Techniques de débogage

-- Débogage avec PHP/Eclipse

Il est préférable d'utiliser un véritable débogueur, plutôt que de se satisfaire des **echo** et **var_dump** insérés dans le code pour évaluer une variable

-- XDebug

XDebug est l'un des outils de débogage existant pour PHP. Il est généralement livré par défaut avec les solutions Wampp, Xampp ou easyPhp.

-- Vérifications

Vérifier son activation en lançant **phpinfo** :

xdebug

xdebug support	enabled
Version	2.2.1
IDE Key	ECLIPSE_DBGP_192.168.1.13

Supported protocols	Revision
DBGp - Common DeBuGger Protocol	$Revision: 1.145 $

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On
xdebug.collect_params	0	0
xdebug.collect_return	Off	Off
xdebug.collect_vars	Off	Off
xdebug.coverage_enable	On	On
xdebug.default_enable	On	On

Dans certains cas, il faut l'activer en éditant **php.ini** :

```
[XDebug]
zend_extension = "C:\xampp\php\ext\php_xdebug.dll"
xdebug.profiler_append = 0
xdebug.profiler_enable = 1
xdebug.profiler_enable_trigger = 0
xdebug.profiler_output_dir = "C:\xampp\tmp"
```

```
xdebug.profiler_output_name = "cachegrind.out.%t-%s"
xdebug.remote_enable = 1
xdebug.remote_handler = "dbgp"
xdebug.remote_host = "127.0.0.1"
xdebug.remote_port = "9000"
xdebug.trace_output_dir = "C:\xampp\tmp"
xdebug.idekey = "ECLIPSE_DBGP"

xdebug.auto_trace=0       ;to always profile, set to 1 & comment out
xdebug.trace_enable_trigger
xdebug.trace_enable_trigger=1
xdebug.collect_params=4   ;displays full parameter variable names and values
xdebug.collect_return=1   ;display function return values
xdebug.trace_format=2
```

- Vérifier la présence de la dll sous Windows et son emplacement
- Noter la valeur de **xdebug.idekey** qui servira dans la communication avec Eclipse et Chrome
- Vérifier l'adresse (remote_host) et le port de communication (9000 par défaut)

-- Configuration d'Eclipse

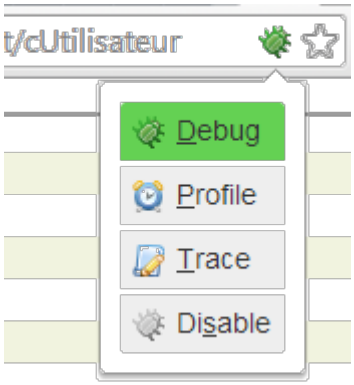
L'extension pdt doit être installée.

-- Installation du Helper de Chrome

Installer l'extension XDebug Helper de chrome : XDebug helper

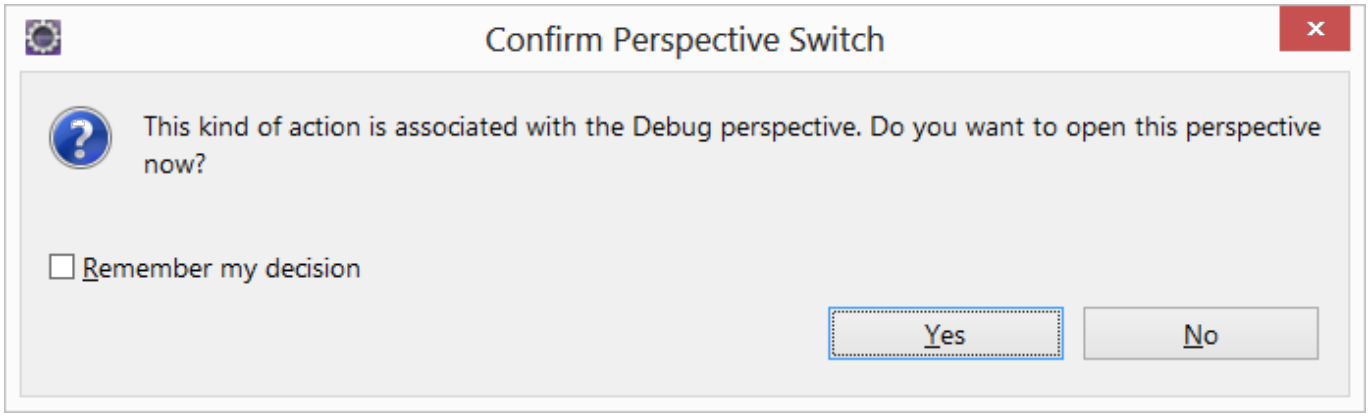
-- Débogage

- Démarrer Eclipse
- Dans chrome, aller à l'adresse de la page à déboguer et activer le débogage sur la droite de la barre d'adresse :

