Name ] ) ; |
| 214 | } |
| 215 | } **else** { |
| 216 | *// On i n i t i a l i s e l e s valeurs des a t t r i b u t s à ””* |
| 217 | **for** ( **var** i = 0 ; i **< this** . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++i ) { |
| 218  219  220 | **var** attribute Name = **this** . g e t A t tr i b u te L i s t ( ) [ i ] ;  **i f** ( attribute Name === *” id ”* ) {  p r i v a te I n s ta n c e . d a ta A ttr i b u te s [ attribute Name ] = p r i v a te I n s ta n c e . generateRandomId ( ) ; |
| 221 | } **else** { |
| 222 | p r i v a te I n s ta n c e . d a ta A ttr i b u te s [ attribute Name ] = *””* ; |

223

}

}

}

*// Construction d ’ une instance avec*

**return** i n s t a n c e I n t e r f a c e ;

} *// f i n de l a mé thode create Instance*

] ) ; *// f i n de l ’ appel ” apply ” de l a mé thode myApp. addModule*

224

225

226

227

228

229

230

### Utilisation avec l’affichage générique d’objets métier

Les méthodes de test d’implémentation d’interfaces de la partie [2.7](#_bookmark65) sont utilisables telles quelles. Pour adapter l’exemple de test d’affichage générique (code source [2.31](#_bookmark69)), il suffit de modifier l’inclusion des fichiers *JavaScript* comme suit :

Code Source 3.10 : Fichiers *JS* inclus dans ex06-fabriqueAdresseProtoTest.html

**<***!−− Cré ation de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . . /pattern−fonct/ex 05 −module Pattern Regex . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . /ex 05 −create Module Metier Proto . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . / ex 05 −fabrique Objet Metier Proto . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 06 −module Metier Adresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 08 −fabrique Adresse . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation Metier . j s ”***></s cript>**

**<***!−− Cré ation de f onct i ons d ’ a f f i ch ag e géné r i que dans myApp. metier . view −−***>**

**< ipt** */ t t .*

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**s cr src=***” . . pa tern−fonc / ex 12 −objet Metier View j s ”***></s cript>**

**<s cript src=***” . . / pattern−fonct/ ex 12 −objet Metier View Test . j s ”***></s cript>**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

Les fichiers du répertoire ../pattern-fonct correspondent aux exemples définis dans le cha- pitre [2](#_bookmark32), qui sont réutilisables tels quels du fait de la généricité et de l’homogénéité des interfaces d’objets métier.

* 1. **Patterns *pseudo-classique* (à éviter)**

##### Dans l’exemple suivant, nous créons une classe Personne qui hérite des propriétés de la classe

Adresse. Pour celà :

1. le constructeur d’Adresse est appelé explicitement dans le constructeur de Personne ;
2. la classe Adresse est déclarée comme superclass de la classe Personne ;
3. Les méthodes qui existent au niveau du prototype de la classe Adresse et qui doivent être spécifiées pour des personnes sont surchargées au niveau du prototype de la classe Personne.

Dans l’exemple suivant, nous surchargeons l’accesseur de la propriété ville et la méthode

toString.

Code Source 3.11 : /pattern-proto/ex08-extension-de-classe.js

1 **function** Adresse ( numeroRue , rue , complement , code Postal , v i l l e ) {

2 **i f** ( numeroRue . match ( */^([ 0 9 ]\*) (([ 0 9]+) (\ ?) ((( b i s ) | ( t er ) ) ) ?)$/* ) ) {

− −

1. **this** . numeroRue = numeroRue . r e p l a c e (/\ s+/g , *’ ’* ) ;
2. } **else** {
3. **throw new** Error ( *”Numéro de l a rue i n v a l i d e . ”* ) ; 6 }

7

8 **i f** ( rue . match ( */^ ( ( ( [ a−zA−ZêéèöàöÉÈÊÀÖË\ \− \.\ ,0 −9\ ] ) | (\ ” ) ) | (\ ’ ) ) {1 ,300 }$/* ) )

{

9 **this** . rue = rue . r e p l a c e (/\ s+/g , *’ ’* ) ; 10 ;

1. } **else** {
2. **throw new** Error ( *”Nom de l a rue/ p l ace i n v a l i d e . ”* ) ; 13 }

14

1. **i f** ( complement . match ( */^ ( ( ( [ a zA ZêéèöàöÉÈÊÀÖË\ \ \.\ ,0 9\ ] ) | (\ ” ) ) | (\ ’ ) )*

− − − −

*{0 ,300 }$/* ) ) {

1. **this** . complement = complement . r e p l a c e (/\ s+/g , *’ ’* ) ;
2. } **else** {
3. **throw new** Error ( *” Complement d ’ adresse i n v a l i d e . ”* ) ; 19 }

20

1. **i f** ( code Postal . match ( */^[ 0 9]{5}$/* ) ) {

−

1. **this** . code Postal = code Postal ;
2. } **else** {
3. **throw new** Error ( *” Code p os t a l i n v a l i d e . ”* ) ; 25 }

26

27 **i f** ( v i l l e . match ( */^ ( ( ( [ a−zA−ZêéèöàöÉÈÊÀÖË\ \− \.\ ,0 −9\ ] ) | (\ ” ) ) | (\ ’ ) ) {0 ,300 }$/* )

) {

1. **this** . v i l l e = v i l l e . r e p l a c e (/\ s+/g , *’ ’* ) ;
2. } **else** {
3. **throw new** Error ( *”Nom de v i l l e i n v a l i d e . ”* ) ; 31 }

32 }

33

1. Adresse . prototype . g e t V i l l e = **function** () {
2. **return this** . v i l l e ; 36 }

37

1. Adresse . prototype . to S tr i n g = **function** () {
2. **var** r e s u l t a t = **this** . numeroRue ;
3. **i f** ( **this** . numeroRue != *””* )
4. r e s u l t a t += *” , ”* ;
5. r e s u l t a t += **this** . rue + *” , ”* ;
6. r e s u l t a t += **this** . complement ;
7. **i f** ( **this** . complement != *””* )
8. r e s u l t a t += *” , ”* ;

46

}

r e s u l t a t += **this** . v i l l e + *”***<br** */***>***”* ;

**return** r e s u l t a t ;

**function** Personne (nom , prenom , numeroRue , rue , complement , code Postal , v i l l e ) { Adresse . c a l l ( **this** , numeroRue , rue , complement , code Postal , v i l l e ) ;

**this** . nom = nom ;

**this** . prenom = prenom ;

}

Personne . s u p e r c l a s s = Adresse ;

Personne . prototype . g e t V i l l e = **function** () {

**return** Adresse . prototype . g e t V i l l e . c a l l ( **this** ) ;

}

Personne . prototype . to S tr i n g = **function** () {

**return this** . nom+*” , ”*+**this** . prenom+*” , ”*+Adresse . prototype . to S tr i n g . c a l l ( **this** ) ;

}

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

Code Source 3.12 : /pattern-proto/ex08-extension-de-classe.html

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**Chainage de c o n s tr u c te u r s**</ t i t l e>**

**<s cript src=***” . /ex 08 −extension−de−classe . j s ”***></s cript>**

**</head>**

**<body>**

**<p>**

**<s cript> try** {

**var** pers **= new** Personne ( *” Dujardin ”* , *” Jean”* , *” 10 t er ”* , *” rue de l ’ avenir ”* , ’ *” Le Rastou”* ’ , *” 86098 ”* , *” Les Flots Bleus ”* ) ;

document . w r i te ( pers ) ;

document . w r i te ( *”***<br** */***>***L ’ adresse se t rouve dans l a v i l l e de \””*+pers . g e t V i l l e ( )+

*” \”. ”* ) ;

} **catch** ( e r r ) { a l e r t ( e r r ) ;

}

**</s cript>**

**<p>**

**</body>**

**</html>**

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

**Chapitre 4**

**Formulaires, Filtrage, *Pattern Mediator***

## Filtrage Basique des Inputs d’un Formulaire

##### L’exemple suivant montre comment filtrer les attributs d’un formulaires côté client en affichant immédiatement un message d’erreur lors de la saisie d’une valeur incorrecte. On associe à chaque événement onchange de chaque attribut une fonction *JavaScript* qui réalisera le filtrage.

Code Source 4.1 : /form-filter/ex01-basicForm.js

* 1. *// a l i a s vers l e module d ’ expressions r é g u l i è r e s*
  2. **var** r e g e x U t i l = myApp. metier . r e g e x U t i l ; 3

1. */\*\* @ description Gestionnaire d ’ événement change de l ’ input d ’ ID ” mainForm\_titre ” .*
2. *\* Cette mé thode e f f ec t u e l e f i l t r a g e par exepression r é g u l i è r e .*
3. *\* @method* **f i l t e r** *Titre*

7 *\*/*

1. **var f i l t e r** Ti tr e = **function** () {
2. **var** t i t r e V a l u e = $ ( *”#mainForm\_titre”* ) . val () ;
3. *// Expressions du l angage courant et c h i f f r e s*
4. **var** result Regex Test = r e g e x U t i l . test Regex Latin 1 With Digits ({
5. chaine : t i tr e V a l u e ,
6. minLength : 1 14 }) ;
7. *// Modification du contenu du span d ’ ID ” error\_main Form\_titre”*
8. **i f** ( result Regex Test === **true** ) {
9. $ ( *”#error\_main Form\_titre”* ) . empty ( ) ;
10. } **else** {
11. $ ( *”#error\_main Form\_titre”* ) . html (
12. *” Erreur : l e t i t r e ne d o i t contenir que l e s l e t t r e s et c h i f f r e s***<br** */***>***”* ) ; 21 }

22 } ;

23

1. */\*\* @ description Gestionnaire d ’ événement onchange de l ’ input d ’ ID ” mainForm\_resume ” .*
2. *\* Cette mé thode e f f ec t u e l e f i l t r a g e par exepression r é g u l i è r e .*
3. *\* @ function* **f i l t e r** *Resume*

27 *\*/*

28 **var f i l t e r** Resume = **function** ( ) {

29

} ;

**var** t i t r e V a l u e = $ ( *”#mainForm\_resume”* ) . val ( ) ;

*// Expressions du l angage courant et c h i f f r e s et ponctuation*

**var** result Regex Test = r e g e x U t i l . test Regex Latin 1 With Digits Punctuation ({ chaine : t i tr e V a l u e ,

minLength : 1

}) ;

*// Modification du contenu du span d ’ ID ” error\_mainForm\_resume”*

**i f** ( result Regex Test === **true** ) {

$ ( *”#error\_mainForm\_resume”* ) . empty ( ) ;

} **else** {

$ ( *”#error\_mainForm\_resume”* ) . html (

*” Erreur : l e r ésumé ne d o i t contenir que l e s l e t t r e s et c h i f f r e s ”* +

*” ou des caractères de ponctuation***<br** */***>***”* ) ;

}

30

31

32

33

34

35

36

37

38

39

40

41

42

43



Figure 4.1 : Illustration du code source [4.2](#_bookmark85)

Code Source 4.2 : /form-filter/ex01-basicFormTest.html (cf. Fig [4.1](#_bookmark84))

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**F i l t r a g e d ’ i np uts**</ t i t l e>**

**<l i nk rel=***” s t y l e s h e e t ”* **href=***” b as i c S t y l e . css ”* /**>**

**</head>**

**<body>**

**<h1>**S a i s i e d ’ un f i l m**</h1>**

**<form id=***”mainForm”* **action=***” post ”***>**

**<***!−− input avec g es t i on n a i re de l ’ événement onchange −−***>**

**<span id=***” error\_mainForm\_titre”* **class=** *” errorMsg”***></span>**

**<label for=***” mainForm\_titre”***>**Ti tr e :**</ label>**

**<i nput type=***” t ex t ”* **id=***” mainForm\_titre”* **size=***” 15 ”* **placeholder=***” Titre du f i lm ”* onchange= *”* **f i l t e r** *Titre () ”* /**><br** /**>**

**<***!−− t ex t area avec g es t i on n a i re de l ’ événement onchange −−***>**

**<span id=***” error\_mainForm\_resume”* **class=** *” errorMsg ”***></span>**

**<label for=***” mainForm\_resume”***>**Résumé :**</ label>**

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

**<textarea id=***” mainForm\_resume”* **rows=***” 10 ”* **cols=***” 50 ”*

**placeholder=***” S a i s i s s e z votre r ésumé”* onchange= *”* **f i l t e r** *Resume() ”***></textarea>**

**</form>**

**<***!−− Inclusion de l a s t ru c t u r e d ’ ap p l i cat i on et du module r eg ex U t i l −−***>**

**<s cript src=***” modules Metier . j s ”***></s cript>**

**<***!−− Pattern Mediator sp é c i a l i s é pour l e s événement ” change” des* **<i nput>** *−−***>**

**<s cript src=***” ex 02 −mediator Input Filter . j s ”***></s cript>**

**<s cript src=***” j query . j s ”***></s cript> <s cript src=***” ex01−basicForm . j s ”***></s cript>**

**<**

**<***!−− Inclusion de jQuery pour l e s événements et manipulation du DOM −−***>**

**</html>**

**/body>**

21

22

23

24

25

26

27

28

29

30

* 1. ***Pattern Mediator* pour le filtrage d’attributs**

##### L’inconvénient du filtrage présenté dans la partie [4.1](#_bookmark83) est que, dans le code *HTML* d’un champs du formulaire lui-même, on doit déclarer une méthode de filtrage spécifique pour ce champs (attribut onchange de l’élément *HTML* input ou textarea).

Dans l’architecture d’application que nous proposons par la suite, la méthode de filtrage ne sera pas codée en dûr dans le module chargé de générer le formulaire, mais plutôt dans les tests d’expressions régulières effectués dans les modules métier. En particulier, la méthode précise dépendra de l’instance et de l’attribut considéré, ce qui entraînerait un fort *couplage*

(interdépendance) des méthodes chargées de l’*IHM* et des classes métier (ou du *modèle*).

Nous savons par expérience que ce type de couplage va provoquer des difficultés pour la maintenance et l’évolution de notre application (comme par exemple la migration de nos objet métier côté serveur avec *NodeJS*). Nous allons maintenant introduire un *pattern* qui a pour vocation de découpler le déclenchement des événements (via, en l’occurence, des événements utilisateurs onchange) de l’implémentation des opérations correspondantes sur les données

métier, ou les données du modèle. Ce pattern est une généralisation du pattern *Observer*.

Dans notre exemple, un module *Mediator* va enregistrer les méthodes *callbacks* (qui ne sont que des fonctions *JavaScript*) associées à des événements. L’exécution des ces claabacks (en l’occurence la réaction à un événement onchange) sera déclenchée par la publication de l’événement en question par l’intermédiaire du *Mediator*.

Code Source 4.3 : /form-filter/ex02-mediatorInputFilter.js

|  |  |
| --- | --- |
| 1 |  |
| 2 | */\*\** |
| 3 | *\* Ajout d ’ un module c t r l ( contr ô l eu r s ) à l ’ ap p l i cat i on .* |
| 4 | *\* @module c t r l* |
| 5 | *\* @augments myApp* |
| 6 | *\*/* |
| 7 | myApp. addModule . apply (myApp, [ *” c t r l ”* , {}] ) ; |
| 8 |  |
| 9  10 | */\*\**  *\* Impl é mentation du pattern ”Mé d i at or ” pour gé rer l e f i l t r a g e des inputs de formulaires .* |
| 11 | *\* @module mediator Input Filter* |
| 12 | *\* @augments myApp. c t r l* |
| 13 | *\*/* |
| 14 | myApp. addModule . apply (myApp. c t r l , [ *” mediator Input Filter ”* , **function** () { |
| 15 |  |
| 16 | *////////////////////////////////////////////////* |

17 // Propri é t é s et mé thodes ” s t a t i q u e s ” priv é es

18

19 */\*\**

1. *\* Collection , index ée par ID de formulaire de cal l b ac k s d ’ événements l i é s à d i f f é rens formulaires ( typiquement : événement onchange d ’ un input ) .*
2. *\* @ private*

22 *\*/*

23 **var** m\_ subscription Lists ; 24

25 */\*\**

1. *\* I n i t i a l i s e ( ou r é i n i t i a l i s e ) l ’ ensemble des l i s t e s d ’ événements à l a c o l l e c t i o n v ide .*
2. *\* @ private*

28 *\*/*

1. **var** i n i t = **function** () {
2. m\_ subscription Lists ={} ; 31 } ;

32

1. *// I n n i t i a l i s e r une f o i s l ’ ensemble des l i s t e s d ’ événements à l a c o l l e c t i o n v ide .*
2. i n i t ( ) ; 35

36 ////////////////////////////////////////////////

37 // In t er f ace publique du module

38

39 */\*\**

1. *\* Cré ation d ’ un ob j e t contenant l e s donné es et mé thodes pu b l i q u es*
2. *\* ( l e s propri é t é s pu b l i q u es sont retourn é es par l a fonction ” module ”) .*

42 *\*/*

43 **var** p u b l i c I n te r f a c e M e d i a to r = { 44

45 */\*\**

1. *\* Ajoute un formulaire et l a l i s t e ( i n i t i a l em en t v ide ) de ses cal l b ack s associ é s .*
2. *\* Si l e formulaire es t dé j à gér é , l a l i s t e de ses cal l b ack s associ é s es t supprim ée et r é i n i t i a l i s ée à l a l i s t e v ide .*
3. *\* @param { s t r i n g } form Id l ’ Id du formulaire ( en t ant qu ’ é l ément HTML)*

−

49 *\*/*

1. addForm : **function** ( form Id ) {
2. m\_ subscription Lists [ form Id ] = {} 52 } ,

53

54 */\*\**

1. *\* Supprime un formulaire et ses cal l b ac k s associ é s*
2. *\* @param { s t r i n g } form Id l ’ Id du formulaire ( en t ant qu ’ é l ément HTML)*

−

57 *\*/*

1. removeForm : **function** ( form Id ) {
2. **i f** ( ! m\_ subscription Lists . hasOwnProperty ( form Id ) ) {
3. **return f al s e** ; 61 }
4. **delete** m\_ subscription Lists [ form Id ] ;
5. **return true** ; 64 } ,

65

66 */\*\**

67 \* Ajout d ’ un événement associ é à un a t t r i b u t de formulaire et de sa

|  |  |
| --- | --- |
|  | *fonction cal l b ack .* |
| 68 | *\* Si l ’ événement e x i s t a i t dé j à pour cet input , i l es t é cras é .* |
| 69 | *\* @param { s t r i n g } form Id − l ’ Id du formulaire ( en t ant qu ’ é l ément HTML)* |
| 70 | *\* @param { s t r i n g } inputName − l e nom de l ’ input ( ou de l a propri é t é de l ’*  *ob j e t mé t i e r associ é ) .* |
| 71  72 | *\* @param { function } callback Function − l a fonction ( cal l b ack ) à appeler en cas de p u b l i cat i on de l ’ événement .*  *\*/* |
| 73 | s u b s c r i b e : **function** ( formId , inputName , c a l l b a c k Fu n c t i o n ) { |
| 74 | **i f** ( m\_ subscription Lists . hasOwnProperty ( form Id ) ) { |
| 75 | m\_ subscription Lists [ form Id ] [ inputName ] = { c a l l b a c k : c a l l b a c k Fu n c t i o n } ; |
| 76 | } **else** { |
| 77 | **throw** {name : *” I l l egal Argument Exception ”* , |
| 78 | message : *” Caté gorie d ’ événements ”* + eventCateg + *” inconnue du mé* |
|  | *d i ateur ”* |
| 79 | } ; |
| 80 | } |
| 81 | } , |
| 82 |  |
| 83 | */\*\** |
| 84 | *\* Publication d ’ un événement associ é à un a t t r i b u t de formulaire provoquant* |
|  | *l ’ ex é cution de l a fonction cal l b ack associ ée* |
| 85 | *\* @param { s t r i n g } form Id − l ’ Id du formulaire ( en t ant qu ’ é l ément HTML)* |
| 86 | *\* @param { s t r i n g } inputName − l e nom de l ’ input ( ou de l a propri é t é de l ’*  *ob j e t mé t i e r associ é ) .* |
| 87 | *\*/* |
| 88 | p u b l i s h : **function** ( formId , inputName ) { |
| 89 |  |
| 90  91  92 | **i f** ( m\_ subscription Lists . hasOwnProperty ( form Id ) ) {  **i f** ( m\_ subscription Lists [ form Id ] . hasOwnProperty ( inputName ) ) {  *// On ap p e l l e l e cal l b ak avec son* |
| 93 | m\_ subscription Lists [ form Id ] [ inputName ] . c a l l b a c k ( ) ; |
| 94 | } |
| 95 | } **else** { |
| 96 | **throw** {name : *” I l l egal Argument Exception ”* , |
| 97 | message : *” Formulaire d ’ ID ”* + form Id + *” inconnu du mé d i ateur ”* |
| 98 | } ; |
| 99 | } |
| 100 | } , |
| 101 |  |
| 102 | */\*\** |
| 103 | *\* Ré i n i t i a l i s e l a c o l l e c t i o n des formulaires gér é s à une c o l l e c t i o n v ide .* |
| 104 | *\*/* |
| 105 | empty : **function** ( ) { |
| 106 | i n i t () ; |
| 107 | } |
| 108 | } ; |
| 109 |  |
| 110 | **return** p u b l i c I n te r f a c e M e d i a to r ; |
| 111 |  |
| 112 | }() ] ) ; |

##### Code Source 4.4 : /form-filter/ex02-mediatorInputFilterTest.html

1

**<!doctype HTML>**

**<html lang=***” f r ”***>**

2

1. **<head>**
2. **<meta charset=***”UTF 8”* /**>**

−

1. **<t i t l e>**F i l t r a g e d ’ i np uts**</ t i t l e>**
2. **<l i nk rel=***” s t y l e s h e e t ”* **href=***” b as i c S t y l e . css ”* /**>**
3. **</head>**
4. **<body>**
5. **<h1>**S a i s i e d ’ un f i l m**</h1>**
6. **<form id=***”mainForm”* **action=***” post ”***>**
7. **<***! input avec g es t i on n a i re de l ’ événement onchange* **>**

−− −−

1. **<span id=***” error\_mainForm\_titre”* **class=** *” errorMsg”***></span>**
2. **<label for=***” mainForm\_titre”***>**Ti tr e :**</ label>**
3. **<i nput type=***” t ex t ”* **id=***” mainForm\_titre”* **size=***” 15 ”*
4. **placeholder=***” Titre du f i lm ”* onchange= *”* **f i l t e r** *Data ( ’ mainForm ’ , ’ t i t r e ’ ) ”* /**><br** /

>

1. **<***! t ex t area avec g es t i on n a i re de l ’ événement onchange* **>**

−− −−

1. **<span id=***” error\_mainForm\_resume”* **class=** *” errorMsg ”***></span>**
2. **<label for=***” mainForm\_resume”***>**Résumé :**</ label>**
3. **<textarea id=***” mainForm\_resume”* **rows=***” 10 ”* **cols=***” 50 ”*
4. **placeholder=***” S a i s i s s e z votre r ésumé”* onchange= *”* **f i l t e r** *Data ( ’ mainForm ’ , ’ resume ’ ) ”***></textarea>**
5. </form>
6. **<***! Inclusion de l a s t ru c t u r e d ’ ap p l i cat i on et du module r eg ex U t i l* **>**

−− −−

1. **<s cript src=***” modules Metier . j s ”***></s cript>**
2. **<***! Inclusion de jQuery pour l e s événements et manipulation du DOM* **>**

−− −−

1. **<s cript src=***” j query . j s ”***></s cript>**
2. **<s cript src=***” ex 02 mediator Input Filter . j s ”***></s cript>**

−

1. **<s cript src=***” ex01 basicForm . j s ”***></s cript>**

−

1. <s cript>
2. *// Ajout du formulaire ”mainForm” au mé d i ateur qui gèrera ses événements .*
3. myApp. c t r l . m e d i a to r In p u t Fi l te r . addForm ( ’ mainForm ’ ) ;
4. *// Enregistrement du cal l b ack asscoci é à l ’ événement onchange du t i t r e*
5. myApp. c t r l . m e d i a to r In p u t Fi l te r . s u b s c r i b e ( ’ mainForm ’ , ’ t i t r e ’ , **f i l t e r** Ti tr e ) ;
6. *// Enregistrement du cal l b ack asscoci é à l ’ événement onchange du r ésumé*
7. myApp. c t r l . m e d i a to r In p u t Fi l te r . s u b s c r i b e ( ’ mainForm ’ , ’ resume ’ , **f i l t e r** Resume )

;

35

1. */\*\* @ description Publie l ’ événement onchange d ’ un input auprès du mé diateur ,*
2. *\* provoquant l ’ ex é cution du cal l b ack en r eg i s t r é pour cet événement .*
3. *\* @ function* **f i l t e r** *Data*

39 *\*/*

1. **var f i l t e r Data = function** ( formId , inputName ) {
2. myApp. c t r l . m e d i a to r In p u t Fi l te r . p u b l i s h ( formId , inputName ) ; 42 } ;
3. **</s cript>**
4. **</body>**
5. **</html>**

## Exemple :

**Génération automatique de formulaire d’adresse**

Dans l’exemple suivant, des méthodes d’un module myApp.gui permettent de générer auto- matiquement les inputs d’un formulaire permettant de saisir les attributs (ici supposées de type texte) d’un objet qui implémente des interface qui apparaissent dans l’exemple de la

partie [2.7.1](#_bookmark66).

Nous appliquons cette méthode pour afficher et filtrer automatiquement un formulaire de saisie d’une adresse.

Code Source 4.5 : /form-filter/ex03-formsGui.js

1. */\*\* @ description Module ” gui ” ( pour Graphical User In t er f ace )*
2. *\* @module gui*
3. *\* @augments myApp*

4 *\*/*

1. myApp. addModule . apply (myApp, [ *” gui ”* , {}] ) ; 6
2. */\*\* @ description Géné ration de l ’ ID d ’ un é l ément HTML de type input pr é f i x é par l ’ ID du formulaire*
3. *\* @method get Input Id*
4. *\* @augments myApp. gui*
5. *\* @param { Object } input Spec contient l e s sp é c i f i c a t i o n s de l ’ input*

−

1. *\* @param { s t r i n g } input Spec . form Id id du formulaire dans l eq u e l l ’ input sera ins ér é*

−

1. *\* @param { s t r i n g } input Spec . attribute Name nom de l ’ a t t r i b u t de input Spec . objet Metier*

−

1. *\* à s a i s i r dans l ’ input*

14 *\*/*

1. myApp. addModule . apply (myApp. gui , [ *” get Input Id ”* , **function** ( input Spec ) {
2. **return** input Spec . form Id + *”\_”* + input Spec . attribute Name ; 17 } ] ) ;

18

1. */\*\* @ description Publie auprès du Mediator un événement onchange d ’ un Input*
2. *\* @method publish Input Change*
3. *\* @augments myApp. gui*
4. *\* @param { s t r i n g } form Id id du formulaire dans l eq u e l l ’ input sera ins ér é*
5. *\* @param { s t r i n g } attribute Name nom de l ’ a t t r i b u t de input Spec . objet Metier à s a i s i r dans l ’ input*

24 *\*/*

1. myApp. addModule . apply (myApp. gui , [ *” publish Input Change ”* , **function** ( formId , attribute Name ) {
2. myApp. c t r l . m e d i a to r In p u t Fi l te r . p u b l i s h ( formId , attribute Name ) ; 27 } ] ) ;

28

1. */\*\*@ description Géné ration du code HTML d ’ un input .*
2. *\* @method get Text Input Code*
3. *\* @augments myApp. gui*
4. *\* @param { Object } input Spec contient l e s sp é c i f i c a t i o n s de l ’ input*
5. *\* @param { Object } input Spec . instance instance d ’ ob j e t mé t i e r qui impl émente :*

−

1. *\* ” get Module ” , ” g e t A t t r i b u t e ” , ” s e t A t t r i b u t e ” ,*
2. *\* ” has Error ” , ” get Error Message ” , ” get Error List ”*
3. *\* a i n s i que l e s mé thodes s t a t i q u e s :*
4. *\* ” g e t A t t r i b u t e Li s t ” , ” get Label Text ” , ” t est Regex ” ,*
5. *\* ” create Instance ” , ” get Module ” .*
6. *\* @param { s t r i n g } input Spec . form Id id du formulaire dans l e q u e l l ’ input sera ins ér é*
7. *\* @param { s t r i n g } input Spec . attribute Name nom de l ’ a t t r i b u t de input Spec . objet Metier à s a i s i r dans l ’ input*
8. *\* @param { s t r i n g } [ input Spec . type=t ex t ] type de l ’ input*
9. *\* @param {number} [ input Spec . input S i z e =10] t a i l l e de l ’ input ( nombre de caractères )*

43 *\*/*

1. myApp. addModule . apply (myApp. gui , [ *” get Input Code”* , **function** ( input Spec ) {
2. *// Calcul de l ’ ID de l ’ input :*
3. **var** input Id = myApp. gui . get Input Id ( input Spec ) ; 47
4. *// Valeur de l ’ a t t r i b u t de l ’ ob j e t pour l ’ a t t r i b u t value de l ’ input*
5. **var** a ttr i b u te V a l u e = input Spec . o b je t M e t i e r . g e t A ttr i b u te ( input Spec . attribute Name ) | | *””* ;
6. *// Cré ation d ’ un é ventuel message s i l ’ ob j e t comportait dé j à une erreur*
7. **var** error Message = input Spec . o b je t M e t i e r . get Error Message ( input Spec . attribute Name ) !== undefined
8. ? input Spec . o b je t M e t i e r . get Error Message ( input Spec . attribute Name

) + *”***<br** */***>***”* : *””* ;

53

54 //////////////////////////////////////////////////////////////////////

1. *// Callback de g est i on du f i l t r a g e de l ’ input :*
2. myApp. c t r l . m e d i a to r In p u t Fi l te r . s u b s c r i b e ( input Spec . formId , input Spec . attribute Name , **function** () {

57

1. **var** r e s u l ta t Te s t Re g e x = input Spec . o b je t M e t i e r . test Regex ( input Spec . attribute Name ,
2. document . getElementBy Id ( input Id ) . value ) ;
3. **i f** ( r e s u l ta t Te s t Re g e x !== **true** ) {
4. document . getElementBy Id ( *” error\_”*+input Id ) . innerHTML = r e s u l ta t Te s t Re g e x

+*”***<br** */***>***”* ;

1. } **else** {
2. document . getElementBy Id ( *” error\_”*+input Id ) . innerHTML = *””* ; 64 }

65 }) ; // f i n du cal l b ack //////////////////////////////////////////////

66

1. **var** inputType = input Spec . inputType === undefined ? *” t ex t ”* : input Spec . inputType ;
2. **var** i n p u t S i z e = input Spec . i n p u t S i z e === undefined ? *” 10 ”* : input Spec . i n p u t S i z e

;

1. **var** l a b e l Te xt = input Spec . o b je t M e t i e r . get Label Text ( input Spec . attribute Name ) ; 70
2. *// retour du code HTML de l ’ input*
3. **return** *”***<span** *c l as s =\” errorMsg \” id=\” error\_”*+input Id+*”\”***>***”* + error Message + *”*

**</span>***”* +

1. *”***<l abel** *for=\””* + input Spec . attribute Name + *”\”***>***”* + l a b e l Te xt + *”***</ label>***”*

+

1. *”***<i nput** *type=\””* + inputType + *”\” name=\””* + input Spec . attribute Name +
2. *”\” id=\””* + input Id + *”\” ”* + *” value=\””* + a ttr i b u te V a l u e + *”\” ”* +
3. *”* **size** *=\””* + i n p u t S i z e + *”\” ”* +
4. *” onchange=\”myApp. gui . publish Input Change ( ’ ”* + input Spec . form Id + *” ’ , ’ ”* + input Spec . attribute Name + *” ’) \” ”* + *” /***>***”* ;

78 } ] ) ;

79

80 /\*\* @ description Géné ration du code HTML de l ’ ensemble des inputs d ’ un formulaire .

81 *\**

1. *\* @method get Html Form Inputs*
2. *\* @augments myApp. gui*
3. *\* @ public*
4. *\* @param { Object } instance instance d ’ ob j e t mé t i e r qui impl émente :*

−

1. *\* ” get Module ” , ” g e t A t t r i b u t e ” , ” s e t A t t r i b u t e ” ,*

|  |  |
| --- | --- |
| 87 | *\* ” has Error ” , ” get Error Message ” , ” get Error List ”* |
| 88 | *\* a i n s i que l e s mé thodes s t a t i q u e s :* |
| 89 | *\* ” g e t A t t r i b u t e Li s t ” , ” get Label Text ” , ” t est Regex ” ,* |
| 90 | *\* ” create Instance ” , ” get Module ” .* |
| 91 | *\* @param { s t r i n g } form Id id du formulaire dans l eq u e l l ’ input sera ins ér é* |
| 92  93 | *\* @return { s t r i n g } l e code HTML des tous l e s inputs correspondant aux propri é t é s de l ’ ob j e t mé t i e r .*  *\*/* |
| 94 | myApp. addModule . apply (myApp. gui , [ *” get Html Form Inputs”* , **function** ( objet Metier , |
|  | form Id ) { |
| 95 | *// On vé r i f i e que l ’ objet Metier impl émente b ien notre i n t er f ace géné r i que .* |
| 96 | **var** t e s t I n t e r f a c e s M e t i e r = myApp. metier . te s t I n te r f a c e I m p l e m e n ta t i o n ( |
|  | o b je t M e t i e r ) ; |
| 97 | **i f** ( t e s t I n t e r f a c e s M e t i e r !== **true** ) { |
| 98 | **throw new** Error ( t e s t I n t e r f a c e s M e t i e r ) ; |
| 99 | } |
| 100 |  |
| 101 | *// Ajour du formulaire ”mainForm” au mé d i ateur qui gèrera ses événements .* |
| 102 | myApp. c t r l . m e d i a to r In p u t Fi l te r . addForm ( form Id ) ; |
| 103 |  |
| 104 | **var** htmlCode = *””* ; |
| 105 |  |
| 106 | **var** a t t r i b u t e L i s t = o b je t M e t i e r . g e t A t tr i b u te L i s t ( ) ; |
| 107 |  |
| 108 | *// Tous l e s inputs sont de type t exte , donc on peut* |
| 109 | *// f a i r e une boucle automatique sur l e s propri é t é s .* |
| 110 | **for** ( **var** i =0 ; i **<** a t t r i b u t e L i s t . l e n g th ; i++){ |
| 111 | **var** attribute Name = a t t r i b u t e L i s t [ i ] ; |
| 112  113  114  115 | *// l ’ u t i l i s a t e u r ne peut pas modifier l ’ ID :*  **i f** ( attribute Name != *” id ”* ) {  *// Concaté nation du code HTML de l ’ input*  htmlCode += myApp. gui . getInputCode ({ |
| 116 | o b je t M e t i e r : objet Metier , |
| 117 | attribute Name : a t t r i b u t e L i s t [ i ] , |
| 118 | l a b e l Te xt : o b je t M e t i e r . get Label Text ( a t t r i b u t e L i s t [ i ] ) , |
| 119 | form Id : form Id |
| 120 | }) + *”***<br** */***>***”* ; |
| 121 | } |
| 122 | } |
| 123 |  |
| 124 | *// champs cach é repr é sentant l ’ ID de l ’ instance* |
| 125 | htmlCode += *”***<i nput** *type=\” hidden \” id=\””* + form Id + *”\_id\” value=\””* + |
| 126 | o b je t M e t i e r . g e t A ttr i b u te ( *” id ”* ) + *”\” /***>***”* ; |
| 127 |  |
| 128 | **return** htmlCode ; |
| 129 | } ] ) ; |

##### Code Source 4.6 : /form-filter/ex03-formsGuiTest.js (cf. Fig [4.2](#_bookmark89))

1

*// Ajout d ’ une mé thode mainFunction*

myApp. addModule ( *” mainFunction”* , **function** () {

*// cr é ation d ’ une instance*

**var** a d r e s s e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ id : *” 04 abf 85 bc 9 ”* ,

numeroRue : *” 2 bis@”* ,

2

3

4

5

6

7



Figure 4.2 : Illustration du code source [4.6](#_bookmark88)

8

rue : *”Rue de l ’ a Paix ”* , complementAddr : *”Bâtiment 3D”* , code Postal : *” 63000 ”* ,

pays : *” France”*

}) ;

v i l l e : *” Clermont−Ferrand”* ,

*// Géné ration du formulaire avec l e s cal l b ac k s*

codeHTML = *”***<form** *id=\”mainForm\” method=\” post \”***>***”* +

myApp. gui . getHtmlFormInputs ( adresse , *”mainForm”* ) +

*”***<i nput** *type=\” submit \” value=\” v a l i d er \” /***>***”* +

*”***</form>***”* ;

*// U t i l i s a t i o n de l a valeur retourn ée pour géné rer l a vue*

document . getElementBy Id ( *” paragraphe Resultat ”* ) . innerHTML = codeHTML ;

}) ;

*// Exé cution de l a mé thode mainFunction*

myApp. mainFunction ( ) ;

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

Code Source 4.7 : Fichiers *JS* inclus dans ex03-formsGuiTest.html

**<***!−− Structure de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 04 −structure Application . j s ”***></s cript>**

**<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**

**<s cript src=***” . . /pattern−fonct/ex 05 −module Pattern Regex . j s ”***></s cript>**

**<***!−− Sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 06 −create Module Metier . j s ”***></s cript>**

**<***!−− Mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 07 −fabrique Objet Metier . j s ”***></s cript>**

**<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 06 −module Metier Adresse . j s ”***></s cript>**

**<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 08 −fabrique Adresse . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation . j s ”***></s cript>**

**<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**

**<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation Metier . j s ”***></s cript>**

**<***!−− Fonctions d ’ a f f i ch ag e géné r i que dans myApp. metier . view −−***>**

**< ipt** *t e i* **pt**

*−−*

**s cr src=***” . . / pattern−fonc / ex 12 −obj t Met er View . j s ”***></s cri >**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

**<***!−− Inclusion de jQuery pour l e s événements et manipulation du DOM −−***>**

**<s cript src=***” j query . j s ”***></s cript>**

**<***!−− Mediator sp é c i a l i s é pour f i l t r e r l e s inputs ( evt ” change ”) −−***>**

**<s cript src=***” ex 02 −mediator Input Filter . j s ”***></s cript>**

**<***!−− Géné ration automatique de formulaires avec f i l t r a g e des a t t r i b u t s −−***>**

*−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**

**<s cript src=***” ex03−formsGui . j s ”***></s cript>**

**<s cript src=***” ex03−formsGuiTest . j s ”***></s cript>**

21

22

23

24

25

26

27

28

Le test avec la modélisation des objets métiers de la partie [3.3](#_bookmark77), utilisant la notion de prototype, diffère seulement au niveau de l’inclusion de la définition des modules métier dans le fichier *HTML* :

Code Source 4.8 : Fichiers *JS* inclus dans ex03-formsGuiTestProto.html

* 1. **<***!−− Structure de l ’ ap p l i cat i on v ide avec deux mé thodes −−***>**
  2. **<s cript src=***” . . / pattern−fonct/ ex 04 −structure Application . j s ”***></s cript>**
  3. **<***!−− Cré ation de sous−module r eg ex U t i l de myApp. metier −−***>**
  4. **<s cript src=***” . . /pattern−fonct/ex 05 −module Pattern Regex . j s ”***></s cript>**
  5. **<***!−− Sous−module adresse de myApp. metier −−***>**
  6. **<s cript src=***” . . / pattern−proto/ ex 05 −create Module Metier Proto . j s ”***></s cript>**
  7. **<***!−− Mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r −−***>**
  8. **<s cript src=***” . . / pattern−proto/ ex 05 −fabrique Objet Metier Proto . j s ”***></s cript>**
  9. **<***!−− Cré ation de sous−module adresse de myApp. metier −−***>**
  10. **<s cript src=***” . . / pattern−fonct/ ex 06 −module Metier Adresse . j s ”***></s cript>**
  11. **<***!−− Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse −−***>**
  12. **<s cript src=***” . . / pattern−fonct/ ex 08 −fabrique Adresse . j s ”***></s cript>**
  13. **<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**
  14. **<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation . j s ”***></s cript>**
  15. **<***!−− Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces −−***>**
  16. **<s cript src=***” . . / pattern−fonct/ ex 11 −interface Implementation Metier . j s ”***></s cript>**
  17. **<***!−− Fonctions d ’ a f f i ch ag e géné r i que dans myApp. metier . view −−***>**
  18. **<s cript src=***” . . / pattern−fonct/ ex 12 −objet Metier View . j s ”***></s cript>**
  19. *−−***<***! Inclusion de jQuery pour l e s événements et manipulation du DOM* **>**

−− −−

20

1. **<s cript src=***” j query . j s ”***></s cript>**
2. **<***!−− Mediator sp é c i a l i s é pour f i l t r e r l e s inputs ( evt ” change ”) −−***>**
3. **<s cript src=***” ex 02 −mediator Input Filter . j s ”***></s cript>**
4. **<***!−− Géné ration automatique de formulaires avec f i l t r a g e des a t t r i b u t s −−***>**
5. **<s cript src=***” ex03−formsGui . j s ”***></s cript>**

26

1. *−−***<***!−− Ajout d ’ une fonction de t est , d ’ une mé thode ” main” , et ex é cution −−***>**
2. **<s cript src=***” ex03−formsGuiTest . j s ”***></s cript>**

# Chapitre 5

**Exemple d’Application avec *IHM***

## Principe de l’application et analyse fonctionnelle

##### Notre application, qui possède un *modèle* constitué d’une collection de personnes, permet (voir les *storyboards* sur la figure [5.1](#_bookmark93)) :

* + - D’afficher la liste des noms de personnes (*items*) ;

De sélectionner une personne en cliquant sur l’*item* correspondant (l’*item* est alors surli- gné et les détails concernant cette personne sont affichés) ;

*•*

De modifier les données de la personnes (en l’occurence le nom) en cliquant sur un bouton ”*Modifier*”.

*•*

* + - D’ajouter une personne ;
    - De supprimer la personne sélectionnée.
    - d’ajouter, de supprimer ou de modifier une adresse pour la personne sélectionnée.

Comme on peut le voir, nous avons une *agrégation* entre les personnes et les adresses, une personne pouvant avoir plusieurs adresses.

En recenssant les événements (*clics* de boutons d’*items*, liens) possibles sur les *storyboards*

##### de la figure [5.1](#_bookmark93), on dresse le diagramme de cas d’utilisation représenté sur la figure [5.2](#_bookmark94).

## Modèle de donnée

Dans notre modèle de données, une classe personne comporte un nom et une composition avec des instances d’adresse. Nous créons, pour le moment, quelques instances ”en dur”, dans un tableau personnes, avec chacune une adresse. Une autre propriété selectedPersonne contient une référence vers l’instance de personne sélectionnée (*item* surligné et détails affichés).

Code Source 5.1 : /ihm-demo/modelModule.js

1

myApp. addModule . apply (myApp, [ *” modele”* , { s e l e c te d Pe r s o n n e : **null** ,

personnes : [ ] ,

2

3

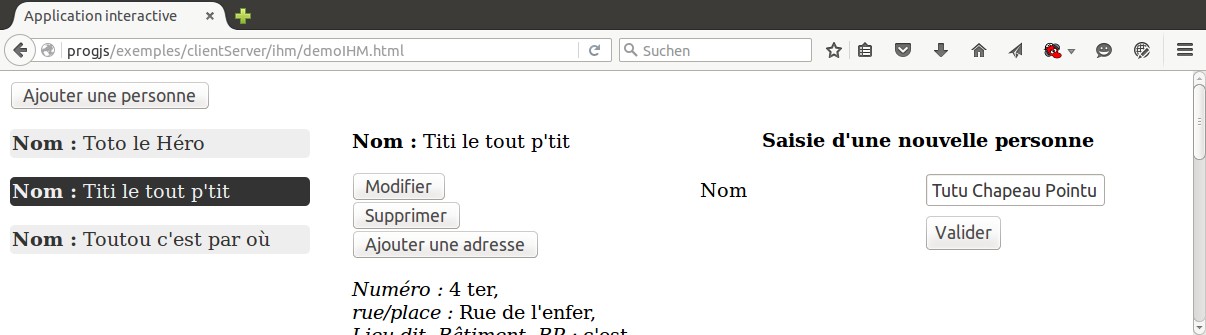
4



1. Sélection d’une personne (*item* surligné à gauche)



1. Ajout d’une adresse pour la personne sélectionnée



1. Ajout d’une personne



1. Après ajout d’une personne

Figure 5.1 : Captures d’écran de notre application

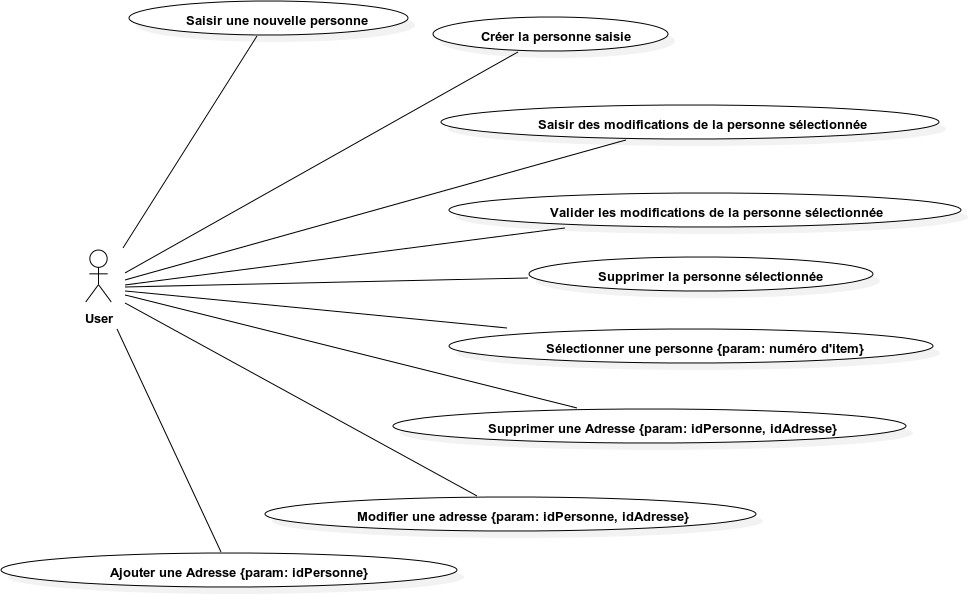


Figure 5.2 : Diagramme de cas d’utilisation de notre application avec son *IHM*

5 } ] ) ;

6

7 myApp. modele . personnes . push (myApp. metier . personne . c r e a te I n s ta n c e ({ 8 id : *” 0123 abcdef ”* ,

9 nom : *” Toto l e Héro”* ,

10 a d r e s s e : myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 11 id : *” 04 abf 85 bc 9 ”* ,

1. numeroRue : *” 2 b i s ”* ,
2. rue : *”Rue de l ’ a Paix”* ,
3. complementAddr : *” Dalle ”* ,
4. code Postal : *” 630000 ”* ,
5. v i l l e : *” Clermont Ferrand”* ,

−

1. pays : *” France 2 ”*

18 })

19 }) ) ;

20

21 myApp. modele . personnes . push (myApp. metier . personne . c r e a te I n s ta n c e ({ 22 id : *” 0123 abcd 12 ”* ,

1. nom : *” Tit i l e t out p ’ t i t ”* ,
2. a d r e s s e : myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ 25 id : *” 04 abf 85 bb 5 ”* ,
3. numeroRue : *” 4 t er ”* ,
4. rue : *”Rue de l ’ enfer ”* ,
5. complementAddr : *” c ’ es t l e s autres . . . ”* ,
6. code Postal : *” 75000 ”* ,
7. v i l l e : *” Ris pas”* ,
8. pays : *” France”*

32 })

33

}) ) ;

myApp. modele . personnes . push (myApp. metier . personne . c r e a te I n s ta n c e ({ id : *” 0123 abcd 01 ”* ,

nom : *” Toutou c ’ es t par où”* ,

a d r e s s e : myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ id : *” 04 abf 85 ba 4 ”* ,

numeroRue : *” 1 ”* ,

rue : *” Place de l ’ Al t ern at i v e ”* , complementAddr : *” Pourquoi pas”* , code Postal : *” 63123 ”* ,

v i l l e : *” Les Paumiers”* , pays : *” France”*

})

}) ) ;

34

35

36

37

38

39

40

41

42

43

44

45

46

47

* 1. ***Pattern Mediator* : centraliser les événements**

##### Notre module mediator va nous permettre :

De découpler l’implémentation de la réaction aux événements utilisateurs (modification du modèle, mise à jour des vues) de la gestion de ces événements utilisateurs via la technologie *jQuery*, qui, de ce fait, se trouve circonscrite à une seule classe (*Wrapper*).

*•*

* + - D’éliminer les dépendances cycliques entre les modules de notre application ;
    - De recenser les événements utilisateurs de manière lisible dans un module centralisé ;

De provoquer des mises à jour de panneaux de la vue qui observent des propriétés du modèle.

*•*

Contrairement au médiateur spécialisé dans le filtrage des attributs de formulaires décrit dans la partie [4.2](#_bookmark86), le module mediator va nous permettre d’exécuter plusieurs *callback* en réaction à un même événement (par exemple pour mettre à jour différentes parties de la vue après une modification du modèle).

Code Source 5.2 : /ihm-demo/mediator.js

1

*/\*\**

*\* Impl é mentation du pattern ”Mé d i at or ” pour l a g est i on des événements u t i l i s a t e u r s ,*

*\*/*

myApp. addModule . apply (myApp. gui , [ *” mediator ”* , **function** () {

*/\*\**

* *List e des événements pour l e s q u e l s une l i s t e de c a l l b ac k s peut ê t re en r eg i s t r ée*
* *@ private*

*\*/*

**var** m\_ subscription Lists ;

*/\*\**

*\* et l a mise à jour des vues ( ou des sous−arbres du DOM)*

2

3

4

5

6

7

8

9

10

11

12

13

1. *\* I n i t i a l i s e l a l i s t e des événements , avec pour chacun , une l i s t e de cal l b ac k s v ide .*
2. *\* @method i n i t*

16 *\*/*

1. **var** i n i t = **function** () {
2. m\_ subscription Lists = { 19
3. *// Opé rations CRUD sur l e s personnes*
4. *” personne/ read ”* : [ ] , *// Lire t ou t es l e s personnes pour ( re ) onstru i re l e modèle*

22

1. *” personne/ update”* : [ ] , *// v a l i d a t i on du formulaire de mise à jour de l a personne s é l ect i on n ée .*
2. *” personne/ create ”* : [ ] , *// v a l i d a t i on du formulaire d ’ ajout d ’ une personne .*
3. *” personne/ d e l e t e ”* : [ ] , *// Suppression d ’ une personne*

26

1. *// Opé rations CRUD sur l e s adresses*
2. *” adresse / create ”* : [ ] , *// v a l i d a t i on du formulaire d ’ ajout d ’ une adresse .*
3. *” adresse / update”* : [ ] , *// mise à jour d ’ une adresse*
4. *” adresse / d e l e t e ”* : [ ] , *// Suppression d ’ une adresse*

31

1. *// Actions U t i l i s a t eu r donnant l i eu à un changement de l e vue*
2. *” personne/ s e l e c t D e t a i l s ”* : [ ] , *// Sé l ec t i on d ’ une personne pour voir l e s d é t a i l s*
3. *” personne/ ed i t ”* : [ ] , *// c l i c k sur l a modification de l a personne s é l ect i on n ée*
4. *” personne/ s a i s i e ”* : [ ] , *// c l i c k sur l a modification de l a personne s é l ect i on n ée*

36

1. *” adresse / ed i t ”* : [ ] , *// Suppression d ’ une adresse*
2. *” adresse / s a i s i e ”* : [ ] , *// c l i c k sur l a modification de l a personne s é l ect i on n ée*

39

1. *// N o t i f i cat i on s de modification du modèle pour requ ê t e AJAX et /ou mise à jour de l a vue*
2. *” personne/ changed ”* : [ ] , *// mise à jour d ’ une personne*
3. *” personne/ created ”* : [ ] , *// mise à jour d ’ une personne*
4. *” personne/ details Changed ”* : [ ] , *// Mise à jour requise du panneau des dé t a i l s*

44

1. *// N o t i f i cat i on s de modification du modèle pour requ ê t e AJAX et /ou mise à jour de l a vue*
2. *” adresse / changed”* : [ ] , *// mise à jour d ’ une adresse*
3. *” adresse / created ”* : [ ] , *// mise à jour d ’ une adresse*

48

49

1. *// Demande de r é enregistrement d ’ événements u t i l i s a t e u r s s u i t e à reconstruction d ’ é l éments HTML*

−

1. *” personne/ h tml Liste I t em Rebuilt ”* : [ ] , *// Ré enregistrement des événements de c l i c k sur l e s i tems*
2. *// s u i t e à reconstruction complète du*

code HTML des i tems .

1. *” personne/ d e t a i l s R e b u i l t ”* : [ ] , *// Ré enregistrement des événements de c l i c k sur l e s boutons ” Supprimer ” , ” Modifier ”*
2. *// s u i t e à reconstruction du code HTML*

des dé t a i l s .

|  |  |
| --- | --- |
| 55 |  |
| 56 | } ; |
| 57 | } ; |
| 58 |  |
| 59 | *// Appel de l a mé thode d ’ i n i t i a l i s a t i o n* |
| 60  61 | i n i t ( ) ; |
| 62  63 | */\*\**  *\* In t er f ace publique du module mediator* |
| 64 | *\*/* |
| 65 | **var** p u b l i c I n te r f a c e M e d i a to r = { |
| 66 | */\*\** |
| 67 | *\* Enregistrement d ’ un cal l b ack sur un événement .* |
| 68 | *\* I l peut y avoir p l u s i eu r s cal l b ack s sur un même événement* |
| 69 | *\* ( par exemple : mise à jour de deux p ar t i es d i s t i n c t e s de l a vue )* |
| 70 | *\* @param { s t r i n g } event Categ événement , qui d o i t ê t re un nom de propri é t é* |
|  | *de m\_ subscription Lists* |
| 71 | *\* @param { function } callback Function l a fonction qui sera appel ée en r é* |
|  | *action à l ’ événement .* |
| 72 | *\*/* |
| 73 | s u b s c r i b e : **function** ( eventCateg , c a l l b a c k Fu n c t i o n ) { |
| 74 | **i f** ( m\_ subscription Lists . hasOwnProperty ( eventCateg ) ) { |
| 75 | m\_ subscription Lists [ eventCateg ] . push ({ c a l l b a c k : c a l l b a c k Fu n c t i o n }) ; |
| 76 | } **else** { |
| 77 | **throw new** Error ( *” Caté gorie d ’ événements ”* + eventCateg + *” inconnue du m* |
|  | *é d i ateur ”* ) ; |
| 78 | } |
| 79  80 | } , |
| 81  82 | */\*\**  *\* Publication d ’ un événement survenu et ex é cution de tous l e s cal l b ack s correspondants .* |
| 83 | *\* @param { s t r i n g } event Categ événement , qui d o i t ê t re un nom de propri é t é* |
|  | *de m\_ subscription Lists* |
| 84 | *\* @param { Object } context Arg argument optionnel à t ransmettre au cal l b ack (* |
|  | *exemple : item c l i q u é . . . )* |
| 85 | *\*/* |
| 86 | p u b l i s h : **function** ( eventCateg , contextArg ) { |
| 87 | **var** i ; |
| 88 | **i f** ( m\_ subscription Lists . hasOwnProperty ( eventCateg ) ) { |
| 89 | **for** ( **var** i =0 ; i**<** m\_ subscription Lists [ eventCateg ] . l e n g th ; ++i ) { |
| 90 | *// On ap p e l l e l e cal l b ak avec son* |
| 91 | m\_ subscription Lists [ eventCateg ] [ i ] . c a l l b a c k ( contextArg ) ; |
| 92 | } |
| 93 | } **else** { |
| 94 | **throw new** Error ( *” Caté gorie d ’ événements ”* + eventCateg + *” inconnue du m* |
|  | *é d i ateur ”* ) ; |
| 95 | } |
| 96 | } , |
| 97 |  |
| 98  99  100  101 | *// Ré i n i t i a l i s e l e s l i s t e s de cal l b ack s à v ide .*  empty : **function** ( ) { i n i t () ;  } |
| 102 | } ; |
| 103 |  |

104

**return** p u b l i c I n te r f a c e M e d i a to r ;

}() ] ) ;

105

## Événements concernant les personnes

### Enregistrement des événements utilisateurs via *jQuery*

Tous les événements recensés dans le **diagramme de cas d’utilisation** (voir la figure [5.2](#_bookmark94)) se verront ici attribué un gestionnaire qui, généralement, ne fera que publier l’événement auprès de *mediator* (partie [5.3](#_bookmark95)). Les éléments *HTML* constants de la vue (<span>, <button>, <div>, <p>, etc.) sur lesquels ces événements seront appliqués sont définis dans le fichier *HTML* principal décrit dans la partie [5.4.9](#_bookmark105).

Ces événements utilisateurs doivent parfois être réenregistrés suite à la reconstruction des éléments *HTML* concernés. Les événements sont alors détruits (méthodes jQuery.off(), ou jQuery.empty(), ou encore jQuery.remove()), puis, le code *HTML* est regénéré, et enfin, les événements utilisateur sont ré-enregistrés (méthode jQuery.on()).

S’il faut prévoir de ré-enregistrer un gestionnaire d’événement utilisateur, nous allons per- mettre de déclencher ce ré-enregistrement via le *mediator*. Ceci permet d’éviter notamment des problèmes de dépendance cyclique des fonctions *JavaScript* ou modules, par exemples du fait que les événements *jQuery* doivent être initialisés après la génération de la vue.

Code Source 5.3 : /ihm-demo/guiJQueryEventsPersonne.js

|  |  |
| --- | --- |
| 1 | */\*\** |
| 2 | *\* Mé thode d ’ i n i t i a l i s a t i o n des événements u t i l i s a t e u r s Java Script .* |
| 3 | *\* Enregistrement des g es t i on n a i r es de ces événements v ia jQuery .* |
| 4 | *\*/* |
| 5 | myApp. addModule . apply (myApp. gui , [ *” init JQuery Events Personne”* , **function** ( ) { |
| 6 |  |
| 7 | */////////////////////////////////////////////////////////* |
| 8 | *// c l i c k sur l e bouton ” Ajouter une personne” f a i s an t s o r t i r l e formulaire* |
| 9 |  |
| 10 | */\*\** |
| 11 | *\* Gestionnaire c l i c k sur l e bouton f a i s an t s o r t i r l e formulaire* |
| 12 | *\*/* |
| 13 | **var** c l i c k Bo u to n Sa i s i e Pe r s o n n e = **function** ( event ) { |
| 14 | *// p u b l i cat i on auprès du mé d i at or* |
| 15 | myApp. gui . mediator . p u b l i s h ( *” personne/ s a i s i e ”* , { |
| 16 | personne : myApp. modele . s e l e c te d Pe r s o n n e |
| 17 | }) ; |
| 18 | } ; |
| 19 |  |
| 20 | *// Enregistrement du Handler du c l i c k pour modifier l e s dé t a i l s de l ’ item s é l ect i on n é v ia jQuery* |
| 21  22 | $ ( *”#bouton Ajouter Personne”* ) . on ( *” c l i c k ”* , c l i c k Bo u to n Sa i s i e Pe r s o n n e ) ; |
| 23 | */////////////////////////////////////////////////////////* |
| 24 | *// c l i c k sur l e bouton ” Modifier l e nom” f a i s an t s o r t i r l e formulaire* |
| 25 |  |
| 26 | */\*\** |
| 27 | *\* Gestionnaire c l i c k sur l e bouton f a i s an t s o r t i r l e formulaire* |
| 28 | *\*/* |

29 **var** c l i ck Bouton Modifier Personne = **function** ( event ) { 30

1. *// p u b l i cat i on auprès du mé d i at or*
2. myApp. gui . mediator . p u b l i s h ( *” personne/ ed i t ”* , {
3. personne : myApp. modele . s e l e c te d Pe r s o n n e 34 }) ;

35 } ;

36

37 /////////////////////////////////////////////////////////

38 // c l i c k sur l e bouton ” Supprimer l a personne” f a i s an t s o r t i r l e formulaire

39

40 */\*\**

41 \* Gestionnaire c l i c k sur l e bouton f a i s an t s o r t i r l e formulaire

42 *\*/*

1. **var** click Bouton Supprimer Personne = **function** ( event ) {
2. *// p u b l i cat i on auprès du mé d i at or*
3. myApp. gui . mediator . p u b l i s h ( *” personne/ d e l e t e ”* , {
4. personne : myApp. modele . s e l e c te d Pe r s o n n e 47 }) ;

48 } ;

49

50

51 ////////////////////////////////////////////////////////

52 // Gestionnaire de submit formulaire de modification de personne .

53

54 */\*\**

1. *\* Gestionnaire de l ’ événement submit du formulaire .*
2. *\* @param { Event} jQuery event correspondant au handler .*

57 *\*/*

58 **var** form Handler Modif Personne = **function** ( event ) { 59

1. *// Év i t e r d ’ appeler l ’” action ” par dé f aut () s c r i p t PHP, etc . . . )*
2. *// du formulaire l or s du submit*
3. event . prevent Default ( ) ; 63
4. *// p u b l i cat i on auprès du mé d i ator*
5. myApp. gui . mediator . p u b l i s h ( *” personne/ update”* , {
6. personne : myApp. modele . s e l e c te d Pe r s o n n e 67 }) ;

68 } // f i n du g es t i on n a i re form HandlerModif ()

69

1. *// Enregistrement du Handler du submit du formulaire v ia jQuery*
2. $ ( *”#modifierPersonneForm”* ) . on ( *” submit ”* , form Handler Modif Personne ) ; 72

73

74 ///////////////////////////////////////////////////////

75 // Gestionnaire de submit formulaire d ’ ajout de personne .

76

77 */\*\**

1. *\* Gestionnaire de l ’ événement submit du formulaire .*
2. *\* @param { Event} jQuery event correspondant au handler .*

80 *\*/*

81 **var** form Handler AjoutPersonne = **function** ( event ) { 82

1. *// Év i t e r d ’ appeler l ’” action ” par dé f aut () s c r i p t PHP, etc . . . )*
2. *// du formulaire l or s du submit*

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

event . prevent Default ( ) ;

// p u b l i cat i on auprès du mé d i ator

myApp. gui . mediator . p u b l i s h ( *” personne/ create ”* , {

personne : myApp. modele . s e l e c te d Pe r s o n n e

}) ;

} // f i n du g es t i on n a i re form Handler Ajout ()

// Enregistrement du Handler du submit du formulaire v ia jQuery

$ ( *”#ajouterPersonneForm”* ) . on ( *” submit ”* , form Handler AjoutPersonne ) ;

/\*\*

* *Enregistre l e s événements de c l i c k s sur l e s boutons ” Modifier ” et ” Supprimer”*
* *l a personne s é l ect i on n ée .*
* *Cette fonction d o i t ê t re invoqu ée en cas de s é l ec t i on d ’ une nou v e l l e personne*
* *( reconstruction deu code HTML du panneau des dé t a i l s .*

\*/

**var** r e g i s te r Bu tto n C l i c k Ev e n ts = **function** () {

// Enregistrement du Handler du c l i c k pour modifier l e s dé t a i l s de l ’ item s é l ect i on n é v ia jQuery

$ ( *”#bouton Modifier Personne”* ) . on ( *” c l i c k ”* , c l i ck Bouton Modifier Personne ) ;

// Enregistrement du Handler du c l i c k pour supprimer l ’ item s é l ect i on n é v ia jQuery

$ ( *”#bouton Supprimer Personne”* ) . on ( *” c l i c k ”* , click Bouton Supprimer Personne ) ;

}

/////////////////////////////////////////////////////////

// Clicks sur l e s é l éments de l a l i s t e d ’ i tems

/\*\* Mé thode qui permet de cr é er un g es t i on n aire d ’ événement de c l i c k

* *sur chaque nom de personnes ( s é l ec t i on des dé t a i l s )*
* *Ces g es t i on n a i r es p u b l i en t l ’ événnement ” nou v e l l e personne s é l ect i on n ée” auprès du mé d i ator .*
* *@param { i n t } index indice de l ’ item pour l eq u e l on en r eg i s t r e l ’ événement .*

\*/

**var** r e g i s t e r H e l p e r S e l e c t D e t a i l s = **function** ( index ) {

**return function** ( ) {

myApp. gui . mediator . p u b l i s h ( *” personne/ s e l e c t D e t a i l s ”* ,

{

personne : myApp. modele . personnes [ index ]

}) ;

} ;

} ;

/\*\*

* + *Enregistre l e s événements j av as c r i p t de c l i c k sur l e s é l éments de l a l i s t e*
  + *( noms des personnes ) .*
  + *Cette mé thode d o i t ê t re appel ée l or s de l a reg éné ration du code de l a l i s t e*

.

* + *@method r eg i s t er List e Person n es Cl i ck s*
  + *@param { Object } context Args non u t i l i s é*
  + *@return { function } une fonction cal l b ack qui gères l e c l i c k sur l ’ item index*

134

**var** r e g i s t e r L i s t e P e r s o n n e s C l i c k s = **function** ( context Args ) {

**for** ( **var** i =0 ; i**<**myApp . modele . personnes . l e n g th ; ++i ) {

$ ( *”#master\_”*+ myApp. modele . personnes [ i ] . g e t Id () ) . on (

*” c l i c k ”* , r e g i s t e r H e l p e r S e l e c t D e t a i l s ( i ) ) ;

}

} ;

*// Enregistrer l e s c l i c k s l or s de l ’ i n i t i a l i s a t i o n*

r e g i s te r Bu tto n C l i c k Ev e n ts ( ) ;

r e g i s t e r L i s t e P e r s o n n e s C l i c k s ( ) ;

*// Permet à l a mé thode qui reg é nère t oute l a l i s t e des i tems*

*// de recr éer , v ia l e mé diator , l e s événements ” c l i c k ” sur l e s i tems .*

myApp. gui . mediator . s u b s c r i b e ( *” personne/ h tml Liste I t em Rebuilt ”* , r e g i s t e r L i s t e P e r s o n n e s C l i c k s ) ;

*// Permet à l a mé thode qui reg é nère l e panneau des dé t a i l s de recr éer ,*

*// v ia l e mé diator , l e s événements ” c l i c k ” sur l e s boutons dans l e panneau des dé t a i l s .*

myApp. gui . mediator . s u b s c r i b e ( *” personne/ d e t a i l s R e b u i l t ”* , r e g i s te r Bu tto n C l i c k Ev e n ts ) ;

} ] ) ;

*\*/*

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

### Mise à jour du panneau des détails

Le panneau des détails de l’*item* sélectionné doit être mis à jour lors de la modification de la personne par validation du formulaire, ou lors du changement de l’*item* sélectionné (*click* sur un autre *item*). dans ce cas, les événements utilisateurs sur les éléments *HTML* qui sont générés dynamiquement sur le panneau des détails doivent aussi être reconstruit (événement personne/detailsRebuilt du *mediator*).

Code Source 5.4 : /ihm-demo/guiDetailsChanged.js

|  |  |
| --- | --- |
| 1 | */\*\** |
| 2 | *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de mise à jour des dé t a i l s de l ’* |
|  | *item s é l ect i on n ée .* |
| 3 | *\*/* |
| 4 | myApp. addModule . apply (myApp. gui , [ *” cal l b acks U pdat e D et ai l s ”* , **function** () { |
| 5 |  |
| 6 | */\*\** |
| 7 | *\* Géné ration du code HTML des dé t a i l s de l a personne s é l ect i on n ée .* |
| 8 | *\*/* |
| 9  10 | **var** getHtml Code Detail = **function** () {  **var** htmlCode = *”***<span** *c l as s =\” panel \”***>***”* + |
| 11 | *”***<p><s trong>***Nom :***</s trong>** *”* + myApp. modele . s e l e c te d Pe r s o n n e . getNom ( )  + *”***</p>***”* + |
| 12 | *”***<button** *id=\” bouton Modifier Personne \”***>***M odifier***</button><br** */***>***”* + |
| 13 | *”***<button** *id=\” bouton Supprimer Personne\”***>***Supprimer***</button><br** */***>***”* + |
| 14 | *”***<button** *id=\” bouton Ajouter Adresse \”***>***A jouter une adresse***</button>***”* ; |
| 15 | **for** ( **var** index = 0 ; index **<** myApp. modele . s e l e c te d Pe r s o n n e . getNb Adresses () ; |
|  | ++i ndex ) { |
| 16 | htmlCode += *”***<p>***”* + |
| 17 | myApp. view . a d r e s s e . getHtml Developped (myApp. modele . |
|  | s e l e c te d Pe r s o n n e . get Adresse ( index ) ) |

18

19

20

21

22

23

24

25 }

+ *”***<br** */***><button** *id=\” bouton SupprimerAdresse\_”*

+ myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( index ) . g e t A ttr i b u te ( *’ id ’* )

+*”\”***>***Supprimer l ’ adresse***</button>***”*

+ *”***<br** */***><button** *id=\” bouton Modifier Adresse\_”*

+ myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( index ) . g e t A ttr i b u te ( *’ id ’* )

+*”\”***>***M odifier l ’ adresse***</button>***”*

+ *”***</p>***”* ;

26

27

28 } ;

29

htmlCode += *”***</span>***”* ;

**return** htmlCode ;

30 */\*\**

1. *\* Redessine l e s dé t a i l s d ’ une personne s u i t e à sa s é l ec t i on ou sa modification .*
2. *\* @param { Object } context Arg non u t i l i s é .*

33 *\*/*

34 **var** r e p a i n t D e ta i l = **function** ( contextArg ) { 35

1. $ ( *”#modifierPersonneForm”* ) . empty () ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
2. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
3. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
4. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*

40

41 $ ( *”#vue Detail ”* ) . empty ( ) ; *// Vider l e s dé t a i l s de l ’ item s é l ect i on n é*

42

43 $ ( *”#vue Detail ”* ) . html ( getHtml Code Detail ( ) ) ; *// Géné ration du code HTML*

44

1. *// Recré er l e s événements de c l i c k s sur l e s boutons ” modifier ” , ” supprimer ” , etc .*
2. myApp. gui . mediator . p u b l i s h ( *” personne/ d e t a i l s R e b u i l t ”* ) ; 47 } ;

48

1. *// Enregistrement du cal l b ack de l ’ événement dé di é (m. a . j . des dé t a i l s )*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ details Changed ”* , r e p a i n t D e ta i l ) ;
3. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
4. myApp. gui . mediator . s u b s c r i b e ( *” personne/ changed”* , r e p a i n t D e ta i l ) ; 53

54 }() ] ) ;

### Mise à jour du panneau des *items*

Le panneau qui affiche la liste des l’*items* doit être mis à jour lors de la modification de la personne par validation du formulaire (le nom de la personne peut changer), ou lors du changement de l’*item* sélectionné, celui-ci étant surligné.

En cas de changement de l’*item* sélectionné, la propriété selectedPersonne du modèle sera

modifiée, et le rafraîchissement du panneau des détails sera ensuite provoqué.

Lors de la création d’une nouvelle personne, celle-ci sera automatiquement sélectionnée.

1 */\*\**

##### Code Source 5.5 : /ihm-demo/guiPersonneChanged.js

* 1. *\* Dé f i n i t i o n et abonnement des cal l b ac k s de mise à jour de l a*
  2. *\* l i s t e c l i c k a b l e des items , s o i t l or s de l a modification*
  3. *\* du modèle , s o i t l or s du changement de personne s é l ect i on n ée .*

5 *\*/*

6 myApp. addModule . apply (myApp. gui , [ *” callbacks Main List Update ”* , **function** () { 7

8 */\*\**

9 \* Active ou dé s act i v e l e su r l i g n ag e ( s t y l e CSS) d ’ un item de l a l i s t e .

1. *\* @param { personne} personne item de l a l i s t e à modifier ( v ia l ’ ID de l ’ é l é ment HTML) .*
2. *\* @param { boolean } h i l g h l i g h t e d t rue s i on d o i t surligner , f a l s e pour remettre l e s t y l e par dé f aut .*

12 *\*/*

1. **var** s e t H i g h l i g h te d = **function** ( personne , h i g h l i g h te d ) {
2. **i f** ( h i g h l i g h te d ) {
3. *// Mettre l e s t y l e s u r l i g n é sur l ’ item de l a l i s t e*
4. $ ( *”#master\_”*+ personne . g e t Id ( ) ) . c s s ( *”***background color** *”* , *”#333”* )

−

1. . c s s ( *”* **color** *”* , *”#eee”* )
2. . c s s ( *”* **border** *radius ”* , *” 4 px”* )

−

1. . c s s ( *”***padding***”* , *” 2 px”* ) ;
2. } **else** {
3. *// Remettre l e s t y l e normal sur l ’ item de l a l i s t e*
4. $ ( *”#master\_”*+ personne . g e t Id ( ) ) . c s s ( *”***background color** *”* , *”#eee”* ) 23 . c s s ( *”* **color** *”* , *”#333”* )

−

1. . c s s ( *”* **border** *radius ”* , *” 4 px”* )

−

1. . c s s ( *”***padding***”* , *” 2 px”* ) ;

26 }

27 }

28

29 */\*\**

30 \* Géné ration du code HTML de l a l i s t e de personnes

31 *\*/*

1. **var** get Html Code Liste Personnes = **function** () {
2. **var** htmlCode = *””* ;
3. **for** ( **var** i =0 ; i**<**myApp . modele . personnes . l e n g th ; ++i ) {
4. htmlCode +=
5. *”***<p** *id=\”master\_”*+ myApp. modele . personnes [ i ] . g e t Id () + *”\”***>***”* +
6. *”***<s trong>***Nom :***</s trong>** *”* + myApp. modele . personnes [ i ] . getNom ( ) + *”***</p>***”* ; 38 }

39 **return** htmlCode ; 40 } ;

41

42 */\*\**

1. *\* Raffraichissement ( ou a f f i ch ag e ) de t oute l a vue .*
2. *\* @param context Arg non u t i l i s é .*

45 *\*/*

46 **var** repaint Vue = **function** ( contextArg ) { 47

1. $ ( *”#l i s t e Personnes ”* ) . empty () ; *// Vider l a l i s t e et ses événements*
2. $ ( *”#l i s t e Personnes ”* ) . html ( get Html Code Liste Personnes () ) ; *// a f f i c h e r*

50

1. *// Appliquer l e s t y l e par dé f aut sur tous l e s i tems*
2. **for** ( **var** i =0 ; i **<** myApp. modele . personnes . l e n g th ; ++i ) {
3. s e t H i g h l i g h te d (myApp. modele . personnes [ i ] , **f al s e** ) ;

54

55

56

57

58

59

60 } ;

61

62 */\*\**

}

// Surligner l ’ item s é l ect i on n é

s e t H i g h l i g h te d (myApp. modele . se l e cted Per so nn e , **true** ) ;

// Recré er l e s événements de c l i c k s sur l e s i tems de l a l i s t e

myApp. gui . mediator . p u b l i s h ( *” personne/ h tml Liste I t em Rebuilt ”* ) ;

1. *\* Changer l ’ item s é l ect i on n é en r é action à un c l i c k .*
2. *\* @param { Object } context Arg argument indiquant l a nou v e l l e personne s é l ect i on n ée .*
3. *\* @param { personne} context Arg . personne nou v e l l e personne s é l ect i on n ée .*

66 *\*/*

1. **var** s e l e c t Pe r s o n n e = **function** ( contextArg ) {
2. *// Supprimer l e su r l i g n ag e de l ’ ancienne personne s é l ect i on n ée*
3. s e t H i g h l i g h te d (myApp. modele . s e l ec te d Pe r s on ne , **f al s e** ) ; 70
4. *// Changer l ’ item s é l ect i on n é*
5. myApp. modele . s e l e c te d Pe r s o n n e = contextArg . personne ; 73
6. *// Mettre l e s t y l e s u r l i g n é sur l ’ item s é l ect i on n é de l a l i s t e*
7. s e t H i g h l i g h te d (myApp. modele . s e l ec te d Pe r s on ne , **true** ) ; 76
8. *// Provoquer l a mise à jour du panneau des dé t a i l s*
9. myApp. gui . mediator . p u b l i s h ( *” personne/ details Changed ”* , {
10. personne : myApp. modele . s e l e c te d Pe r s o n n e 80 }) ;

81 } ;

82

83 */\*\**

1. *\* Changer l ’ item s é l ect i on n é s u i t e à cr é ation d ’ une personne et mise à jour de l a vue .*
2. *\* @param { Object } context Arg argument indiquant l a nou v e l l e personne s é l ect i on n ée .*
3. *\* @param { personne} context Arg . personne nou v e l l e personne s é l ect i on n ée .*

87 *\*/*

1. **var** select Personne An Repaint = **function** ( contextArg ) {
2. s e l e c t Pe r s o n n e ( contextArg ) ;
3. repaint Vue () ; 91 }

92

1. *// Enregistrement du cal l b ack de modification de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ changed”* , repaint Vue ) ;
3. *// Enregistrement du cal l b ack de cr é ation de l a personne*
4. myApp. gui . mediator . s u b s c r i b e ( *” personne/ created ”* , select Personne An Repaint ) ;
5. *// Enregistrement du cal l b ack de s é l ec t i on d ’ une nou v e l l e personne .*
6. myApp. gui . mediator . s u b s c r i b e ( *” personne/ s e l e c t D e t a i l s ”* , s e l e c t Pe r s o n n e ) ; 99 }() ] ) ;
   * 1. **Bouton ”*Supprimer*”**

##### Lorsque l’utilisateur clique sur ”*Supprimer*”, la personne sélectionnée est supprimée du modèle. Une nouvelle personne est sélectionnée (personne par défaut) et la vue est réinitialisée.

Code Source 5.6 : /ihm-demo/guiBoutonSupprimerPersonne.js

|  |  |  |
| --- | --- | --- |
| 1  2 | */\*\**  *bouton* |  |
| 3 | *\* ” modifier ” l a personne s é l ect i on n ée .* |
| 4 | *\*/* |
| 5 | myApp. addModule . apply (myApp. gui , [ *” callbacks Click Supprimer ”* , **function** () { |
| 6 |  |
| 7 | */\*\** |
| 8 | *\* Callback qui supprime l a personne pass ée dans l ’ ob j e t pass é en argument .* |
| 9 | *\* @param { Object } context Arg argument indiquant l a personne à supprimer .* |
| 10 | *\* @param { personne} context Arg . personne r é f é rence de l ’ instance de personne* | *à* |
|  | *supprimer dans l e modèle .* |  |
| 11 | *\*/* |  |
| 12 | **var** d e l e te Pe r s o n n e = **function** ( contextArg ) { |  |
| 13 | *// Indice dans l e t ab l eau de l a personne à supprimer .* |  |
| 14 | **var** index Selected Personne = myApp. modele . personnes . index Of ( contextArg . |  |
|  | personne ) ; |  |
| 15 | *// Suppression de l a personne dans l e modèle* |  |
| 16 | myApp. modele . personnes . s p l i c e ( index Selected Personne , 1 ) ; |  |
| 17 | *// Personne s é l ect i on n ée par dé f aut* |  |
| 18 | myApp. modele . s e l e c te d Pe r s o n n e = myApp. modele . personnes [ 0 ] ; |  |
| 19 |  |  |
| 20 | *// Provoquer l a mise à jour de l a vue :* |  |
| 21  22  23 | myApp. gui . mediator . p u b l i s h ( *” personne/ changed”* , {  }) ; |  |
| 24 | } |  |
| 25 |  |  |
| 26 | *// Enregistrement du cal l b ack* |  |
| 27 | myApp. gui . mediator . s u b s c r i b e ( *” personne/ d e l e t e ”* , d e l e te Pe r s o n n e ) ; |  |
| 28 |  |  |
| 29 | }() ] ) ; |  |

\* Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e

personne : myApp. modele . s e l e c te d Pe r s o n n e

### Bouton ”*Modifier*” et affichage du formulaire

Lorsque l’utilisateur clique sur ”*Modifier*”, le formulaire doit être affiché avec les données de la personnes dans les *inputs*.

Code Source 5.7 : /ihm-demo/guiBoutonModifierPersonne.js

1

*/\*\**

*\*/*

myApp. addModule . apply (myApp. gui , [ *” cal l backs Click Modifier Personne ”* , **function** () {

*/\*\**

*\* Géné ration du code HTML du formulaire de modification de l a personne s é l ect i on n ée .*

*\*/*

**var** getHtmlFormInputs = **function** () {

* *Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e bouton*
* *” modifier ” l a personne s é l ect i on n ée .*

**return** *”***<span** *s t y l e =\”***width** *:360 px ;* **display** *: in l ine −block ;* **vertical** *−***align** *:*

**top** *;\”***>***”* +

2

3

4

5

6

7

8

9

10

11

12

13

14

15 }

16

17 */\*\**

myApp. gui . getHtmlFormInputs (myApp. modele . s e l ected Pe r s on ne , *” modifierPersonneForm”* ) +

*”***<l abel></ label><i nput** *type=\” submit \” value=\” Valider \”***></ i nput>***”* +

*”***</span>***”* ;

1. *\* Callback d ’ Affichage ( v ia l e DOM) du formulaire dans l ’ é l ément d ’ ID ” modifierPersonneForm”*
2. *\* @param { Object } context Arg non u t i l i s é .*

20 *\*/*

1. **var** repaint Form Inputs = **function** ( contextArg ) {
2. $ ( *”#modifierPersonneForm”* ) . empty () ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
3. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
4. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
5. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*

26

27 $ ( *”#modifierPersonneForm”* ) . append ( getHtmlFormInputs () ) ; *// ajouter l e s nouveaux inputs*

28 } ;

29

1. *// Enregistrement du cal l b ack*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ ed i t ”* , repaint Form Inputs ) ; 32

33 }() ] ) ;

* + 1. **Bouton ”*Ajouter une personne*”**

##### Lorsque l’utilisateur clique sur ”*Ajouter une personne*”, le formulaire doit être affiché avec les valeurs par défaut (typiquement des champs vides) dans les *inputs*.

Pour cela, on utilise la possibilité offerte par la fabrique de nos modules métier (partie [2.6.3](#_bookmark58)) de créer une objet par défaut en passant null en argument de la fabrique. Ceci permet de ne pas générer de messages d’erreur en cas de champs obligatoire initialement vide.

Après validation du formulaire, la personne est ajoutée dans le modèle, elle est automati- quement sélectionnée, et la vue est mise à jour.

Code Source 5.8 : /ihm-demo/guiBoutonAjouterPersonne.js

|  |  |
| --- | --- |
| 1 | */\*\** |
| 2 | *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e* |
|  | *bouton* |
| 3  4 | *\* ” modifier ” l a personne s é l ect i on n ée .*  *\*/* |
| 5 | myApp. addModule . apply (myApp. gui , [ *” cal l b ack s Cl i ck A j ou t er ”* , **function** () { |
| 6 |  |
| 7 | */\*\** |
| 8 | *\* Géné ration du code HTML du formulaire de modification de l a personne s é* |
|  | *l ect i on n ée .* |
| 9 | *\*/* |
| 10 | **var** getHtmlFormInputs = **function** () { |

11

12

13

14

15

16 }

17

**return** *”***<span** *s t y l e =\”***width** *:360 px ;* **display** *: in l ine b l ock ;* **vertical align** *:*

**top** *;\”***>***”* +

− −

*”***<s trong** *s t y l e =\”***width** *: 360 px ;* **display** *: in l ine b l ock ;* **text align** *:*

− −

*center ;* **padding** *: 15 px ;\”***>***S a i s i e d ’ une nou v e l l e personne***</strong>**

*”* +

myApp. gui . getHtmlFormInputs (myApp. metier . personne . c r e a te I n s ta n c e (

**null** ) , *” ajouter PersonneForm”* ) +

*”***<l abel></ label><i nput** *type=\” submit \” value=\” Valider \”***></ i nput>***”* +

*”***</span>***”* ;

18 */\*\**

1. *\* Callback d ’ Affichage ( v ia l e DOM) du formulaire dans l ’ é l ément d ’ ID ” mainForm”*
2. *\* @param { Object } context Arg non u t i l i s é .*

21 *\*/*

1. **var** repaint Form Inputs = **function** ( contextArg ) {
2. $ ( *”#modifierPersonneForm”* ) . empty () ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
3. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
4. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
5. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*

27

28 $ ( *”#ajouter PersonneForm”* ) . append ( getHtmlFormInputs ( ) ) ; *// ajouter l e s nouveaux inputs*

29 } ;

30

1. *// Enregistrement du cal l b ack*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ s a i s i e ”* , repaint Form Inputs ) ; 33

34 }() ] ) ;

### Validation du formulaire de modification

Lors de la validation (événement *submit*) du formulaire de modification, les données de la personne sélectionnée doivent être mises à jour à partir des valeurs saisies dans le formulaire. Les panneaux potentiellement impactés (liste des *items*, panneau des détails) sont alors mis à jour.

Code Source 5.9 : /ihm-demo/guiModifierPersonneFormValidate.js

1

*/\*\**

* *Dé f i n i t i o n et enregistrement du cal l b ack r é ag i s s san t à l a v a l i d a t i on ( submit )*
* *du formulaire de modification d ’ une personne .*

*\*/*

myApp. addModule . apply (myApp. gui , [ *” callbacks Validate Modifier Form ”* , **function** () {

*// Formulaire de modification d ’ une personne*

*/\*\**

*\* Modifie l e modèle à p ar t i r des donné es s a i s i e s dans l e formulaire*

*\*/*

**var** updateModel = **function** () {

2

3

4

5

6

7

8

9

10

11

|  |  |
| --- | --- |
| 12 |  |
| 13 | *// 1 ) Mise à jour des donné es du modèle* |
| 14 | *// à p ar t i r des valeurs des inputs du formulaire* |
| 15 | **var** attribute Name , |
| 16 | input Id ; |
| 17  18  19 | *// On r é cupère l e s a t t r i b u t s du formulaire dans une nou v e l l e instance*  **var** changed Personne = myApp. metier . personne . c r e a te I n s ta n c e ( **null** ) ;  *// Pour chaque propri é t é ( chaque input du formulaire )* |
| 20 | **for** ( **var** j=0 ; j**<** myApp. metier . personne . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++j ) { |
| 21 | **var** attribute Name = myApp. metier . personne . g e t A ttr i b u te L i s t () [ j ] ; |
| 22 | **i f** ( attribute Name != *” id ”* ) { |
| 23 | *// cal cu l de l ’ ID de l ’ input* |
| 24 | **var** input Id = myApp. gui . get Input Id ({ |
| 25 | attribute Name : attribute Name , |
| 26 | form Id : *” modifierPersonneForm”* |
| 27 | }) ; |
| 28 | *// Modification de l a propri é t é de l a personne* |
| 29 | *// avec l a valeur s a i s i e dans l ’ input .* |
| 30 | changed Personne . s e t A ttr i b u te ( attribute Name , |
| 31 | document . getElementBy Id ( input Id ) . value |
| 32 | ) ; |
| 33 |  |
| 34 | } |
| 35 | } |
| 36 | *// Seulement s ’ i l n ’ y a pas d ’ erreur ( f i l t r a g e s t r i c t cô t é c l i e n t )* |
| 37 | **i f** ( ! changed Personne . has Error ( ) ) { |
| 38 | **for** ( **var** j=0 ; j**<** myApp. metier . personne . g e t A ttr i b u te L i s t () . l e n g th ; ++j ) { |
| 39  40  41  42 | **var** attribute Name = myApp. metier . personne . g e t A t tr i b u te L i s t ( ) [ j ] ;  **i f** ( attribute Name != *” id ”* ) {  myApp. modele . s e l e c te d Pe r s o n n e . s e t A ttr i b u te ( attribute Name ,  changed Personne . g e t A ttr i b u te ( attribute Name ) ) ; |
| 43 | } |
| 44 | } |
| 45 | *// Provoquer l a mise à jour des é l éments de l a vue observant l a personne* |
| 46 | myApp. gui . mediator . p u b l i s h ( *” personne/ changed ”* , { |
| 47 | personne : myApp. modele . |
|  | s e l e c te d Pe r s o n n e |
| 48 | }) ; |
| 49 | } |
| 50 | } ; |
| 51 |  |
| 52 | *// Enregistrement du cal l b ack de l ’ événement de v a l i d a t i on du formulaire* |
| 53 | myApp. gui . mediator . s u b s c r i b e ( *” personne/ update”* , updateModel ) ; |
| 54 | }() ] ) ; |

### Validation du formulaire d’ajout d’une personne

Lors de la validation (événement *submit*) du formulaire d’ajout, une personne doit être ajoutée au modèle à partir des valeurs saisies dans le formulaire. Les panneaux potentiellement impactés (liste des *items*, panneau des détails) sont alors mis à jour.

Code Source 5.10 : /ihm-demo/guiAjouterPersonneFormValidate.js

1

*/\*\**

|  |  |
| --- | --- |
| 2 | *\* Dé f i n i t i o n et enregistrement du cal l b ack r é ag i s s san t à l a v a l i d a t i on ( submit )* |
| 3 | *\* du formulaire de modification d ’ une personne .* |
| 4 | *\*/* |
| 5 | myApp. addModule . apply (myApp. gui , [ *” callbacks Validate Ajouter Form ”* , **function** ( ) { |
| 6 |  |
| 7  8  9 | */\*\**  *\* Modifie l e modèle à p ar t i r des donné es s a i s i e s dans l e formulaire*  *\*/* |
| 10 | **var** updateModel = **function** () { |
| 11 | *// 1 ) Mise à jour des donné es du modèle* |
| 12 | *// à p ar t i r des valeurs des inputs du formulaire* |
| 13 | **var** attribute Name , |
| 14 | input Id ; |
| 15 |  |
| 16 | *// Récupé ration des donné es du formulaire dans une nou v e l l e instance* |
| 17 | **var** nouvelle Personne = myApp. metier . personne . c r e a te I n s ta n c e ( **null** ) ; |
| 18 | *// Pour chaque propri é t é ( chaque input du formulaire )* |
| 19 | **for** ( **var** j=0 ; j**<** myApp. metier . personne . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++j ) { |
| 20 | **var** attribute Name = myApp. metier . personne . g e t A ttr i b u te L i s t () [ j ] ; |
| 21 | **i f** ( attribute Name != *” id ”* ) { |
| 22 | *// cal cu l de l ’ ID de l ’ input* |
| 23 | **var** input Id = myApp. gui . get Input Id ({ |
| 24 | attribute Name : attribute Name , |
| 25 | form Id : *” ajouter PersonneForm”* |
| 26 | }) ; |
| 27 | *// Modification de l a propri é t é de l a personne* |
| 28 | *// avec l a valeur s a i s i e dans l ’ input .* |
| 29  30  31  32 | nouvelle Personne . s e t A ttr i b u te ( attribute Name ,  document . getElementBy Id ( input Id ) . value  ) ; |
| 33 | } |
| 34 | } |
| 35 | *// Seulement s ’ i l n ’ y a pas d ’ erreur ( f i l t r a g e s t r i c t cô t é c l i e n t )* |
| 36 | **i f** ( ! nouvelle Personne . has Error ( ) ) { |
| 37 | *// Ajout de l a Personne au modèle* |
| 38 | myApp. modele . personnes . push ( nouvelle Personne ) ; |
| 39 | *// Provoquer l a s é l ec t i on de l a nou v e l l e personne ( et par s u i t e l a mise à* |
|  | *jour de l a vue )* |
| 40 | myApp. gui . mediator . p u b l i s h ( *” personne/ created ”* , { |
| 41 | personne : nouvelle Personne |
| 42 | }) ; |
| 43 | } |
| 44 | } ; |
| 45 |  |
| 46 | *// Enregistrement du cal l b ack de l ’ événement de v a l i d a t i on du formulaire* |
| 47 | myApp. gui . mediator . s u b s c r i b e ( *” personne/ create ”* , updateModel ) ; |
| 48 |  |
| 49 | }() ] ) ; |

### Code *HTML* de la vue et invocation des méthodes

Il faut surtout penser à inclure jqueryjs le plus tard possible et à invoquer la méthode d’en- registrement des événements utilisateurs après la génération de la vue, qui crée les éléments *HTML* sur lesques on applique ces événements.

Code Source 5.11 : /ihm-demo/index.html

* 1. **<!doctype HTML>**
  2. **<html lang=***” f r ”***>**
  3. **<head>**
  4. **<meta charset=***”UTF 8”* /**>**

−

* 1. **<t i t l e>**A p p l i c a t i o n i n t e r a c t i v e**</ t i t l e>**
  2. **<l i nk rel=***” s t y l e s h e e t ”* **href=***” b as i c S t y l e . css ”* /**>**
  3. </head>
  4. **<body>**
  5. **<***! Paragraphe contenant l e r é s u l t a t s du s c r i p t :* **>**

−− −−

* 1. **<p id=***” paragraphe Resultat ”***></p>**
  2. **<***! Structure de l ’ ap p l i cat i on v ide avec deux mé thodes* **>**

−− −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 04 s t ructure Application . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de sous module r eg ex U t i l de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . /pattern fonct/ex 05 module Pattern Regex . j s ”***></s cript>**

− −

* 1. **<***! Sous module adresse de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . / pattern proto/ ex 05 create Module Metier Proto . j s ”***></s cript>**

− −

* 1. **<***! Mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r* **>**

−− −−

* 1. **<s cript src=***” . . / pattern proto/ ex 05 fabrique Objet Metier Proto . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de sous module adresse de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 06 module Metier Adresse . j s ”***></s cript>**

− −

* 1. **<***! Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse* **>**

−− −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 08 fabrique Adresse . j s ”***></s cript>**

− −

* 1. **<***! Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces* **>**

−− −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 11 interface Implementation . j s ”***></s cript>**

− −

* 1. **<***! Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces* **>**

−− −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 11 interface Implementation Metier . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de f onct i ons d ’ a f f i ch ag e dans myApp. metier . view . adresse* **>**

−− −−

* 1. **<s cript src=***” . . / pattern fonct/ ex 09 adresse View . j s ”***></s cript>**

− −

29

1. **<***! Inclusion de jQuery pour l e s événements et manipulation du DOM* **>**

−− −−

1. **<s cript src=***” j query . j s ”***></s cript>**
2. **<***! Mediator sp é c i a l i s é pour f i l t r e r l e s inputs ( evt ” change ”)* **>**

−− −−

1. **<s cript src=***” . . /form* **f i l t e r** */ ex 02 mediator Input Filter . j s ”***></s cript>**

− −

1. **<***! Géné ration automatique de formulaires avec f i l t r a g e des a t t r i b u t s* **>**

−− −−

1. **<s cript src=***” . . /form* **f i l t e r** */ex03 formsGui . j s ”***></s cript>**

− −

36

1. **<***! Module Mé t i e r myApp. metier . personnes aves c o l l e c t i o n get Adresses ()* **>**

−− −−

1. **<s cript src=***” . /personneModule . j s ”***></s cript>**
2. **<***! Construction en dur d ’ un modèle de donné es : c o l l e c t i o n de personnes* **>**

−− −−

1. **<s cript src=***” . /modelModule . j s ”***></s cript>**
2. **<***! Pattern Mé d i ator pour enchainement arch i t ect u r e t r o i s t i e r s* **>**

−− −−

1. **<s cript src=***” . /mediator . j s ”***></s cript>**

43

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Modifier ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . / gui Bouton Modifier Personne . j s ”***></s cript>**
2. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Supprimer ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . /guiBouton Supprimer Personne . j s ”***></s cript>**
2. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . /gui Bouton Ajouter Personne . j s ”***></s cript>**
2. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . / gui Bouton Ajouter Adresse . j s ”***></s cript>**
2. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . / gui Bouton Modifier Adresse . j s ”***></s cript>**
2. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Supprimer ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . /gui Bouton Supprimer Adresse . j s ”***></s cript>**

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

**<**! Impl émentaton de l ’ action ” Validation de formulaire de modif” **>**

**<s cript src=***” . / gui Modifier Personne Form Validate . j s ”***></s cript>**

−− −−

**<**! Impl émentaton de l ’ action ” Validation de formulaire d ’ ajout ” **>**

−− −−

**<s cript src=***” . / gui Ajouter Personne Form Validate . j s ”***></s cript>**

**<**! Impl émentaton de l ’ action ” Validation de formulaire de modif” **>**

−− −−

**<s cript src=***” . / gui Modifier Adresse Form Validate . j s ”***></s cript>**

**<**! Impl émentaton de l ’ action ” Validation de formulaire d ’ ajout ” **>**

−− −−

**<s cript src=***” . / gui Ajouter Adresse Form Validate . j s ”***></s cript>**

**<**! Mise à jour de l a vue ( panneau ” dé t a i l s ” uniquement ) **>**

−− −−

**<s cript src=***” . / gui Details Changed . j s ”***></s cript>**

**<**! Mise à jour de l a vue ( re géné rer t oute l a vue ) **>**

−− − −−

**<s cript src=***” . /guiPersonneChanged . j s ”***></s cript>**

**<***!−− Code HTML de l a vue −−* Str u c tu r e g éné r a l e de l a **page HTML** *−−***>**

**<button id=***” bouton Ajouter Personne”***>**Ajouter une personne**</button><br** /**>**

**<span id=***” l i s t e Personnes ”* **class=** *” panel ”***></span>**

**<span class=** *” panel ”***>**

**<span id=***” vue Detail ”***>**

</span><br /><br />

**</span>**

**<span id=***” spanMainForm”* **class=** *” panel ”***>**

**<form id=***” ajouter PersonneForm”* **method=***” post ”* **></form>**

**<form id=***” modifierPersonneForm”* **method=***” post ”* **></form>**

**<form id=***” ajouter Adresse Form ”* **method=***” post ”* **></form>**

**<form id=***” modifier AdresseForm ”* **method=***” post ”* **></form>**

</span>

**<**! Inclusion de jQuery pour l e s événements et manipulation du DOM **>**

−− −−

**<s cript src=***” . / jquery . j s ”***></s cript>**

**<**! Événements u t i l i s a t e u r s concernant l e s personnes **>**

−− −−

**<s cript src=***” . /gui JQuery Events Personne . j s ”***></s cript>**

**<**! Événements u t i l i s a t e u r s concernant l e s adresses **>**

−− −−

**<s cript src=***” . / gui JQuery Events Adresse . j s ”***></s cript>**

**<**! Ajout d ’ un main et ex é cution **>**

−− −−

<s cript>

/\*\*

* *Sé r i e d ’ i n s t r u c t i on s e f f e c t u é es pour i n i t i a l i s e r l ’ ap p l i cat i on /*
* *@method mainFunction*
* *@augments myApp*

\*/

myApp. addModule ( *” mainFunction”* , **function** () {

// Personne s é l ect i on n ée par dé f aut

myApp. modele . s e l e c te d Pe r s o n n e **=** myApp. modele . personnes [ 0 ] ;

// Provoquer l e premier a f f i ch ag e de l a vue :

myApp. gui . mediator . p u b l i s h ( *” personne/changed ”* , {

personne : myApp. modele . s e l e c te d Pe r s o n n e

}) ;

// Enregistrement des événements u t i l i s a t e u r s gér é s par jQuery

myApp. gui . init JQuery Events Personne () ; myApp. gui . init JQuery Events Adresse () ;

|  |  |
| --- | --- |
| 112 | }) ; |
| 113 |  |
| 114 | */////////////////////////////////////////////////////* |
| 115 | *// Exé cution du Main avec un t e s t d ’ exception* |
| 116 | *// t ry {* |
| 117  118  119  120 | *// Exé cution de l a mé thode mainFunction*  myApp. mainFunction () ;  *// } catch ( e ){*  *// a l e r t ( e . message ) ;* |
| 121 | *// }* |
| 122 | **</s cript>** |
| 123 | **</body>** |
| 124 | **</html>** |

## Événements concernant les Adresses

### Enregistrement des événements utilisateurs via *jQuery*

De même que pour les personnes, l’utilisation de jQuery est limitée à un module *Wrapper*, qui va définir tous les *handler*.

Comme il peut y avoir plusieurs adresses, dont les éléments *HTML* sont générés dynami-

quement, sur le panneau des détails, les événements concernant les adresses doivent pouvoir être reconstruits dans le cas d’une reconstruction du panneau des détails de la vue (événement personne/detailsRebuilt du *mediator*. De plus, nous devons créer un *handler* pour chacune des adresses de la personne sélectionnée. Ces *handler* seront créés grâce à des *helpers*.

Code Source 5.12 : /ihm-demo/guiJQueryEventsAdresse.js

1 */\*\**

* 1. *\* Mé thode d ’ i n i t i a l i s a t i o n des événements u t i l i s a t e u r s Java Script .*
  2. *\* Enregistrement des g es t i on n a i r es de ces événements v ia jQuery .*

4 *\*/*

5 myApp. addModule . apply (myApp. gui , [ *” init JQuery Events Adresse ”* , **function** () { 6

7 */\*\**

8 \* Gestionnaire c l i c k sur l e bouton f a i s an t s o r t i r l e formulaire

9 *\*/*

1. **var** c l i c k Bo u to n S a i s i e A d r e s s e = **function** ( event ) {
2. *// p u b l i cat i on auprès du mé d i at or*
3. myApp. gui . mediator . p u b l i s h ( *” adresse / s a i s i e ”* , {
4. personne : myApp. modele . s e l e c te d Pe r s o n n e 14 }) ;

15 } ;

16

1. */\*\* Mé thode qui permet de cr é er un g es t i on n a i re d ’ événement de c l i c k*
2. *\* du bouton de suppression sur chaque adresse de l a personne s é l ect i on n ée .*
3. *\* Ces g es t i on n a i r es p u b l i en t l ’ événnement ” nou v e l l e personne s é l ect i on n ée” auprès du mé d i at or .*
4. *\* @param { i n t } index indice de l ’ adresse pour l e q u e l on en r eg i s t r e l ’ évé nement .*

21 *\*/*

1. **var** r e g i s te r H e l p e r Su p p r i m e r A d r e s s e = **function** ( index ) {
2. **return function** ( ) {
3. myApp. gui . mediator . p u b l i s h ( *” adresse / d e l e t e ”* ,

|  |  |
| --- | --- |
| 25 | { |
| 26 | personne : myApp. modele . sel ec te d Per son ne , |
| 27 | a d r e s s e : myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( index ) |
| 28 | }) ; |
| 29 | } ; |
| 30  31 | } ; |
| 32 | */\*\* Mé thode qui permet de cr é er un g es t i on n a i re d ’ événement de c l i c k* |
| 33 | *\* du bouton de suppression sur chaque adresse de l a personne s é l ect i on n ée .* |
| 34 | *\* Ces g es t i on n a i r es p u b l i en t l ’ événnement ” nou v e l l e personne s é l ect i on n ée”* |
|  | *auprès du mé d i ator .* |
| 35 | *\* @param { i n t } index indice de l ’ adresse pour l eq u e l on en r eg i s t r e l ’ événement* |
|  | *.* |
| 36 | *\*/* |
| 37 | **var** r e g i s te r H e l p e r M o d i f i e r A d r e s s e = **function** ( index ) { |
| 38 | **return function** ( ) { |
| 39 | myApp. gui . mediator . p u b l i s h ( *” adresse / ed i t ”* , |
| 40 | { |
| 41 | personne : myApp. modele . sel ec te d Per son ne , |
| 42 | a d r e s s e : myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( index ) |
| 43 | }) ; |
| 44 | } ; |
| 45 | } ; |
| 46 |  |
| 47 | */\*\** |
| 48 | *\* Enregistre l e s événements de c l i c k s sur l e s boutons ” Ajouter une adresse ”* |
|  | *et* |
| 49 | *\* l e s boutons ” Supprimer” ou modifier de t ou t es l e s adresses de l a personne s* |
|  | *é l ect i on n ée .* |
| 50 | *\* Cette fonction d o i t ê t re invoqu ée en cas de s é l ec t i on d ’ une nou v e l l e personne* |
| 51 | *\* ( reconstruction deu code HTML du panneau des dé t a i l s ) .* |
| 52 | *\*/* |
| 53 | **var** r e g i s te r Bu tto n C l i c k Ev e n ts = **function** () { |
| 54 | **var** id Bouton Supprimer Adresse , |
| 55 | id Bouton Modifier Adresse ; |
| 56 | *// Enregistrement du Handler du c l i c k pour ajouter une adresse* |
| 57 | $ ( *”#bouton Ajouter Adresse”* ) . on ( *” c l i c k ”* , c l i c k Bo u to n S a i s i e A d r e s s e ) ; |
| 58 |  |
| 59 | **for** ( **var** i =0 ; i **<** myApp. modele . s e l e c te d Pe r s o n n e . getNb Adresses ( ) ; ++i ) { |
| 60 | id Bouton Supprimer Adresse = *” bouton SupprimerAdresse\_”* + |
| 61 | myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( i ) . g e t A ttr i b u te ( *’ id ’* ) ; |
| 62 | $ ( *”#”* + id Bouton Supprimer Adresse ) . on ( *” c l i c k ”* , |
|  | r e g i s te r H e l p e r Su p p r i m e r A d r e s s e ( i ) ) ; |
| 63 | id Bouton Modifier Adresse = *” bouton Modifier Adresse\_”* + |
| 64 | myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( i ) . g e t A ttr i b u te ( *’ id ’* ) ; |
| 65 | $ ( *”#”* + id Bouton Modifier Adresse ) . on ( *” c l i c k ”* , r e g i s te r H e l p e r M o d i f i e r A d r e s s e |
|  | ( i ) ) ; |
| 66 | } |
| 67 | } |
| 68 |  |
| 69 | *///////////////////////////////////////////////////////* |
| 70  71 | *// Gestionnaire de submit formulaire d ’ ajout de adresse .* |
| 72 | */\*\** |
| 73 | *\* Gestionnaire de l ’ événement submit du formulaire .* |

\* Gestionnaire de l ’ événement submit du formulaire .

|  |  |  |
| --- | --- | --- |
| 74 | *\* @param { Event} jQuery event correspondant au handler .* |  |
| 75 | *\*/* |
| 76 | **var** form Handler AjoutAdresse = **function** ( event ) { |
| 77 |  |
| 78 | *// Év i t e r d ’ appeler l ’” action ” par dé f aut () s c r i p t PHP, etc . . . )* |
| 79 | *// du formulaire l or s du submit* |
| 80 | event . prevent Default ( ) ; |
| 81 |  |
| 82 | *// p u b l i cat i on auprès du mé d i ator* |
| 83 | myApp. gui . mediator . p u b l i s h ( *” adresse / create ”* , { |
| 84 | personne : myApp. modele . s e l e c te d Pe r s o n n e |
| 85 | }) ; |
| 86 | } *// f i n du g es t i on n a i re form Handler Ajout ()* |
| 87 |  |
| 88 | *// Enregistrement du Handler du submit du formulaire v ia jQuery* |
| 89 | $ ( *”#ajouter AdresseForm ”* ) . on ( *” submit ”* , form Handler AjoutAdresse ) ; |
| 90 |  |
| 91 | *///////////////////////////////////////////////////////* |
| 92 | *// Gestionnaire de submit formulaire d ’ ajout de adresse .* |
| 93 |  |
| 94  95 | */\*\** |
| 96  97 | *\* @param { Event} jQuery event correspondant au handler .* |
| 98 | **var** form Handler Modifier Adresse = **function** ( event ) { |
| 99 |  |
| 100 | *// Év i t e r d ’ appeler l ’” action ” par dé f aut () s c r i p t PHP, etc . . . )* |
| 101 | *// du formulaire l or s du submit* |
| 102 | event . prevent Default ( ) ; |
| 103 |  |
| 104 | *// p u b l i cat i on auprès du mé d i ator* |
| 105 | myApp. gui . mediator . p u b l i s h ( *” adresse / update”* , { |
| 106 | personne : myApp. modele . s e l e c te d Pe r s o n n e |
| 107 | }) ; |
| 108  109 | } *// f i n du g es t i on n a i re form Handler Ajout ()* |
| 110 | *// Enregistrement du Handler du submit du formulaire v ia jQuery* |
| 111  112 | $ ( *”#modifier AdresseForm ”* ) . on ( *” submit ”* , form Handler Modifier Adresse ) ; |
| 113 | *// Enregistrer l e s c l i c k s l or s de l ’ i n i t i a l i s a t i o n* |
| 114 | r e g i s te r Bu tto n C l i c k Ev e n ts ( ) ; |
| 115 |  |
| 116 | *// Permet à l a mé thode qui reg é nère l e panneau des dé t a i l s de recr éer ,* |
| 117 | *// v ia l e mé diator , l e s événements ” c l i c k ” sur l e s boutons dans l e panneau* | *des* |
|  | *dé t a i l s .* |  |
| 118 | myApp. gui . mediator . s u b s c r i b e ( *” personne/ d e t a i l s R e b u i l t ”* , |  |
|  | r e g i s te r Bu tto n C l i c k Ev e n ts ) ; |  |
| 119 | } ] ) ; |  |

\*/

### Boutons d’ajout, de suppression, et de modification

##### Le bouton d’ajout d’une adresse, qui existe un un seul exemplaire car il dépend uniquement de la personne, est le plus simple. Il faut créer un formulaire vierge pour la saisie d’une adresse.

Comme pour une personne, on utilise la possibilité de passer null comme argument de la

fabrique d’adresse, qui crée alors une adresse par défaut, sans créer d’erreurs pour les champs vides (même pour les champs obligatoires).

Code Source 5.13 : /ihm-demo/guiBoutonAjouterAdresse.js

1 */\*\**

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e bouton*
  2. *\* ” modifier ” l a personne s é l ect i on n ée .*

4 *\*/*

5 myApp. addModule . apply (myApp. gui , [ *” cal l b ack s Cl i ck A j ou t er ”* , **function** () { 6

7 */\*\**

8 \* Géné ration du code HTML du formulaire de modification de l a personne s é l ect i on n ée .

9 *\*/*

1. **var** getHtmlFormInputs = **function** () {
2. **return** *”***<span** *s t y l e =\”***width** *:360 px ;* **display** *: in l ine b l ock ;* **vertical align** *:*

− −

**top** *;\”***>***”* +

1. *”***<p** *s t y l e =\”***width** *: 360 px ;* **display** *: in l ine b l ock ;* **text align** *:center ;*

− −

**padding** *: 15 px ;\”***>***”* +

1. *”***<tr** *ong***>***S a i s i e d ’ une nou v e l l e adresse***</strong>***”* +
2. *”***<br** */***>***pour ”* + myApp. modele . s e l e c te d Pe r s o n n e . g e t A ttr i b u te ( *”nom”* ) + *”*

**</p>***”* +

1. myApp. gui . getHtmlFormInputs (myApp. metier . a d r e s s e . c r e a te I n s ta n c e ( **null**

) , *” ajouter AdresseForm ”* ) +

1. *”***<l abel></ label><i nput** *type=\” submit \” value=\” Valider \”***></ i nput>***”* + 17 *”***</span>***”* ;

18 }

19

20 */\*\**

1. *\* Callback d ’ Affichage ( v ia l e DOM) du formulaire dans l ’ é l ément d ’ ID ” mainForm”*
2. *\* @param { Object } context Arg non u t i l i s é .*

23 *\*/*

1. **var** repaint Form Inputs = **function** ( contextArg ) {
2. $ ( *”#modifierPersonneForm”* ) . empty () ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
3. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
4. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
5. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*

29

30 $ ( *”#ajouter Adresse Form ”* ) . append ( getHtmlFormInputs ( ) ) ; *// ajouter l e s nouveaux inputs*

31 } ;

32

1. *// Enregistrement du cal l b ack*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / s a i s i e ”* , repaint Form Inputs ) ; 35

36 }() ] ) ;

Les boutons de modification et de suppression des adresse doivent exister en autant d’exem- plaire qu’il y a d’adresse. On crée donc un *helper* chargé de créer le *callback* correspondant à chaque adresse.

1 */\*\**

##### Code Source 5.14 : /ihm-demo/guiBoutonModifierAdresse.js

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e bouton*
  2. *\* ” modifier ” l a personne s é l ect i on n ée .*

4 *\*/*

5 myApp. addModule . apply (myApp. gui , [ *” cal l backs Click Modifier Personne ”* , **function** () { 6

7 */\*\**

8 \* Géné ration du code HTML du formulaire de modification de l a personne s é l ect i on n ée .

9 *\*/*

1. **var** getHtmlFormInputs = **function** ( a d r e s s e ) {
2. **return** *”***<span** *s t y l e =\”***width** *:360 px ;* **display** *: in l ine b l ock ;* **vertical align** *:*

− −

**top** *;\”***>***”* +

1. myApp. gui . getHtmlFormInputs ( adresse , *” modifier AdresseForm ”* ) +
2. *”***<l abel></ label><i nput** *type=\” submit \” value=\” Valider \”***></ i nput>***”* + 14 *”***</span>***”* ;

15 }

16

17 */\*\**

1. *\* Callback d ’ Affichage ( v ia l e DOM) du formulaire dans l ’ é l ément d ’ ID ” modifier AdresseForm ”*
2. *\* Si l ’ adresse a des erreurs , potentiellement , e l l e n ’ a pas é t é cr éée sur l e serveur .*
3. *\* I l f aut a l ors u t i l i s e r l e verbe POST. On ouvre a l ors l ’ é l ément d ’ ID ” ajouter AdresseForm ”*
4. *\* @param { Object } context Arg non u t i l i s é .*

22 *\*/*

1. **var** repaint Form Inputs = **function** ( contextArg ) {
2. $ ( *”#modifierPersonneForm”* ) . empty () ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
3. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
4. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*
5. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; *// Vider l e s inputs et l e s événements JS ex i s t an t*

28

1. $ ( *”#modifier AdresseForm ”* ) . append ( getHtmlFormInputs (
2. contextArg . a d r e s s e ) ) ; *// ajouter l e s*

nouveaux inputs

31 } ;

32

1. *// Enregistrement du cal l b ack*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / ed i t ”* , repaint Form Inputs ) ; 35

36 }() ] ) ;

Code Source 5.15 : /ihm-demo/guiBoutonSupprimerAdresse.js

1

*/\*\**

* *Dé f i n i t i o n et enregistrement des c a l l b ac k s appel é s à gé rer l e c l i c sur l e bouton*
* *” modifier ” l a adresse s é l ect i on n ée .*

*\*/*

2

3

4

5

myApp. addModule . apply (myApp. gui , [ *” callbacks Click Supprimer Adresse ”* , **function** () {

*/\*\**

* *Callback qui supprime l a adresse pass ée dans l ’ ob j e t pass é en argument .*
* *@param { Object } context Arg argument indiquant l a adresse à supprimer .*
* *@param { adresse } context Arg . adresse r é f é rence de l ’ instance de adresse à supprimer dans l e modèle .*

*\*/*

**var** d e l e te A d r e s s e = **function** ( contextArg ) {

*// Suppression de l ’ adresse dans l a personne*

contextArg . personne . d e l e te A d r e s s e ( contextArg . a d r e s s e ) ;

*// Provoquer l a mise à jour de l a vue :*

myApp. gui . mediator . p u b l i s h ( *” personne/ details Changed ”* ,

a d r e s s e : myApp. modele . s e l e c te d Pe r s o n n e

{

}

}) ;

*// Enregistrement du cal l b ack*

myApp. gui . mediator . s u b s c r i b e ( *” adresse / d e l e t e ”* , d e l e te A d r e s s e ) ;

}() ] ) ;

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

### Création d’une nouvelle adresse

L’adresse est automatiquement ajoutée à la personne sélectionnée, et son *ID* est généré auto- matiquement. Comme dans le cas d’une personne, les attributs de l’adresse (autre que l’*ID*) sont récupérées à partir des valeurs des *inputs* du formulaire.

Code Source 5.16 : /ihm-demo/guiAjouterAdresseFormValidate.js

1

*/\*\**

* *Dé f i n i t i o n et enregistrement du cal l b ack r é ag i s s san t à l a v a l i d a t i on ( submit )*
* *du formulaire de modification d ’ une adresse .*

*\*/*

myApp. addModule . apply (myApp. gui , [ *” callbacks Validate Ajouter Adresse Form ”* ,

**function** ( ) {

*/\*\**

*\* Modifie l e modèle à p ar t i r des donné es s a i s i e s dans l e formulaire*

*\*/*

**var** updateModel = **function** () {

*// 1 ) Mise à jour des donné es du modèle*

*// à p ar t i r des valeurs des inputs du formulaire*

**var** attribute Name , input Id ;

*// Ajout d ’ un adresse v ide dans l a c o l l e c t i o n*

**var** n o u v e l l e A d r e s s e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ( **null** ) ; myApp. modele . s e l e c te d Pe r s o n n e . add Adresse ( n o u v e l l e A d r e s s e ) ;

*// Pour chaque propri é t é ( chaque input du formulaire )*

**for** ( **var** j=0 ; j**<** myApp. metier . a d r e s s e . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++j ) {

**var** attribute Name = myApp. metier . a d r e s s e . g e t A ttr i b u te L i s t ( ) [ j ] ;

**i f** ( attribute Name != *” id ”* ) {

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33 }

34 }

// cal cu l de l ’ ID de l ’ input

**var** input Id = myApp. gui . get Input Id ({

attribute Name : attribute Name , form Id : *” ajouter AdresseForm ”*

}) ;

// Modification de l a propri é t é de l a adresse

// avec l a valeur s a i s i e dans l ’ input .

n o u v e l l e A d r e s s e . s e t A ttr i b u te ( attribute Name ,

document . getElementBy Id ( input Id ) . value

) ;

35

36

37

38

39

40

41

42

43

44

45

46

47

48 } ;

49

**i f** ( ! n o u v e l l e A d r e s s e . has Error ( ) ) {

// Provoquer l a mise à jour de l a vue ( panneau des dé t a i l s )

myApp. gui . mediator . p u b l i s h ( *” personne/ details Changed ”* , {

personne : myApp. modele . s e l e c te d Pe r s o n n e

}) ;

// Provoquer l a requ ê t e AJAX pour l ’ impl é mentation de l a persist an ce

myApp. gui . mediator . p u b l i s h ( *” adresse / created ”* , {

personne : myApp. modele . se l e cted Per so nn e , a d r e s s e : n o u v e l l e A d r e s s e

}) ;

} **else** {

myApp. modele . s e l e c te d Pe r s o n n e . d e l e te A d r e s s e ( n o u v e l l e A d r e s s e ) ;

}

1. *// Enregistrement du cal l b ack de l ’ événement de v a l i d a t i on du formulaire*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / create ”* , updateModel ) ; 52

53 }() ] ) ;

### Modification d’une adresse

La modification d’une adresse après modification présente la difficulté suivante : il faut retrouver l’instance d’adresse à modifier, parmis les adresses de la personne sélectionnée. Nous avons choisi de mettre un champs caché avec l’*ID* dans le formulaire (voir la partie [4.3](#_bookmark87)). Il nous faut alors rechercher l’*ID* de l’adresse dans les instances d’adresse de la personne sélectionnée. Nous aurions aussi pu ajouter une référence vers l’adresse éditée dans le modèle.

Code Source 5.17 : /ihm-demo/guiModifierAdresseFormValidate.js

1

*/\*\**

* *Dé f i n i t i o n et enregistrement du cal l b ack r é ag i s s san t à l a v a l i d a t i on ( submit )*
* *du formulaire de modification d ’ une personne .*

*\*/*

myApp. addModule . apply (myApp. gui , [ *” callbacks Validate Modifier Adresse Form ”* ,

**function** ( ) {

*// Formulaire de modification d ’ une personne*

*/\*\**

*\* Modifie l e modèle à p ar t i r des donné es s a i s i e s dans l e formulaire*

*\*/*

**var** updateModel = **function** () {

2

3

4

5

6

7

8

9

10

11

12

1. *// 1 ) Mise à jour des donné es du modèle*
2. *// à p ar t i r des valeurs des inputs du formulaire*
3. **var** attribute Name ,
4. input Id ;

17

1. *// Recherche de l ’ adresse qui a é t é modifi ée à p ar t i r de son ID unique*
2. *// L ’ ID se t rouve en champs cach é du formulaire .*
3. **var** input Id\_ id = myApp. gui . get Input Id ({
4. attribute Name : *” id ”* ,
5. form Id : *” modifier AdresseForm ”*

23 }) ;

24

1. *// ID unique de l ’ adresse concern ée par l e formulaire*
2. **var** id Adresse = document . getElementBy Id ( input Id\_ id ) . value ;
3. **var** adresse En Question ; *// Ré f é rence de l ’ adresse concern ée par l e formulaire*
4. **for** ( **var** i = 0 ; i **<** myApp. modele . s e l e c te d Pe r s o n n e . getNb Adresses ( ) ; ++i ) {
5. **i f** ( id Adresse == myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( i ) . g e t A ttr i b u te (

*’ id ’* ) ) {

1. adresse En Question = myApp. modele . s e l e c te d Pe r s o n n e . get Adresse ( i ) ; 31 }

32 }

1. **i f** ( adresse En Question === undefined ) {
2. **throw new** Error ( *” Adresse in t rou v ab l e ( ID i n ex i s t an t )”* ) ; 35 }
3. *// On r é cupère l e s a t t r i b u t s du formulaire dans une nou v e l l e instance*
4. **var** changed Adresse = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ( **null** ) ;
5. *// Pour chaque propri é t é ( chaque input du formulaire )*
6. **for** ( **var** j=0 ; j**<** myApp. metier . a d r e s s e . g e t A t tr i b u te L i s t ( ) . l e n g th ; ++j ) {
7. **var** attribute Name = myApp. metier . a d r e s s e . g e t A ttr i b u te L i s t ( ) [ j ] ;
8. **i f** ( attribute Name != *” id ”* ) {
9. *// cal cu l de l ’ ID de l ’ input*
10. **var** input Id = myApp. gui . get Input Id ({
11. attribute Name : attribute Name ,
12. form Id : *” modifier AdresseForm ”*

46 }) ;

1. *// Modification de l a propri é t é de l a personne*
2. *// avec l a valeur s a i s i e dans l ’ input .*
3. changed Adresse . s e t A ttr i b u te ( attribute Name ,
4. document . getElementBy Id ( input Id ) . value

51 ) ;

52 }

53 }

1. *// Seulement s ’ i l n ’ y a pas d ’ erreur ( f i l t r a g e s t r i c t cô t é c l i e n t )*
2. **i f** ( ! changed Adresse . has Error ( ) ) {
3. **for** ( **var** j=0 ; j**<** myApp. metier . a d r e s s e . g e t A ttr i b u te L i s t () . l e n g th ; ++j ) {
4. **var** attribute Name = myApp. metier . a d r e s s e . g e t A t tr i b u te L i s t ( ) [ j ] ;
5. **i f** ( attribute Name != *” id ”* ) {
6. adresse En Question . s e t A ttr i b u te ( attribute Name ,
7. changed Adresse . g e t A ttr i b u te ( attribute Name ) ) ; 61 }

62 }

1. *// Provoquer l a mise à jour des é l éments de l a vue observant l a personne*
2. myApp. gui . mediator . p u b l i s h ( *” personne/ details Changed ”* , {
3. personne : **null**

66 }) ;

67 // Provoquer l a mise à jour des é l éments de l a vue observant l a personne

68

}

myApp. gui . mediator . p u b l i s h ( *” adresse / changed ”* , {

personne : myApp. modele . se l e cted Per so nn e , a d r e s s e : adresse En Question

}) ;

*// Enregistrement du cal l b ack de l ’ événement de v a l i d a t i on du formulaire*

myApp. gui . mediator . s u b s c r i b e ( *” adresse / update”* , updateModel ) ;

}() ] ) ;

} ;

69

70

71

72

73

74

75

76

**Chapitre 6**

**Requêtes Asynchrones et *API Restful***

## Qu’est-ce qu’une requête asynchrone ?

##### Les requêtes asynchrones *XMLHttpRequest* permettent d’exécuter (suite à une événement côté client) une requête *HTTP* (exécution d’un script ou programme, par exemple en *PHP*) sur le serveur. On parle de requête *asynchrone* car le client n’est pas bloqué en attendant la réponse du serveur : le déroulement du programme côté client peut se poursuivre, et la réponse du serveur est gérée par des *callbacks*.

Malgré le nom *XMLHttpRequest*, les requêtes asynchrones permettent d’échanger avec le serveur d’autres types de données que du *XML*. Nous utiliserons dans ce cours des données *JSON*.

Le codage *JSON* perrmet de coder sous forme de chaîne de caractères des collections d’ob- jets. Ainsi, on pourra, par exemple, coder en *JSON* une collection d’objets en *PHP* (tableau associatif), puis transmettre la chaîne *JSON* via une requête asynchrone, et enfin reconstituer une collection d’objets en *JavaScript* pour générer, par exemple, une mise en forme *HTML* dans le document.

Voici un exemple de code *JSON* d’un tableau associatif *PHP* ou d’un objet *JavaScript* (qui contient lui-même un tableau de descriptions de formats) :

Code Source 6.1 : Code *JSON* d’un tableau associatif *PHP* ou objet *JavaScript*

{

*” id ”* : 654 ,

*” denomination ”* : *” Tutoriel Java Script ”* ,

*” prix u n i t a i r e ”* : 0 . 5 0 ,

*” formats ”* : [ *”PDF”* , *” Post s crip t ”* , *”HTML”* , *” ePub”* ]

}

1

2

3

4

5

6

On peut, par exemple, générer un tel tableau sur un serveur en *PHP* par le code suivant : Code Source 6.2 : Génération et sortie du code *JSON* en *PHP*

<?php

$myArray = **array** ( *” id ”* => 654 ,

*” denomination ”* => *” Tutoriel Java Script ”* ,

*” prix u n i t a i r e ”* => 0 . 5 0 ,

*” formats ”* => **array** ( *”PDF”* , *” Post s crip t ”* , *”HTML”* , *” ePub”* ) ) ;

**echo** json\_encode ( $myArray ) ;

**header** ( *’ content−type : ap p l i cat i on / json ; charset=utf −8 ’* ) ;

*// Header HTTP sp é c i f i q u e :*

1

2

3

4

5

6

7

8

9

?>

* 1. **Requêtes *Ajax***

##### Le méthode ajax de *jQuery* perrmet d’effectuer une requête *XMLHttpRequest* qui transmet des paramètres (un objet *JavaScript*) à un *CGI* (ici en *PHP*), via une *URL*. Dans notre exemple, le serveur reçoit lui-même un objet (propriété data) côdé en *JSON*, et génère lui-même du code *JSON*. Le programme client récupère du code *JSON* générée sur la sortie standard du *CGI*, et reconstitue une objet *JavaScript*.

Voici notre exemple où le code *JavaScript* côté client récupère une collection d’objet créée par le *CGI* et la met en forme en *HTML*. Trois boutons permettent de tester :

* + - Un cas sans erreur ;
    - Un cas où la gestion d’erreur est implémentée en *PHP* côté serveur ;
    - Un cas où la requête *AJAX* elle-même échoue.

Les trois *callbacks* suivants sont utilisés pour gérer la requête :

* + - success en cas de succès de la requête ;
    - error en cas d’échec de la requête

complete, ici utilisé pour mettre à jour la vue, que ce soit en cas de succès ou en cas d’échec de la requête.

*•*

Je programme en *JavaScript* côté client est le suivant :

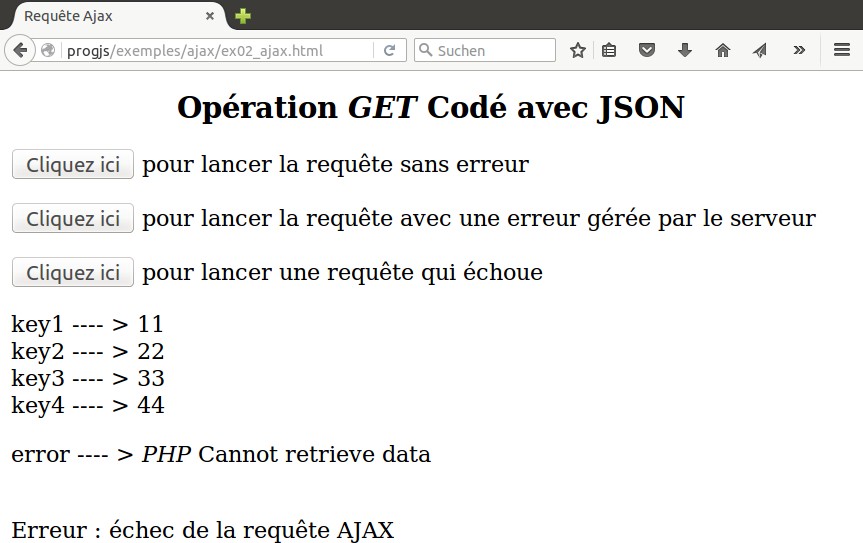


Figure 6.1 : Illustration du code source [6.3](#_bookmark115)

Code Source 6.3 : /ajax/ex02-ajax.html (cf. Fig [6.1](#_bookmark114))

1. <!doctype html>
2. **<html lang=***” f r ”***>**
3. <head>
4. **<meta charset=***” utf 8 ”***>**

−

1. **<t i t l e>**Requ ê te Ajax**</ t i t l e>**
2. **<s cript src=***” . / jquery . j s ”***></s cript>**
3. **<l i nk rel=***” s t y l e s h e e t ”* **href=***” b as i c S t y l e . css ”* /**>**
4. </head>

9

1. **<body>**
2. **<h1>**Opé r a t i o n **<i>**GET**</ i>** Codé avec JSON**</h1>** 12
3. **<p><button** onclick= *” l ancer Requete ( 1 ) ”***>**Cliquez i c i**</button>** pour l a n c e r l a requ ê te sans e r r e u r**</p>**
4. **<p><button** onclick= *” l ancer Requete ( 0 ) ”***>**Cliquez i c i**</button>** pour l a n c e r l a requ ê te avec une e r r e u r g é r é e par l e s e r v e u r**</p>**
5. **<p><button** onclick= *” l ancer Requete ( 1)”***>**Cliquez i c i**</button>** pour l a n c e r une requ ê te qui é choue**</p>**

−

16

1. **<p id=***” output Paragraph ”***></p>**
2. <s cript>

19

1. **var** model **=** {
2. paragraph Text : *””* ,
3. **error** : **null** ,
4. get Error Message : **function** () {
5. **return this** . **error** !== **null** ? *”***<br** */***>***”* + **this** . **error** : *””* ; 25 }

26 } ;

27

28 */\*\**

1. *\* fonction cal l b ack ex é cut ée en cas de succès de l a requ ê t e AJAX.*
2. *\* La mé thode parcourt l e s donné es retourn é es par l e serveur au format JSON,*
3. *\* et concatène l e t ex t e dans l e modèle .*
4. *\* @param { Object } retrieved Data : c o l l e c t i o n des donné es décod é es à p ar t i r du JSON.*
5. *\* La donnée peut ê t re un message d ’ erreur .*

34 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s **= function** ( r e tr i e v e d D a ta ) {
2. model . e r r o r = n u l l ;
3. model . paragraph Text **=** *””* ;
4. *// Parcours et a f f i ch ag e des donné es de l ’ ob j e t*
5. **for** ( **var** key **in** r e tr i e v e d D a ta ) {
6. } model . paragraph Text +**=** key + *” −−−−* **>** *”* + r e tr i e v e d D a ta [ key ] +’**<br** /**>**’ ;

41

42 } ;

43

44 */\*\**

1. *\* fonction cal l b ack ex é cut ée en cas d ’ é chec de l a requ ê t e AJAX.*
2. *\* Une erreur es t ajout ée dans l e modèle et l e t ex t e du paragraphe es t mis à v ide .*

47 *\*/*

1. **var** ajax Callback Error **= function** () {
2. model . paragraph Text **=** *””* ;
3. model . **error =** *” Erreur : é chec de l a requ ê t e AJAX”* ;

51 } ;

52

53 */\*\**

1. *\* fonction cal l b ack ex é cut ée l orsque l a requ ê t e AJAX se termine .*
2. *\* Ce cal l b ack es t appel é en cas d ’ é chec ET en cas de succès de l a requ ê t e AJAX.*
3. *\* Ici , l a mé thode met à jour l a vue en a f f i ch an t l e t ex t e et une é v en t u e l l e erreur .*

57 *\*/*

1. **var** ajax Callback Complete **= function** () {
2. $ ( *”#output Paragraph ”* ) . append ( 60 *”***<p>***”* +
3. model . paragraph Text +
4. model . get Error Message ( ) + 63 *”***</p>***”* ) ;

64 }

65

66 */\*\**

1. *\* Gestionnaire de c l i c k sur l e s boutons , qui dé c l enche une requ ê t e AJAX.*
2. *\* @param { i n t } simple Test Value donnée transmise au serveur v ia l a propri é t é s imple Test*
3. *\* s i simple Test Value es t né g a t i f , une URL du serveur i n ex i s t an t e es t u t i l i s ée ,*
4. *\* provoquant l ’ é chec de l a requ ê t e ( c ’ es t j u s t e pour l ’ exemple*

. . . ) .

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

*\*/*

**var** lancer Requete **= function** ( s imple Test Value ) {

**var** u r l S e r v e u r **=** *” h t t p :// progjs/ exemples/ ajax/ ex 01 \_ encode\_ json . php”* ;

// Pour provoquer une requ ê t e qui échoue complètement

**i f** ( s imple Test Value **<**0 ) { *// URL qui n ’ ex i s t e pas*

u r l S e r v e u r **=** *” h t t p :// progjs/ exemples/ ajax/ b idon . php”* ;

}

// Lancement d ’ une requ ê t e AJAX avec donné es (POST) cod ée en JSON

**var** jqxhr **=** $ . ajax ({

// Envoyer l e s donné es de l a personne avec l e format JSON

dataType : *” json ”* ,

u r l : url Serveur , *// URL du serveur*

**method** : ’ post ’ , *// Envoyer l e s donné es dans l e t ableau $\_POST*

**contentType** : ’ application/x www form urlencoded ’ ,

− − −

// donné es à t ransmettre au serveur

**data** : {

s imple Test : s imple Test Value

} ,

// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès

s u c c e s s : ajax Callback Success ,

// Mé thode cal l b ack qui gère une évé en t u e l l e erreur dans l a requ ê t e

**error** : ajax Callback Error ,

// Mé thode cal l b ack qui met à jours l a vue l a vue en cas de succès ou d ’ erreur

complete : ajax Callback Complete

}) ;

}

101

102

1

2

3

4

5

6

7

8

9

10

11

**</s cript>**

**</body>**

Le programme en *PHP* côté serveur est le suivant :

Code Source 6.4 : /ajax/ex01-encode-json.php

<?php

**i f** ( **i s s e t** ($\_REQUEST[ *’ s imple Test ’* ] ) && $\_REQUEST[ *’ s imple Test ’* ] == 1 ) {

$myArray = **array** ( *’ key 1 ’* => 11 , *’ key 2 ’* => 22 , *’ key 3 ’* => 33 , *’ key 4 ’* => 44 ) ;

} **else** {

$myArray = **array** ( *’ error ’* => *”<i >PHP</i > Cannot r e t r i ev e data”* ) ;

}

*// Header HTTP*

**echo** json\_encode ( $myArray ) ;

?>

**header** ( *’ content−type : ap p l i cat i on / j son ; charset=utf −8 ’* ) ;

* 1. **Qu’est-ce qu’une *API REST* (ou systèmes *Restful*) ?**

Les explications sur les *Web Services* de type *API Restful* se trouvent dans le cours de pro- grammation *Web* côté serveur sur :

<https://malgouyres.org/programmation-php>

Nous rappelons ici quelques principes. L’architecture *REST (representational state transfer)*

##### est, dans notre cadre, une architecture d’application client-serveur, qui permet le lien entre une application côté client en *Javascript* et un serveur web sur lequel s’exécutent des *CGI*.

Le serveur permettra (au moins) d’effectuer au moins les opérations *CRUD* (*Create*, *Read*, *Update*, *Delete*) sur des instances d’objets *métier*, aussi appelées *entités* ou *ressources* :

##### **Opération *Create***. De créer une ressource (ici une ligne d’une table de base de données)

*•*

**avec ou sans son identifiant** unique.

*Exemple 1 :* Créer une ressource de type Adresse en spécifiant les données de l’adresse, en laissant au serveur le choix de l’*Id* de la ressource créée. Le serveur retourne l’*Id* généré pour que le client le connaisse.

*Exemple 2 :* Créer une ressource de type Adresse en spécifiant les données de l’adresse **ET** l’identifiant unique de l’instance à créer, par exemple parce que cet *Id* doit être généré par un algorithme dépendant du client, ou parce que cet *Id* doit correspondre à l’*Id* de la même entité ailleurs sur le réseau (comme l’*ISBN* d’un livre, qui ne peut pas être choisi au hasard par le serveur).

**Opération *Read***. De lire toutes les ressources (ici d’une table de base de données). *Exemple :* Lire toutes les personnes de la table Personne, avec une collection d’adresses pour chaque personne (résultat d’une jointure en *SQL* qui correspond à un agrégat sur les objets métiers).

*•*

**Opération *Read* avec *Id* ou prédicat**. De lire **ou bien** une ressource identifiée de manière unique par un identifiant unique (une ligne d’une table de base de données) **ou bien** un certain nombre de ressources données par le résultat d’une requête (comme un SELECT en *SQL*) ou par les données d’une jointure (par exemple avec l’identifiant d’un

*•*

agrégat).

*Exemple 1 :* Lire l’adresse d’identifiant unique (clé primaire de la table Adresse égaler à

af49bc053de73a0.

*Exemple 2 :* Lire toutes les adresses de la personne d’identifiant unique bd56bc053de12b3. *Exemple 3 :* Lire toutes les personnes de la table Personne qui ont une adresse avec le code postal commençant par les deux chiffre 63.

**Opération *Update***. De mettre à jour une ressource (ici une ligne d’une table de base de données) identifiée de manière unique (par un identifiant unique), avec des données (partielles ou complètes) à modifier.

*•*

*Exemple :* Modifier le code postal d’une adresse d’identifiant unique égal à af49bc053de73a0.

**Opération *Delete***. De détruire une ressource (ici une ligne d’une table de base de don- nées) identifiée de manière unique (par un identifiant unique) ;

*•*

*Exemple :* Détruire la personne d’identifiant unique bd56bc053de12b3, ainsi que (s’agis-

sant d’une composition) toutes ses adresses de la table adresse (utilisation d’une clé étrangère).

En utilisant cette interface (*service web*), l’application côté client pourra accéder à la couche persistance du serveur.

Nous voyons aussi comment implémenter ces opérations sur le serveur en spécifiant les iden- tifiants et les actions au moyen d’une *URI* (*Universal Ressource Identifier*) et des *verbes* (aussi appelés *méthodes*, GET, PUT, POST, PATCH ou DELETE du protocole *HTTP* (norme *RFC 2616* puis *RFC 7230*).

* 1. **Persistance par Requêtes sur une *API Restful***

### Création du Module persistance et Objet statusCode

##### À la racine d’un nouveau module myApp.persistance, nous créons un objet statusCodeObject, destiné à définir les méthodes *callback* de gestion d’un retour de code d’erreur (*status code*) *HTTP*.

Code Source 6.5 : /clientAndAPI/client/persistanceCommon.js

1

*/\*\**

* *Dé f i n i t i o n du module charg é de l a persitance par requ ê t es AJAX*
* *sur une API Rest f u l permettant d ’ acc é der à des en t i t é s et de*
* *s t ocker des en t i t é s .*

*\*/*

myApp. addModule ( *” persist an ce ”* , {}) ;

*/\*\**

* *Dé f i n i t i o n et de l ’ ob j e t dé f i n i s s a n t l e s mé thode cal l b ack*
* *correspondant aux d i f f é rents codes d ’ erreur ( s t a t u s codes ) HTTP.*

*\*/*

myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” s tatus Code Object ”* , { 404 : **function** () {

a l e r t ( *” Resource not found ”* ) ;

} ,

400 : **function** () {

a l e r t ( *”Bad Request ”* ) ;

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

} ] ) ;

405 : **function** () {

a l e r t ( *” Method Not Allowed ”* ) ;

} ,

422 : **function** () {

a l e r t ( *” Unprocessable Entity : A t t r i b u t in correct ?”* ) ;

} ,

500 : **function** () {

a l e r t ( *” In t ern a l Server Error”* ) ;

}

} ,

19

20

21

22

23

24

25

26

27

28

### Construction du modèle à partir de la base de données

Code Source 6.6 : /clientAndAPI/client/persistanceRead.js

1 */\*\**

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de chargement du modèle*
  2. *\* à p ar t i r des donné es sur l e serveur par une requ ê t e AJAX.*
  3. *\* Permet l e chargement du modèle à p ar t i r de l a base de donné es .*

5 *\*/*

6 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Rebuild Model From Server ”* ,

**function** ( ) {

7 */\*\**

1. *\* Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
2. *\* Cette mé thode r econ s t ru i t l e modèle à p ar t i r des donné es du serveur .*
3. *\* @param { Object } retrieved Data donné es re ¸ ues du serveur ( après parsing du JSON)*
4. *\* @param { Object | n u l l } retrieved Data . error n u l l en l ’ absence d ’ erreur dé t ec t é e par l e serveur ,*
5. *\* ou un ob j e t dont l e s propri é t é s sont l e s messages d ’ erreur renvoy é es par l e serveur .*
6. *\* @param { Object } retrieved Data . data donné es renvoy é es par l e serveur :*
7. *\* c o l l e c t i o n s d ’ ob j e t s permettant de construire des personnes*

, avec l eu r s adresses .

15 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {
2. **var** adresses Data , a d r e s s e I n s ta n c e ; 18
3. *// Si aucune erreur n ’ a é t é dé t ec t ée sur l e serveur*
4. **i f** ( r e tr i e v e d D a ta [ *” error ”* ] === **null** && r e tr i e v e d D a ta [ *’ data ’* ] !== undefined ) {
5. *// Parcours des ob j e t s dans l e s donné es*
6. **for** ( **var** key **in** r e tr i e v e d D a ta [ *’ data ’* ] ) {
7. **i f** ( r e tr i e v e d D a ta [ *’ data ’* ] . hasOwnProperty ( key ) ) {
8. *// Cré ation d ’ une personne sans adresse*
9. **var new**Personne = myApp. metier . personne . c r e a te I n s ta n c e ({
10. id : r e tr i e v e d D a ta [ *’ data ’* ] [ key ] [ *” id ”* ] ,
11. nom : r e tr i e v e d D a ta [ *’ data ’* ] [ key ] [ *”nom”* ] 28 }) ;

29

1. *// Parcours des ob j e t s dé f i n i s s a n t l e s adresses*
2. adresses Data = r e tr i e v e d D a ta [ *’ data ’* ] [ key ] [ *” adresses ”* ] ;
3. **for** ( **var** key Adresse **in** adresses Data ) {
4. **i f** ( adresses Data . hasOwnProperty ( key Adresse ) ) {
5. *// Cré ation et ajout d ’ une adresse*

35

36

37

38

39

40

41

42

43

44

45

46 }

a d r e s s e I n s ta n c e = myApp. metier . a d r e s s e . c r e a te I n s ta n c e ({ id : adresses Data [ key Adresse ] [ *” id ”* ] ,

numeroRue : adresses Data [ key Adresse ] [ *” numeroRue”* ] , rue : adresses Data [ key Adresse ] [ *” rue”* ] ,

complementAddr : adresses Data [ key Adresse ] [ *” complementAddr”* ] , code Postal : adresses Data [ key Adresse ] [ *” code Postal ”* ] ,

v i l l e : adresses Data [ key Adresse ] [ *” v i l l e ”* ] , pays : adresses Data [ key Adresse ] [ *” pays ”* ]

}) ;

**new**Personne . add Adresse ( a d r e s s e I n s ta n c e ) ;

}

47

48

49 }

myApp. modele . personnes . push (**new**Personne ) ; *// ajout dans l e modèle*

}

50

51

52

53 } ;

54

} **else** {

a l e r t ( *” I l y a un ob j e t \” error \” non n u l l ”* ) ;

}

55 */\*\**

1. *\* Mé thode appel ée l orsqu e l a requ ê t e AJAX se termine ,*
2. *\* que ce s o i t après une erreur ou après un succès .*
3. *\* Cette mé thode r econ s t ru i t l a vue ( après reconstruction du modèle ) .*

59 *\*/*

60 **var** ajax Callback Complete = **function** ( r e tr i e v e d D a ta ) { 61

1. *// Personne s é l ect i on n ée par dé f aut*
2. myApp. modele . s e l e c te d Pe r s o n n e = myApp. modele . personnes [ 0 ] ; 64
3. *// La vue es t r é i n i t i a l i s ée : on v ide l e s é l éments et événements*
4. $ ( *”#l i s t e Personnes ”* ) . empty ( ) ;
5. $ ( *”#vue Detail ”* ) . empty ( ) ;
6. $ ( *”#ajouter PersonneForm”* ) . empty ( ) ;
7. $ ( *”#modifierPersonneForm”* ) . empty () ;
8. $ ( *”#ajouter Adresse Form ”* ) . empty ( ) ;
9. $ ( *”#modifier AdresseForm ”* ) . empty ( ) ; 72
10. *// Provoquer l e premier a f f i ch ag e de l a vue :*
11. myApp. gui . mediator . p u b l i s h ( *” personne/ changed”* , {
12. personne : myApp. modele . s e l e c te d Pe r s o n n e 76 }) ;

77

1. *// Enregistrement des événements u t i l i s a t e u r s gér é s par jQuery*
2. myApp. gui . init JQuery Events Personne ( ) ;
3. myApp. gui . init JQuery Events Adresse ( ) ; 81 } ;

82

83 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

87 *\*/*

1. **var** read All Personne = **function** () {
2. *// requ ê t e AJAX get cod é en JSON*
3. **var** jqxhr = $ . ajax ({

91

*// Envoyer l e s donné es de l a personne avec l e format JSON*

dataType : *” json ”* ,

u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / personne”* , *// URL du serveur*

method : *’ get ’* , *// Verbe HTTP*

contentType : *’ ap p l i cat i on /x−www−form−u r l encoded ’* ,

} ;

*// action : ” personne−get−a l l ”*

} ,

*// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès*

s u c c e s s : ajax Callback Success ,

*// Mé thode cal l b ack qui gère une évé en t u e l l e erreur dans l a requ ê t e*

complete : ajax Callback Complete ,

*// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*

status Code : myApp. p e r s i s t a n c e . status Code Object

}) ;

*// donné es à t ransmettre au serveur*

data : {

*// Enregistrement du cal l b ack de l ’ événement de reconstruction du modèle*

myApp. gui . mediator . s u b s c r i b e ( *” personne/ read ”* , read All Personne ) ;

}() ] ) ;

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

Code Source 6.7 : /clientAndAPI/client/index.html

* 1. **<!doctype HTML>**
  2. **<html lang=***” f r ”***>**
  3. **<head>**
  4. **<meta charset=***”UTF 8”* /**>**

−

* 1. **<t i t l e>**A p p l i c a t i o n i n t e r a c t i v e**</ t i t l e>**
  2. **<l i nk rel=***” s t y l e s h e e t ”* **href=***” b as i c S t y l e . css ”* /**>**
  3. </head>
  4. **<body>**
  5. **<***! Structure de l ’ ap p l i cat i on v ide avec deux mé thodes* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 04 s t ructure Application . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de sous module r eg ex U t i l de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . / . . /pattern fonct/ ex 05 module Pattern Regex . j s ”***></s cript>**

− −

* 1. **<***! Sous module adresse de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . / . . / pattern proto/ ex 05 create Module Metier Proto . j s ”***></s cript>**

− −

* 1. **<***! Mé thode f ab r i q u e géné r i que d ’ ob j e t s mé t i e r* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern proto/ ex 05 fabrique Objet Metier Proto . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de sous module adresse de myApp. metier* **>**

−− − −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 06 module Metier Adresse . j s ”***></s cript>**

− −

* 1. **<***! Cré ation d ’ une mé thode f ab r i q u e d ’ adresse de myApp. metier . adresse* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 08 fabrique Adresse . j s ”***></s cript>**

− −

* 1. **<***! Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 11 interface Implementation . j s ”***></s cript>**

− −

* 1. **<***! Classe de vé r i f i c a t i o n de l ’ impl é mentation d ’ i n t e r f aces* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 11 interface Implementation Metier . j s ”***></s cript>**

− −

* 1. **<***! Cré ation de f onct i ons d ’ a f f i ch ag e dans myApp. metier . view . adresse* **>**

−− −−

* 1. **<s cript src=***” . . / . . / pattern fonct/ ex 09 adresse View . j s ”***></s cript>**

− −

27

28

29

1. **<***!−− Mediator sp é c i a l i s é pour f i l t r e r l e s inputs ( evt ” change ”) −−***>**
2. **<s cript src=***” . . / . . /form−* **f i l t e r** */ ex 02 −mediator Input Filter . j s ”***></s cript>**
3. **<***! Géné ration automatique de formulaires avec f i l t r a g e des a t t r i b u t s* **>**

−− −−

1. **<s cript src=***” . . / . . /form* **f i l t e r** */ex03 formsGui . j s ”***></s cript>**

− −

34

1. **<***! Module Mé t i e r myApp. metier . personnes aves c o l l e c t i o n get Adresses ()* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/personneModule . j s ”***></s cript>**

−

1. **<***! Construction en dur d ’ un modèle de donné es : c o l l e c t i o n de personnes* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/modelModule . j s ”***></s cript>**

−

1. **<***! Pattern Mé d i ator pour enchainement arch i t ect u r e t r o i s t i e r s* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/mediator . j s ”***></s cript>**

−

41

1. **<***! Inclusion de jQuery pour l e s événements et manipulation du DOM* **>**

−− −−

1. **<s cript src=***” j query . j s ”***></s cript>**
2. **<***! Événements u t i l i s a t e u r s concernant l e s personnes* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiJQueryEventsPersonne . j s ”***></s cript>**

−

1. **<***! Événements u t i l i s a t e u r s concernant l e s adresses* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiJQueryEventsAdresse . j s ”***></s cript>**

−

48

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Modifier ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBoutonModifierPersonne . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Supprimer ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBoutonSupprimerPersonne . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Personne”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBoutonAjouterPersonne . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBoutonAjouterAdresse . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Ajouter ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBouton ModifierAdresse . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” c l i c k sur ’ Supprimer ’ de Adresse”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiBoutonSupprimerAdresse . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” Validation de formulaire de modif”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiModifierPersonneFormValidate . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” Validation de formulaire d ’ ajout ”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiAjouterPersonneFormValidate . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” Validation de formulaire de modif”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiModifier AdresseForm Validate . j s ”***></s cript>**

−

1. **<***! Impl émentaton de l ’ action ” Validation de formulaire d ’ ajout ”* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiAjouterAdresseForm Validate . j s ”***></s cript>**

−

1. **<***! Mise à jour de l a vue ( panneau ” dé t a i l s ” uniquement )* **>**

−− −−

1. **<s cript src=***” . . / . . /ihm demo/guiDetailsChanged . j s ”***></s cript>**

−

1. **<***! Mise à jour de l a vue ( re géné rer t oute l a vue )* **>**

−− − −−

1. **<s cript src=***” . . / . . /ihm demo/guiPersonneChanged . j s ”***></s cript>**

−

73

1. **<s cript src=***” . /persistanceCommon . j s ”***></s cript>**
2. **<s cript src=***” . / persistance Read . j s ”***></s cript>**
3. **<s cript src=***” . / persistance Create Personne . j s ”***></s cript>**
4. **<s cript src=***” . / persistance Delete Personne . j s ”***></s cript>**
5. **<s cript src=***” . / persistance Update Personne . j s ”***></s cript>**
6. **<s cript src=***” . / persistance Create Adresse . j s ”***></s cript>**
7. **<s cript src=***” . / persistance Update Adresse . j s ”***></s cript>**
8. **<s cript src=***” . / persistance Delete Adresse . j s ”***></s cript>**
9. **<***!−− Code HTML de l a vue −−* Str u c tu r e g éné r a l e de l a **page HTML** *−−***>**

83

1. **<button id=***” bouton Ajouter Personne”***>**Ajouter une personne**</button><br** /**>**
2. **<span id=***” l i s t e Personnes ”* **class=** *” panel ”***></span>**
3. **<span class=** *” panel ”***>**
4. **<span id=***” vue Detail ”***>**

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

**</span><br** /**><br** /**>**

**</span>**

**<span id=***” spanMainForm”* **class=** *” panel ”***>**

**<form id=***” ajouter PersonneForm”* **method=***” post ”* **></form>**

**<form id=***” modifierPersonneForm”* **method=***” post ”* **></form>**

**<form id=***” ajouter Adresse Form ”* **method=***” post ”* **></form>**

**<form id=***” modifier AdresseForm ”* **method=***” post ”* **></form>**

</span>

**<**! Ajout d ’ un main et ex é cution **>**

−− −−

<s cript>

/\*\*

* *Sé r i e d ’ i n s t r u c t i on s e f f e c t u é es pour i n i t i a l i s e r l ’ ap p l i cat i on /*
* *@method mainFunction*
* *@augments myApp*

\*/

myApp. addModule ( *” mainFunction”* , **function** () {

// Cré ation d ’ un modèle avec une c o l l e c t i o n de Personne v ide

myApp. addModule . apply (myApp, [ *” modele”* , { s e l e c te d Pe r s o n n e : **null** ,

personnes : [ ] ,

} ] ) ;

// Charger l e modèle :

myApp. gui . mediator . p u b l i s h ( *” personne/ read ”* , {

personne : myApp. modele . s e l e c te d Pe r s o n n e

}) ;

}) ;

/////////////////////////////////////////////////////

// Exé cution de l a mé thode mainFunction

myApp. mainFunction ( ) ;

**</s cript>**

**</body>**

**</html>**

### Création, Mise à jour, et suppression des personnes

Code Source 6.8 : /clientAndAPI/client/persistanceCreatePersonne.js

1 */\*\**

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de cr é ation d ’ une personne*
  2. *\* sur l e serveur par requ ê t e AJAX.*

4 *\*/*

5 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Create Personne Query Server ”* ,

**function** ( ) {

6 */\*\**

1. *\* Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
2. *\* Cette mé thode informe simplement l ’ u t i l i s a t e u r des é v en t u e l l e s erreurs .*
3. *\* En e f f e t , l a requ ê t e n ’ es t pas suppos ée retourner des donné es .*

10 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {
2. **var** concatError Msg=*””* ;

15

16

17

18 }

**i f** ( r e tr i e v e d D a ta [ *’ error ’* ] . hasOwnProperty ( key ) ) {

concatError Msg += key + *” : ”* + r e tr i e v e d D a ta [ *’ error ’* ] [ key ] + *”\n”* ;

}

19

20 }

21 } ;

22

a l e r t ( concatError Msg ) ;

23 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

27 *\*/*

28 **var** create Personne = **function** ( contextArg ) { 29

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({
3. dataType : *” json ”* , *// On envoie l e s donné es l a personne cod ée en JSON*
4. *// L ’URI complète i n c l u s l ’ ID de l a ressource à cr é er*
5. u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / personne/”*

35 + contextArg . personne . g e t A ttr i b u te ( *” id ”* ) ,

1. method : *’ post ’* , *// Verbe HTTP*
2. contentType : *’ ap p l i cat i on /x www form urlencoded ’* ,

− − −

1. *// donné es à t ransmettre au serveur*
2. data : {
3. personne : { *// A t t r i b u t s de l a personne*
4. nom : contextArg . personne . g e t A ttr i b u te ( *”nom”* ) 42 }

43 } ,

1. *// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès*
2. s u c c e s s : ajax Callback Success ,
3. *// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*
4. status Code : myApp. p e r s i s t a n c e . status Code Object 48 }) ;

49 } ;

1. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ created ”* , create Personne ) ; 52 }() ] ) ;

1 */\*\**

##### Code Source 6.9 : /clientAndAPI/client/persistanceUpdatePersonne.js

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de modification d ’ une personne*
  2. *\* sur l e serveur par requ ê t e AJAX.*

4 *\*/*

5 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Update Personne Query Server ”* ,

**function** ( ) {

6 */\*\**

1. *\* Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
2. *\* Cette mé thode informe simplement l ’ u t i l i s a t e u r des é v en t u e l l e s erreurs .*
3. *\* En e f f e t , l a requ ê t e n ’ es t pas suppos ée retourner des donné es .*

10 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {
2. **var** concatError Msg=*””* ;

15

16

17

18 }

**i f** ( r e tr i e v e d D a ta [ *’ error ’* ] . hasOwnProperty ( key ) ) {

concatError Msg += key + *” : ”* + r e tr i e v e d D a ta [ *’ error ’* ] [ key ] + *”\n”* ;

}

19

20 }

21 } ;

22

a l e r t ( concatError Msg ) ;

23 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

27 *\*/*

28 **var** update Personne = **function** ( contextArg ) { 29

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({
3. dataType : *” json ”* , *// On envoie l e s donné es l a personne cod ée en JSON*
4. *// L ’URI complète i n c l u s l ’ ID de l a ressource à cr é er*
5. u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / personne/”*

35 + contextArg . personne . g e t A ttr i b u te ( *” id ”* ) ,

1. method : *’ put ’* , *// Verbe HTTP*
2. contentType : *’ ap p l i cat i on /x www form urlencoded ’* ,

− − −

1. *// donné es à t ransmettre au serveur*
2. data : {
3. personne : { *// A t t r i b u t s de l a personne*
4. nom : contextArg . personne . g e t A ttr i b u te ( *”nom”* ) 42 }

43 } ,

1. *// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès*
2. s u c c e s s : ajax Callback Success ,
3. *// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*
4. status Code : myApp. p e r s i s t a n c e . status Code Object 48 }) ;

49 } ;

1. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” personne/ changed”* , update Personne ) ; 52 }() ] ) ;

1 */\*\**

##### Code Source 6.10 : /clientAndAPI/client/persistanceDeletePersonne.js

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de suppression d ’ une personne*
  2. *\* sur l e serveur par requ ê t e AJAX.*

4 *\*/*

5 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Delete Personne Query Server ”* ,

**function** ( ) {

6 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

10 *\*/*

11 **var** d e l e te Pe r s o n n e = **function** ( contextArg ) { 12

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({

15

method : *’ d e l e t e ’* ,

dataType : *” json ”* , *// On envoie l e s donné es l a personne cod ée en JSON*

*// L ’URI complète i n c l u s l ’ ID de l a ressource à supprimer*

u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / personne/”*

+ contextArg . personne . g e t A ttr i b u te ( *” id ”* ) ,

contentType : *’ ap p l i cat i on /x−www−form−u r l encoded ’* ,

*// Verbe HTTP*

data : {

*// Objet v ide ( l e paramètre ID es t sp é c i f i é dans l ’URI)*

} ,

*// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*

status Code : myApp. p e r s i s t a n c e . status Code Object

}) ;

*// donné es à t ransmettre au serveur*

} ;

*// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*

myApp. gui . mediator . s u b s c r i b e ( *” personne/ d e l e t e ”* , d e l e te Pe r s o n n e ) ;

}() ] ) ;

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

### Création, Mise à jour, et suppression des adresses

Code Source 6.11 : /clientAndAPI/client/persistanceCreateAdresse.js

1 */\*\**

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de cr é ation d ’ une personne*
  2. *\* sur l e serveur par requ ê t e AJAX.*

4 *\*/*

5 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Create Adresse Query Server ”* ,

**function** ( ) {

6 */\*\**

1. *\* Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
2. *\* Cette mé thode informe simplement l ’ u t i l i s a t e u r des é v en t u e l l e s erreurs .*
3. *\* En e f f e t , l a requ ê t e n ’ es t pas suppos ée retourner des donné es .*

10 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {
2. **var** concatError Msg=*””* ;
3. **i f** ( r e tr i e v e d D a ta [ *” error ”* ] !== **null** ) {
4. **for** ( **var** key **in** r e tr i e v e d D a ta [ *’ error ’* ] ) {
5. **i f** ( r e tr i e v e d D a ta [ *’ error ’* ] . hasOwnProperty ( key ) ) {
6. concatError Msg += key + *” : ”* + r e tr i e v e d D a ta [ *’ error ’* ] [ key ] + *”\n”* ; 17 }

18 }

19 a l e r t ( concatError Msg ) ; 20 }

21 } ;

22

23 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

27 *\*/*

28 **var** c r e a te A d r e s s e = **function** ( contextArg ) { 29

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55 } ;

dataType : ” json ” , // On envoie l e s donné es l a personne cod ée en JSON

// L ’URI complète i n c l u s l ’ ID de l a ressource à cr é er

u r l : ” h t t p :// progjs / exemples / ap i Rest f u l / adresse /”

+ contextArg . a d r e s s e . g e t A ttr i b u te ( *” id ”* ) , method : *’ post ’* , *// Verbe HTTP*

contentType : *’ ap p l i cat i on /x www form urlencoded ’* ,

− − −

// donné es à t ransmettre au serveur

data : {

a d r e s s e : { *// A t t r i b u t s de l ’ adresse*

id Personne : contextArg . personne . g e t A ttr i b u te ( *” id ”* ) , numeroRue : contextArg . a d r e s s e . g e t A ttr i b u te ( *” numeroRue”* ) , rue : contextArg . a d r e s s e . g e t A ttr i b u te ( *” rue”* ) ,

complementAddr : contextArg . a d r e s s e . g e t A ttr i b u te ( *” complementAddr”* ) , code Postal : contextArg . a d r e s s e . g e t A ttr i b u te ( *” code Postal ”* ) ,

v i l l e : contextArg . a d r e s s e . g e t A ttr i b u te ( *” v i l l e ”* ) , pays : contextArg . a d r e s s e . g e t A ttr i b u te ( *” pays ”* )

}

} ,

// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès

s u c c e s s : ajax Callback Success ,

// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur

status Code : myApp. p e r s i s t a n c e . status Code Object

}) ;

1. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / created ”* , c r e a te A d r e s s e ) ; 58 }() ] ) ;

1 */\*\**

##### Code Source 6.12 : /clientAndAPI/client/persistanceUpdateAdresse.js

* 1. *\* Dé f i n i t i o n et enregistrement des c a l l b ac k s de cr é ation d ’ une personne*
  2. *\* sur l e serveur par requ ê t e AJAX.*

4 *\*/*

5 myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Update Adresse Query Server ”* ,

**function** ( ) {

6 */\*\**

1. *\* Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
2. *\* Cette mé thode informe simplement l ’ u t i l i s a t e u r des é v en t u e l l e s erreurs .*
3. *\* En e f f e t , l a requ ê t e n ’ es t pas suppos ée retourner des donné es .*

10 *\*/*

1. **var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {
2. **var** concatError Msg=*””* ;
3. **i f** ( r e tr i e v e d D a ta [ *” error ”* ] !== **null** ) {
4. **for** ( **var** key **in** r e tr i e v e d D a ta [ *’ error ’* ] ) {
5. **i f** ( r e tr i e v e d D a ta [ *’ error ’* ] . hasOwnProperty ( key ) ) {
6. concatError Msg += key + *” : ”* + r e tr i e v e d D a ta [ *’ error ’* ] [ key ] + *”\n”* ; 17 }

18 }

19 a l e r t ( concatError Msg ) ; 20 }

21 } ;

22

23 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

27 *\*/*

28 **var** update Adresse = **function** ( contextArg ) { 29

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({
3. dataType : *” json ”* , *// On envoie l e s donné es l a personne cod ée en JSON*
4. *// L ’URI complète i n c l u s l ’ ID de l a ressource à mettre à jour*
5. u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / adresse /”*

35 + contextArg . a d r e s s e . g e t A ttr i b u te ( *” id ”* ) ,

1. method : *’ put ’* , *// Verbe HTTP*
2. contentType : *’ ap p l i cat i on /x www form urlencoded ’* ,

− − −

1. *// donné es à t ransmettre au serveur*
2. data : {
3. a d r e s s e : { *// A t t r i b u t s de l ’ adresse*
4. id Personne : contextArg . personne . g e t A ttr i b u te ( *” id ”* ) ,
5. numeroRue : contextArg . a d r e s s e . g e t A ttr i b u te ( *” numeroRue”* ) ,
6. rue : contextArg . a d r e s s e . g e t A ttr i b u te ( *” rue”* ) ,
7. complementAddr : contextArg . a d r e s s e . g e t A ttr i b u te ( *” complementAddr”* ) ,
8. code Postal : contextArg . a d r e s s e . g e t A ttr i b u te ( *” code Postal ”* ) ,
9. v i l l e : contextArg . a d r e s s e . g e t A ttr i b u te ( *” v i l l e ”* ) ,
10. pays : contextArg . a d r e s s e . g e t A ttr i b u te ( *” pays ”* ) 48 }

49 } ,

1. *// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès*
2. s u c c e s s : ajax Callback Success ,
3. *// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*
4. status Code : myApp. p e r s i s t a n c e . status Code Object 54 }) ;

55 } ;

1. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / changed”* , update Adresse ) ; 58 }() ] ) ;

##### Code Source 6.13 : /clientAndAPI/client/persistanceDeleteAdresse.js

1

*/\*\**

* *Dé f i n i t i o n et enregistrement des c a l l b ac k s de cr é ation d ’ une personne*
* *sur l e serveur par requ ê t e AJAX.*

*\*/*

myApp. addModule . apply (myApp. p e r s i s ta n c e , [ *” callbacks Delete Adresse Query Server ”* ,

**function** ( ) {

*/\*\**

* + *Mé thode cal l b ack qui es t appel ée en cas de succès de l a requ ê t e AJAX.*
  + *Cette mé thode informe simplement l ’ u t i l i s a t e u r des é v en t u e l l e s erreurs .*
  + *En e f f e t , l a requ ê t e n ’ es t pas suppos ée retourner des donné es .*

*\*/*

**var** a ja x C a l l b a c k Su c c e s s = **function** ( r e tr i e v e d D a ta ) {

**var** concatError Msg=*””* ;

**i f** ( r e tr i e v e d D a ta [ *” error ”* ] !== **null** ) {

**for** ( **var** key **in** r e tr i e v e d D a ta [ *’ error ’* ] ) {

**i f** ( r e tr i e v e d D a ta [ *’ error ’* ] . hasOwnProperty ( key ) ) {

concatError Msg += key + *” : ”* + r e tr i e v e d D a ta [ *’ error ’* ] [ key ] + *”\n”* ;

}

}

a l e r t ( concatError Msg ) ;

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20 }

21 } ;

22

23 */\*\**

1. *\* Callback appel é l or s de l ’ événement ” personne/ read ” du mé d i at or .*
2. *\* Effectue une requ ê t e AJAX pour r écupé rer t ou t es l e s personnes*
3. *\* pour recon s t ru i re l e modèle de donné es .*

27 *\*/*

28 **var** d e l e te A d r e s s e = **function** ( contextArg ) { 29

1. *// requ ê t e AJAX get cod é en JSON*
2. **var** jqxhr = $ . ajax ({
3. dataType : *” json ”* , *// On envoie l e s donné es l a personne cod ée en JSON*
4. *// L ’URI complète i n c l u s l ’ ID de l a ressource à supprimer*
5. u r l : *” h t t p :// progjs / exemples / ap i Rest f u l / adresse /”*

35 + contextArg . a d r e s s e . g e t A ttr i b u te ( *” id ”* ) ,

1. method : *’ d e l e t e ’* , *// Verbe HTTP*
2. contentType : *’ ap p l i cat i on /x www form urlencoded ’* ,

− − −

1. *// donné es à t ransmettre au serveur*
2. data : {
3. *// Objet v ide ( l e paramètre ID es t sp é c i f i é dans l ’URI)*

41 } ,

1. *// Mé thode cal l b ack qui r econ s t ru i t l e modèle en cas de succès*
2. s u c c e s s : ajax Callback Success ,
3. *// Objet dé f i n i s s a n t l e s c a l l b ac k s d ’ erreurs avec codes du serveur*
4. status Code : myApp. p e r s i s t a n c e . status Code Object 46 }) ;

47

48 } ;

1. *// Enregistrement du cal l b ack de l ’ événement de mise à jour de l a personne*
2. myApp. gui . mediator . s u b s c r i b e ( *” adresse / d e l e t e ”* , d e l e te A d r e s s e ) ; 51 }() ] ) ;

# Annexe A

**Graphisme avec les Canvas *HTML5***

* 1. **Notion de *canvas***

##### Les *canvas HTML5* fournissent une petite *API* graphique 2D en *javascript* qui permet de réaliser des dessins, des graphiques, etc. sans plugin. Les canvas 2D sont dores et déjà disponible sur tous les grands navigateurs. L’extension *webGL* (qui dépasse le cadre de ce cours) permet de faire des affichage de scènes 3D en accédant aux fonctionnalités d’*OpenGL* via les shaders en *GLSL*. L’extension *webGL* est implémentée dans tous les Grands Navigateurs mais n’est pas implémentée à ce jour dans *internet explorer* car l’éditeur de ce navigateur préfère privilégier une solution propriétaire.

Voici un exemple avec un canvas qui dessine un triangle.

Code Source A.1 : Génération et sortie du code *JSON* en *PHP*

* 1. <!doctype HTML>
  2. **<html lang=***” f r ”***>**
  3. <head>
  4. **<meta charset=***”UTF 8”* /**>**

−

* 1. **<t i t l e>**Mon premier canvas HTML5**</ t i t l e>**
  2. </head>
  3. **<body>**
  4. **<***! Dé c l arat i on d ’ un canvas v ide avec son id* **>**

−− −−

* 1. **<canvas id=***” monCanvas”* **width=***” 2000 ”* **height=***” 1000 ”* **style=** *”* **position** *: ab sol u t e ;”*

></canvas>

* 1. **<s cript>**
  2. *// On r é cupère l e canvas pour dessiner*
  3. **var** myCanvas **=** document . getElementBy Id ( *” monCanvas”* ) ;
  4. *// On r é cupère un contexte du canvas pour u t i l i s e r l e s mé thodes de dessin*
  5. **var** context **=** myCanvas . get Context ( *” 2d”* ) ;
  6. *// couleur de remplissage rouge*
  7. context . f i l l S t y l e **=** *”#FF0000”* ; 17

1. context . begin Path () ;
2. context . moveTo ( 10 , 10 ) ;
3. context . l ine To ( 100 , 100 ) ;
4. context . l ine To ( 190 , 10 ) ;
5. context . l ine To ( 10 , 10 ) ; 23
6. context . f i l l () ;
7. context . c l ose Path () ;

26

**</s cript>**

**<h1>**Page **HTML** avec un canvas**</h1>**

**<p>**

**</p>**

**</body>**

**</html>**

27

28

29

30

31

## Exemple d’animation dans un *canvas*

Voici un exemple qui réalise une animation à l’aide d’un timer qui exécute la fonction animate

toutes les 20*ms*, soit 50 fois par seconde.

Code Source A.2 : Génération et sortie du code *JSON* en *PHP*

* 1. **<!doctype HTML>**
  2. **<html lang=***” f r ”***>**
  3. <head>
  4. **<meta charset=***”UTF 8”* /**>**

−

* 1. **<t i t l e>**Mon premier canvas HTML5**</ t i t l e>**
  2. </head>
  3. **<body>**
  4. **<***! Dé c l arat i on d ’ un canvas v ide avec son id* **>**

−− −−

* 1. **<canvas id=***” monCanvas”* **width=***” 2000 ”* **height=***” 1000 ”* **style=** *”* **position** *: ab sol u t e ;”*

></canvas>

* 1. **<s cript>**
  2. **var** timer **=** s e t I n t e r v a l ( animate , 20 ) ; 12

13 **function** animate () { 14

1. *// On r é cupère l e canvas pour dessiner*
2. **var** canvas **=** document . getElementBy Id ( *” monCanvas”* ) ;
3. *// On r é cupère un contexte du canvas pour u t i l i s e r l e s mé thodes de dessin*
4. **var** context **=** canvas . get Context ( *” 2d”* ) ;
5. *// couleur de remplissage rouge*
6. context . f i l l S t y l e **=** *”#FF0000 ”* ;
7. context . begin Path ( ) ;
8. **var** d **= new** Date ( ) ;
9. **var** n **=** d . getTime () ;
10. *// nombre de mil l i s econ d es depuis l e 01 /01/1970*

25

26 **var** s e c **=** n / 1000 . 0 ;

27 context . **clear** Rect ( 0 , 0 , canvas . **width** , canvas . **height** ) ; 28

29 context . save ( ) ;

30 context . t r a n s l a t e (200+500 \* (1+Math . cos ( 0 . 5 \* s e c ) ) , 200+200 \*(1 .0 + Math . s i n ( s e c ) ) ) ;

1. *// l ’ angle de rot at i on d o i t ê t re entre 0 et 2\*Math . PI ’*
2. context . r o ta te ( s e c 2\*Math . PI\*Math . round ( sec/ ( 2 \* Math . PI ) ) ) ;

−

1. context . moveTo ( 0 , 0 ) ;
2. context . l ine To ( 100 , 100 ) ;
3. context . l ine To ( 200 , 0 ) ;
4. context . l ine To ( 0 , 0 ) ; 37

38 context . f i l l ( ) ;

39

**</s cript>**

**<h1>**Page **HTML** avec un canvas**</h1>**

**<p>**

**</p>**

**</body>**

**</html>**

}

context . c l ose Path ( ) ; context . r e s t o r e ( ) ;

40

41

42

43

44

45

46

47

48

**Annexe B**

**Programmation Événementielle en**

***JavaScript***

## Rappel sur la Gestion d’Événements en *CSS*

Dans un style *CSS*, on peut mettre des styles différents sur une balise *HTML* donnée, suivant le contexte utilisateur, via la notion d’événement. Dans l’exmple suivant, le style d’un lien est modifié suivant que le lien a déja été cliqué, ou si la souris survolle le lien (événement *hover*).

Code Source B.1 : Génération et sortie du code *JSON* en *PHP*

*/\* s t y l e par dé f aut des l i en s \*/*

a : l ink {

**text***−***decoration** : none ;

}

**color** : #00e ; */\* b l eu c l a i r \*/*

*/\* s t y l e des l i en s v i s i t é s \*/*

a : v i s i te d {

**text***−***decoration** : none ;

}

**color** : #c 0 c ; */\* mauve \*/*

*/\* s t y l e des l i en s v i s i t é s \*/*

a :hover {

**text***−***decoration** : u n d e r l i n e ; */\* sou l i g n é \*/*

}

**color** : #e40 ; */\* rouge v i f \*/*

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Voici un autre exemple, dans lequel un élément *HTML* (ici une balise <span> et son contenu) apparaît en *popup* pour afficher les détails d’une personne lors du survol de nom de la personne. La balise *span* (au sein d’un paragraphe d’une classe *CSS* spécifique appelé popupDetails) est par défaut invisible (propriété display à none). Cette même balise *span* devient visible

lorsque le paragraphe est survollé.

Code Source B.2 : Génération et sortie du code *JSON* en *PHP*

< ! doctype html>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*” utf −8”*/>

<l i n k r e l=*” s t y l e s h e e t ”* h r e f=*” ./ my Style . css ”*/>

<s ty l e >

1

2

3

4

5

6

1. body{
2. font f a m i l y : *” Comic Sans MS”* ;

−

1. font s i z e : 120% ; 10

} *−*

11 h1 {

1. margin : 0 auto ;
2. text a l i g n : c e n te r ; 14

} *−*

1. p . popup Details {
2. background c o l o r : yellow ;

−

1. p o s i t i o n : r e l a t i v e ; */\* pour p os i t i on er l e span en absolu \*/*
2. max width : 200 px ; 19

} *−*

1. p . popup Details span {
2. d i s p l a y : none ; 22 }
3. p . popup Details :hover span {
4. p o s i t i o n : a b s o l u te ;
5. l e f t : 200 px ;

26 top : 30 ;

−

1. min width : 500 px ;

−

1. background c o l o r : black ;

−

1. c o l o r : white ;
2. border r a d i u s : 20 px ;

−

1. padding : 10 px ;
2. d i s p l a y : block ; 33 }
3. </ s ty l e >
4. < t i t l e >Popups en HTML et CSS</ t i t l e >
5. </head>
6. <body>
7. <! dé but du corps HTML >

−− −−

1. <h1><i >Popup</i > en <i >HTML</i > et <i >CSS</i ></h1>
2. <p **class**=*” popup Details ”*>
3. S c a r l e t t Johansson
4. <span>né e l e 22 novembre 1984 à New York ,
5. e s t une a c t r i c e et chante **use** amé r i c a i n e .< br/>
6. ( sour ce&nbsp ; : wikip é dia )
7. </span>

46 </p>

1. </body>
2. <! f i n du corps HTML >

−− −−

1. </html>
2. <!*−−* f i n du code HTML *−−*>
   1. **Événements en *Javascript***
      1. **Le principe des événements en *Javascript***

##### Les événements en *Javascript* permettent, en réponse à une événement sur un élément *HTML* du document, d’appeler une fonction *callback* en *Javascript*. Ceci suffit à créer une interface homme machine (*IHM* ) côté client, basée sur de la programmation événementielle en *Javascript*.

Une liste (non exhaustive ; Voir sur le *web* pour la liste complète)

* + - 1. Événements souris
         1. onclick : sur un simple clic
         2. ondblclick : sur un double clic
         3. onmousedown : lorsque le bouton de la souris est enfoncé, sans forcément le relâcher
         4. onmousemove : lorsque la souris est déplacée
         5. onmouseout : lorsque la souris sort de l’élément
         6. onmouseover : lorsque la souris est sur l’élément
         7. onmouseup : lorsque le bouton de la souris est relâché
      2. Événements clavier
         1. onkeydown : lorsqu’une touche est enfoncée
         2. onkeypress : lorsqu’une touche est pressée et relâchée
         3. onkeyup : lorsqu’une touche est relâchée
      3. Événements formulaire (a)

1. onblur : à la perte du focus
2. onchange : à la perte du focus si la valeur a changé
3. onfocus : lorsque l’élément prend le focus (ou devient actif)
4. onreset : lors de la remise à zéro du formulaire (via un bouton ”reset” ou une fonction reset())
5. onselect : quand du texte est sélectionné
6. onsubmit : quand le formulaire est validé (via un bouton de type ”submit” ou une fonction submit())

### Exemple de mise à jour d’un élément

Code Source B.3 : Génération et sortie du code *JSON* en *PHP*

< ! doctype html>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*” utf −8”*/>

<l i n k r e l=*” s t y l e s h e e t ”* h r e f=*” ./ my Style . css ”*/>

<s ty l e >

body{

font *−*f f a m i l y : *” Comic* ; *Sans MS”* ;

}

h1 {

margin : 0 auto ;

text *−*a l i g n : c e n te r ;

}

</ s ty l e >

ont *−*s i z e : 120%

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

< t i t l e >Mise à Jour Par Évé nement</ t i t l e >

</head>

<body>

<!*−−* dé but du corps HTML *−−*>

<!*−−* f i n du code HTML *−−*>

<!*−−* f i n du corps HTML *−−*>

< ml>

<h1>Mise à Jour Par Évé nement <code>onchange</code ></h1>

<p **class**=*” popup Details ”*>

<input id=*” my Input Id”* type=*” t ex t ”* s i z e=*” 15 ”*

onchange=*” fonction Mise AJour ( ’ my Input Content ’ , ’ my Input Id ’) ”*/>

<br/>

<span id=*” my Input Content”*></span>

</p>

<s c r i p t >

**function** fonction Mise AJour ( elementId , input Id ) { document . getElementBy Id ( element Id ) . innerHTML

= document . getElementBy Id ( input Id ) . value ;

}

</ s c r i p t >

</body>

/ht

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

* + 1. **Formulaires Dynamiques an *Javascript***

##### Nous voyons ici un exemple d’utilisation du *javascript* pour créer un formulaire dont les attri- buts dépendent de la valeur d’un premier champ. Lorsqu’on sélectionne “deuxième année”, un nouveau champ apparaît. Pour celà, on utilise l’évennement onchange sur l’input de l’année, qui est géré par la fonction anneeChange. On teste alors la valeur de l’attribut, puis le cas échéant on génére un nouveau champ dans un div d’id attributSupplementaire.



Code Source B.4 : Génération et sortie du code *JSON* en *PHP*

**<!doctype html>**

**<html lang=***” f r ”***>**

**<head>**

**<t i t l e>**Formulaire dynamique**</ t i t l e>**

**</head>**

**<body>**

**<form method=***” post ”* **action=***” reception . php”***>**

**<p>**

**<label for=***”nom”***>**Nom**</ label><i nput name=***”nom”* **id=***”nom”* /**>**

**<meta charset=***”UTF−8”* /**>**

1

2

3

4

5

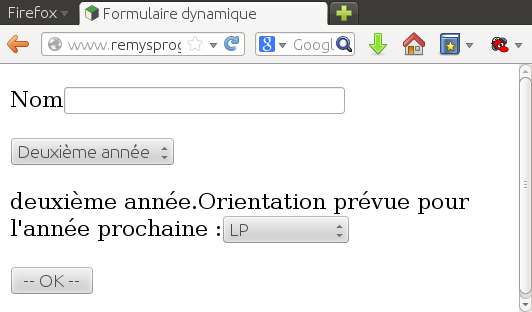
6

7

8

9

10



11 **</p>**

12 **<p>**

1. **<s elect name=***” annee”* **id=***” annee”* **pattern=***” ( premiere ) | ( deuxieme )”*

onchange= ’ anneeChange ( ) ; ’**>**

1. **<option value=***” c h o i s i s s e z ”* **selected disabled>** c h o i s i s s e z **</option>**

−− −−

1. **<option value=***” premiere”***>**Première ann é e**</option>**
2. **<option value=***” deuxième”***>**Deuxième ann é e**</option>**
3. **</s elect>**

18 **</p>**

19 **<div id=***” at t r i but Supplementaire ”***>**

20

21 **</div>**

22 **<p>**

23 **<i nput type=***” submit ”* **value=***” OK ”* /**>**

−− −−

24 **</p>**

1. </form>
2. **<s cript>**
3. **function** anneeChange ( ) {
4. **var** paragraphe **=** document . getElementBy Id ( *” at t r i but Supplementaire ”* ) ;
5. paragraphe . innerHTML=document . getElementBy Id ( *” annee”* ) . **value**+*” année .***<br** */***>***”* ;
6. **i f** ( document . getElementBy Id ( *” annee”* ) . **value** == *” deuxième”* ) {
7. paragraphe . innerHTML+**=***”***<l abel>***Orientation pr évue pour l ’ année prochaine*

*:*</ label>*”*

32 +’**<s elect name=***” orien t at i on ”* **id=***” orien t at i on ”***>**’

33 +’**<option value=***”LP”***>**LP**</option>**’

34 +’**<option value=***” master”***>**master**</option>**’

35 +*”***<option** *value=\” inge \”***>***Ecole d ’ ing é***</option>***”*

36 +’**<option value=***” boulot ”***>**Boulot**</option>**’

37 +’**<option value=***” autre ”***>**Autre**</option>**’ 38 +’**</s elect>**’ ;

39

40 }

41 }

1. anneeChange ( ) ;
2. **</s cript>**
3. **</body>**
4. **</html>**

Code Source B.5 : Génération et sortie du code *JSON* en *PHP*

< ! doctype html>

<html lang=*” f r ”*>

<head>

1

2

3

4

< t i t l e >Formulaire dynamique</ t i t l e >

</head>

<body>

<?php

$nom= ( **i s s e t** ($\_POST[ *”nom”* ] ) ) ? $\_POST[ *”nom”* ] : *”nom ind é termin é”* ;

$ annee = ( **i s s e t** ($\_POST[ *” annee”* ] ) ) ? $\_POST[ *” annee”* ] : *” année ind é temin ée”* ;

**echo** *”Nom : ”* . $nom . *”<br/>”* ;

**echo** *”Année : ”* . $ annee . *”<br/>”* ;

**i f** ( $ annee==*” deuxième”* )

**echo** *” Orientation : ”* .$\_POST[ *” orien t at i on ”* ] ;

<meta c h a r s e t=*”UTF−8”*/>

?>

</body>

</html>

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

**Annexe C**

**Gestion des fenêtres**

## Charger un nouveau document

Code Source C.1 : Génération et sortie du code *JSON* en *PHP*

*// constructeur*

**function** Telephone ( t e l 1 ) {

*// t e s t de t é l éphone fran ¸ ais à 10 c h i f f r e s*

*// 1 ) supprimer l e s espaces , 2 ) t e s t e r l e s c h i f f r e s*

**else**

**i f** ( t e l 1 . r e p l a c e (/\ s /g , *’ ’* ) . match ( */^((\+33) | 0 ) [ 0 −9]{9}$/*g ) )

**t**

**his** . t e l 1=t e l 1 ;

}

**throw new** Error ( *”Numéro de t é l éphone i n v a l i d e ”* ) ;

Telephone . prototype . a f f i c h e = **function** ( ) {

document . w r i te ( *”Té l éphone 1 : ”*+**this** . t e l 1+*”<br/>”* ) ;

}

1

2

3

4

5

6

7

8

9

10

11

12

13

Code Source C.2 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Charger un document</ t i t l e >

< s c r i p t s r c=*” ./ ex01\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

<p>

<s c r i p t >

**try** {

**var** numero = prompt ( *” Merci d ’ entrer un numéro de t é l éphone en France mé t r o p o l i t a i n e ”* ) ;

**var** t e l = **new** Telephone ( numero ) ; t e l . a f f i c h e ( ) ;

} **catch** ( e r r ) {

l o c a t i o n = *” ex 01 \_error . html ”* ;

}

</ s c r i p t >

<p>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

</body>

</html>

21

Code Source C.3 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Charger un document</ t i t l e >

< s c r i p t s r c=*” ./ ex10\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

<p>

Bonjour , I l s *’ es t produit une erreur . Merci d ’* e n tr e r un numé ro v a l i d e .

Si l e problème p e r s i s t e , merci de c o n ta c te r l e s t a g i a i r e qui a f a i t l e s i t e . . .

<button o n c l i c k=*” l ocat i on = ’ ex01\_loadNewDoc . html ’ ; ”*>Retour à l a s a i s i e </button>

<p>

</body>

</html>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

## Naviguer dans l’historique

la propriété history a deux méthodes back() et forward() qui permettent respectivement de reculer ou d’avancer dans l’historique.

Code Source C.4 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Charger un document</ t i t l e >

< s c r i p t s r c=*” ./ ex10\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

<p>

Bonjour , bla , bla . . . < br/>

<a h r e f = *” ex 02 \_ history Back . html ”*>Cliquez i c i </a> pour a l l e r à l a page s u i v a n te .

<p>

</body>

</html>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

Code Source C.5 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Charger un document</ t i t l e >

< s c r i p t s r c=*” ./ ex10\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

<p>

1

2

3

4

5

6

7

8

9

10

Bla , bla . . . < br/>

Vous avez r a t é quelque chose ?

<button o n c l i c k=*” h i s t ory . back () ;”*>Retour à l a page pr é c é dente </button>

<p>

</body>

</html>

11

12

13

14

15

## Ouvrir une nouvelle fenêtre (popup)

Code Source C.6 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Ouvrir un f en ê tre </ t i t l e >

< s c r i p t s r c=*” ./ ex10\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

<p>

Bonjour , bla , bla . . .

<button o n c l i c k=*” window . open ( ’ ex03\_windowPopup . html ’ , ’ma popup ’ , ’ width =400 , heig h t =400 , r e s i z ea b l e=yes ’) ;”*>

Plus d *’ i n f os*

*</button>*

*<p>*

*</body>*

*</html>*

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

Code Source C.7 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Charger un document</ t i t l e >

< s c r i p t s r c=*” ./ ex10\_* **class** *es\_ telephone . j s ”*></s c r i p t >

</head>

<body>

Coucou !

<p>

<p>

<a h r e f=*” j av as c r i p t :window . c l os e () ;”*>Fermer l a f en ê tre </a>

</p>

</body>

</html>

<p s t y l e=*” font−s i z e : 100 ; t ext −al i g n : center ;”*>

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

**Annexe D**

***Document Object Model (DOM)***

La programmation côté client permet de modifier certaines parties d’un document *HTML* dans recharger toute la page. Il y a plusieurs avantages : on évite de surcharger le serveur et le trafic réseau et on améliore la réactivité de l’application *web* pour le plus grand bonheur de l’utilisateur.

Pour faire cela, le langage *Javascript* côté client fournit une structure de données permettant d’accéder aux éléments du document *HTML* et de modifier les éléments du document *HTML*. Cette structure de données s’appelle le *Document Object Model*, en abrégé *DOM*. Il existe un *DOM* legacy qui s’est sédimenté informellement au travers des versions successives du *javascript* en tenant compte des implémentations des différents navigateurs, qui collaboraient plus ou moins bien pour être mutuellement compatibles. Il existe aussi le *DOM* tel qu’il a été finalement spécifié par le *W3C*.

Les éléments du document *HTML* ayant, de par leur imbrication, une structure arbores- cente, le *DOM W3C* a une structure d’arbre. On peut accèder et manipuler via un ensemble de propriétés et de méthodes *javascript*, notamment de l’interface Document et de l’interface Element et ses classes filles, qui permettent de manipulet les éléments (*HTML* entre autres) du document.

## Qu’est-ce que le *DOM* ?

Le *Document Object Model* (en abrégé *DOM* ) corrrespond à l’arborescence des imbrications des balises *HTML* d’un document. Voici un fichier *HTML* simple et une représentation schématique du *DOM* correspondant.

Code Source D.1 : Génération et sortie du code *JSON* en *PHP*

< ! doctype HTML>

<html lang=*” f r ”*>

<head>

<meta c h a r s e t=*”UTF−8”* />

< t i t l e >Un document HTML5</ t i t l e >

</head>

<body>

<h1>Exemple de f i c h i e r <i >HTML5</i ></h1>

<p>

Un document <i >HTML5</i > e s t une s t r u c tu r e a r b o r e s c e n te .

<p>

</body>

1

2

3

4

5

6

7

8

9

10

11

12

13

</html>

Document

<html>

<head>

<body>

<title>

<meta>

<h1>

<p>

"Un document HTML5"

"Exemple de fichier"

<i>

"Un document"

<i>

"est une structure arborescente."

"HTML5"

"HTML5"

Le *DOM* dont nous parlons ici est le *DOM* du *W3C*, qui est aujourd’hui supporté par tous les grands navigateurs.

Le langage *Javascript* côté client propose une hiérarchie de classes pour parcourir et mani- puler le *DOM* d’un document. Il s’agit essentiellement d’une structure de donnée d’arbre, où chaque noeud (correspondant à une balise ou commentaire ou texte, etc. du document) possède une collection de noeuds fils, qui sont les éléments ou structures imbriquées.

La bibliothèque *jQuery* permet un accès plus haut niveau au *DOM* pour sonder et manipuler le code du document.

## Sélection et Manipulation de Base sur le *DOM*

### Sélection de tout ou partie des éléments

L’exemple suivant cherche tous les éléments du document et affiche leur nom de balise (tagName ou nodeName). On apprend aussi à ajouter du code *HTML* à l’interieur d’un élément (au début ou à la fin).

L’exemple suivant montre commen sélectionner certains éléments du document, par nom de balise, classe *CSS*, etc. On apprend aussi à modifier des propriétés *CSS* des éléments.

Code Source D.2 : Génération et sortie du code *JSON* en *PHP*

**<!doctype html>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***” utf−8 ”***>**

**<t i t l e>**Mo d i f i e r l e **style** de c e r t a i n s

**<s cript src=***” . /jquery−1 . 1 0 . 2 . j s ”***></s cript>**

é l é ments**</ t i t l e>**

**<s tyle>**

**p** . myClass {

**background***−***color** : #ddd ;

}

**padding** : 10 px ;

1

2

3

4

5

6

7

8

9

10

11



12

**</s tyle>**

**</head>**

**<body>**

**<h1>**M o d i f i e r l e **style** de c e r t a i n s é l é ments**</h1>**

**<div>**

**<h2>**P a r t i e 1**</h2>**

**<p>**C eci e s t l e te x te de l a p a r t i e 1 .**</p>**

**</div>**

**<div>**

**<h2>**P a r t i e 2**</h2>**

**<p class=** *” myClass”***>**Le te x te de l a p a r t i e 2 e s t d i f f é rent .**</p>**

**</div>**

**<s cript>**

*// Récupé ration d ’ é l éments jQuery pour l e s b a l i s e s* **<p>** *et* **<h2> var** e l ements **=** $ ( *”p , h2 ”* ) ;

e l ements . c s s ( *”* **border***”* , *” 2 px s o l i d ”* ) ;

*// Modification du s t y l e du t i t r e* **<h1>**

$ ( *” h1 ”* ) . c s s ( *”* **text***−***align** *”* , *” center ”* ) ;

**</s cript>**

**</body>**

$ ( *” p . myClass”* ) . c s s ( *”* **border** *−radius”* , *” 20 px”* ) ;

*// Modification du s t y l e du ( ou des ) paragraphe ( s ) de l a c l as s e CSS myClass*

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

### Filtrage par le texte

L’exemple suivant montre comment sélectionner des éléments par mots du texte (sensible à la casse).

Code Source D.3 : Génération et sortie du code *JSON* en *PHP*

**<!doctype html>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***” utf−8 ”***>**

**<t i t l e>**F i l t r a g e du te x te**</ t i t l e>**

**<s cript src=***” . /jquery−1 . 1 0 . 2 . j s ”***></s cript>**

1

2

3

4

5

6



|  |  |
| --- | --- |
| 7 | **<s tyle>** |
| 8 | **p** { |
| 9 | **padding** : 10 px 0 ; |
| 10 | } |
| 11  12  13 | **</s tyle>**  **</head>**  **<body>** |
| 14 | **<h1>**F i l t r a g e du te x te**</h1>** |
| 15 | **<div>** |
| 16 | **<h2>**P a r t i e 1**</h2>** |
| 17 | **<p>**C eci e s t l e te x te de l a p a r t i e 1 .**</p>** |
| 18 | **</div>** |
| 19 | **<div>** |
| 20 | **<h2>**P a r t i e 2**</h2>** |
| 21 | **<p>**Le te x te de l a **<em>**p a r t i e 2**</em>** e s t d i f f é rent .**</p>** |
| 22 | **</div>** |
| 23 | **<s cript>** |
| 24 | $ ( *” p :contains ( ’ d i f f é rent ’ ) ”* ) . prepend ( ’**<strong>**Ce paragraphe c o n t i e n t l e mot |
| 25 | **</s cr** *” d i f f é rent ”***</s trong>**.**<br** /**>** ’ ) . c s s ( *”***background***−***color** *”* , *”#ddd”* ) ;  **ipt>** |
| 26 | **</body>** |

### Application de Méthode aux éléments

L’exemple suivant montre comment appliquer une fonction à chacun des éléments sélectionnés. Ici, on met le contenu des paragraphes en gras. On ajoute une information au début de chaque paragraphe de la classe myClass.

Code Source D.4 : Génération et sortie du code *JSON* en *PHP*

**<!doctype html>**

**<html lang=***” f r ”***>**

**<head>**

**<t i t l e>**Appliquer une mé thode à chaque é l ément**</ t i t l e>**

**<s cript src=***” . /jquery−1 . 1 0 . 2 . j s ”***></s cript>**

**<meta charset=***” utf−8 ”***>**

1

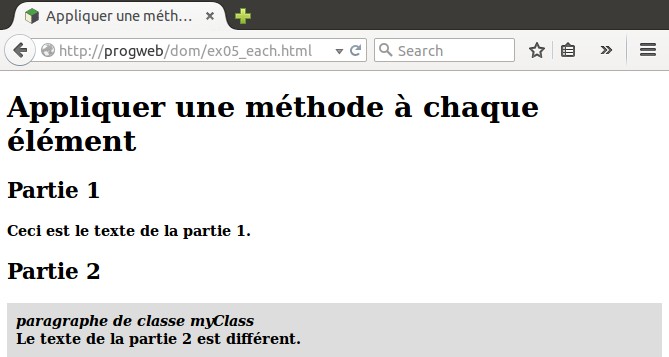
2

3

4

5

6



7

**<s tyle>**

**p** . myClass {

**background***−***color** : #ddd ;

**</s tyle>**

**</head>**

**<body>**

**<h1>**A ppliquer une mé thode à chaque é l ément**</h1>**

**<div>**

**<h2>**P a r t i e 1**</h2>**

**<p>**C eci e s t l e te x te de l a p a r t i e 1 .**</p>**

**</div>**

**<div>**

**<h2>**P a r t i e 2**</h2>**

**<p class=** *” myClass”***>**Le te x te de l a p a r t i e 2 e s t d i f f é rent .**</p>**

**</div>**

**<s cript>**

$ ( *” p”* ) . each ( **function** () {

}

**padding** : 10 px ;

$ ( **this** ) . c s s ( *”* **font***−***weight***”* , *” bolder ”* ) ;

**</s cript>**

**</body>**

**i f** ( $ ( **this** ) . has Class ( *” myClass”* ) ) {

$ ( **this** ) . prepend ( *”***<em>***p aragraphe de c l as s e myClass***</em><br** */***>***”* ) ;

}

}) ;

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

* + 1. **Événements et *Callbacks***

##### EL’exemple suivant montre comment, en réaction au click sur un bouton, transformer es pa- ragraphes en div.

Code Source D.5 : Génération et sortie du code *JSON* en *PHP*



1. <!doctype html>
2. **<html lang=***” f r ”***>**
3. <head>
4. **<meta charset=***” utf 8 ”***>**

−

1. **<t i t l e>**Év é nement de c l i c k**</ t i t l e>**
2. **<s cript src=***” . /jquery 1 . 1 0 . 2 . j s ”***></s cript>**

−

1. <s tyle>
2. **p** {
3. **background color** : #ddd ;

−

1. **padding** : 10 px ; 11 }
2. **div** . myClass {
3. **font weight** : bo l d er ;

−

1. **padding** : 10 px ;
2. **border style** : dashed ; 16

} *−*

1. **em** {
2. **font variant** : small caps ;

− −

1. **font size** : 120% ; 20

} *−*

1. button {
2. **margin** : 10 px 0 ; 23 }
3. </s tyle>
4. **</head>**

26

1. **<body>**
2. **<h1>**Év é nement de c l i c k**</h1>**
3. <div>
4. **<h2>**P a r t i e 1**</h2>**
5. **<p>**C eci e s t l e te x te de l a p a r t i e 1 .**</p>**
6. **</div>**
7. **<div>**
8. **<h2>**P a r t i e 2**</h2>**
9. **<p>**Le te x te de l a **<em>**p a r t i e 2**</em>** e s t d i f f é rent .**</p>**

36

**</div>**

**<button>**M o d i f i e r l e s paragraphes**</button>**

**<s cript>**

*// Événement de c l i c k*

$ ( *” button ”* ) . c l i c k ( **function** () {

*// Application d ’ une mé thode à chaque paragraphe*

$ ( *” p”* ) . each ( **function** () {

*// Remplacer l e* **<p>** *par un* **<div>** *en l a i s s a n t l e HTML inchang é*

$ ( **this** ) . replace With ( ’**<div class=** *” myClass”***>**’ + $ ( **this** ) . **html** ()

+ *”***</div>***”* ) ;

}) ;

}) ;

**</s cript>**

**</body>**

37

38

39

40

41

42

43

44

45

46

47

48

### Fitrage d’un Tableau

L’exemple suivant montre comment, en utilisant les utilitaires de *jQuery* permettant de traiter des Array *Javascript* génériques :

* + - 1. Filtrer le contenu d’un tableau avec une méthode de choix booléenne pour les éléments (ici, valeur multiple de 3) ;
      2. Générer le *HTML* en appliquant une méthode à chaque élément du tableau.



Code Source D.6 : Génération et sortie du code *JSON* en *PHP*

**<!doctype HTML>**

**<html lang=***” f r ”***>**

**<head>**

**<meta charset=***”UTF−8”* /**>**

**<t i t l e>**F i l t r a g e Grep sur Tableau**</ t i t l e>**

**</head>**

1

2

3

4

5

6

* 1. **<body>**
  2. **<h1>**F i l t r a g e Grep sur Tableau**</h1>**
  3. **<s cript src=***” . /jquery 1 . 1 0 . 2 . j s ”***></s cript>**

−

10

11 **<p id=***” output ”***><p>**

12

1. **<s cript>**
2. *// Cré ation d ’ un t ableau avec l e s en t i er s de 0 à 19*
3. **var** tab **= new** Array ( ) ;

16 **for** ( **var** i=0 ; **i<**20 ; **i** ++){

17 tab . push ( **i** ) ; 18 }

19

1. *// Sé l ec t i on des é l éments du t ableau par l a fonction ” mu l t i p l e de 3”*
2. **var** tab Multiple De 3 **=** $ . grep ( tab , **function** ( key , **value** ) { 22 **i f** ( key%3 == 0 )
3. return true ;
4. **else**
5. **return f al s e** ; 26 }) ;

27

1. *// Affichage du t ableau des m u l t i p l es de 3*
2. **var** outHTML **=** *””* ;
3. *// Application d ’ une fonction ( géné ration d ’HTML)*
4. *// à chaque é l ément du t ab l eau*
5. $ . each ( tab Multiple De 3 , **function** ( key , **value** ) {
6. outHTML +**=** *” tab [ ”*+key +*” ] = ”*+**value**+*”***<br** */***>***”* ; 34 }) ;
7. $ ( *”#output ”* ) . append ( outHTML ) ;
8. </s cript>
9. **</body>**
10. **</html>**