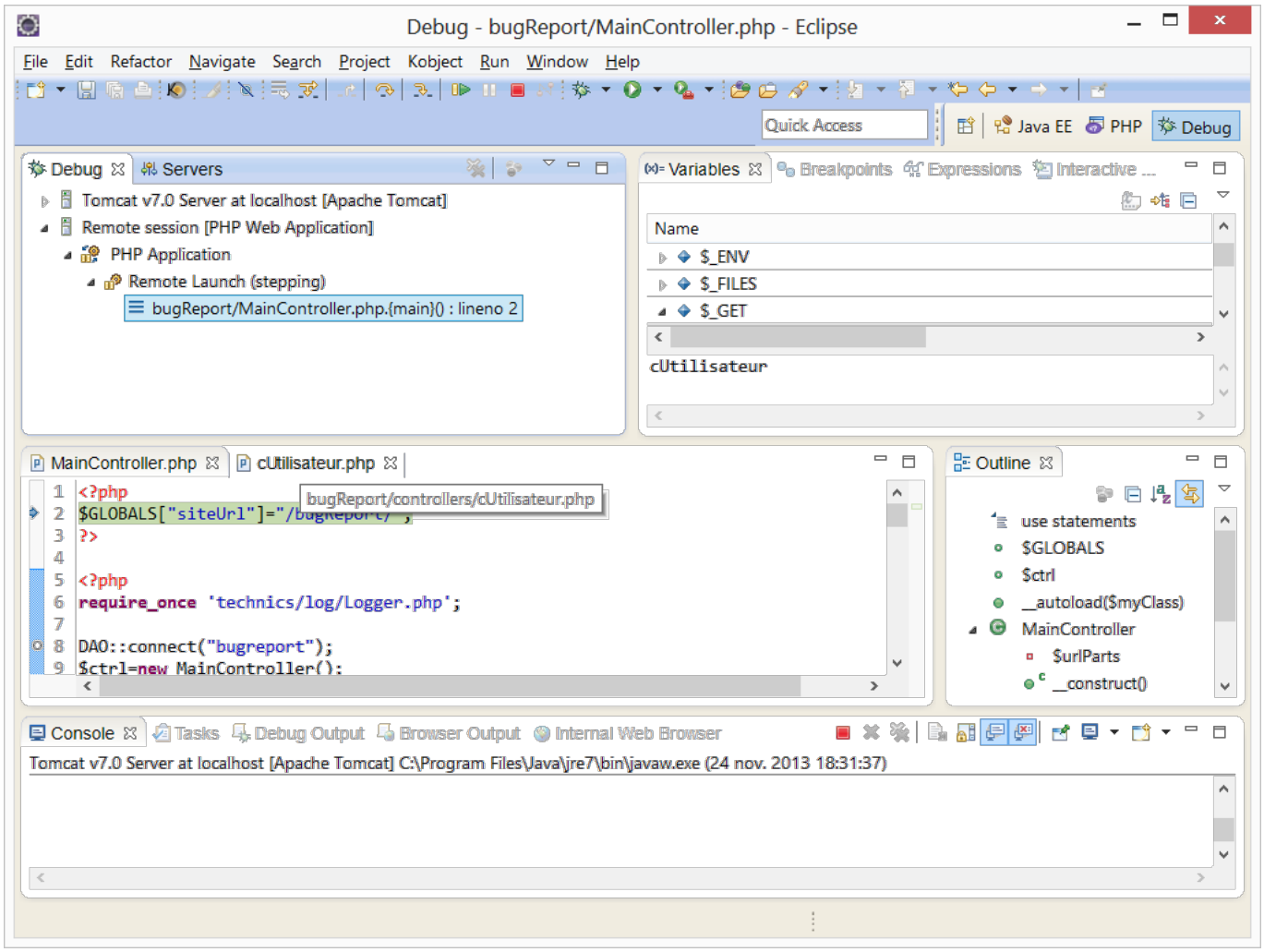


Rafraîchir la page pour provoquer une nouvelle exécution du script, puis retourner à Eclipse :

Eclipse propose de passer en perspective Debug :



L'exécution s'est arrêtée sur la première ligne.



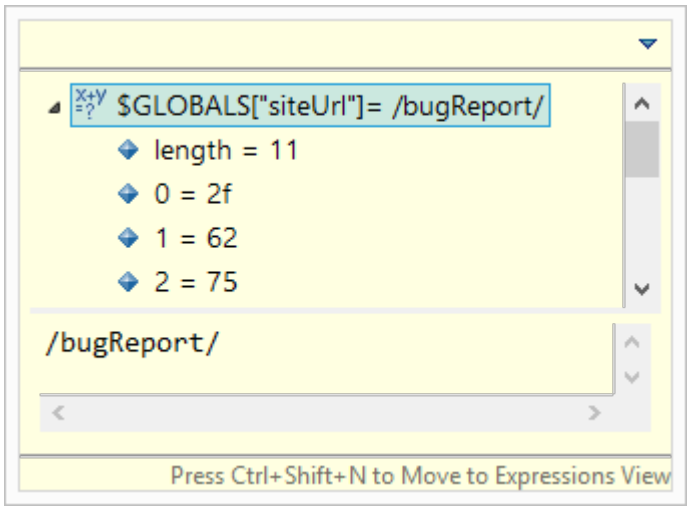
Il est alors possible de reprendre l'exécution :

	en pas à pas, d'instruction en instruction sans parcourir les fonctions appelées (**step over**)
	en pas à pas, d'instruction en instruction en parcourant les fonctions appelées (**step into**)
	de reprendre normalement l'exécution du script (jusqu'à sa fin ou jusqu'au prochain point d'arrêt (**resume**)
	d'arrêter la session de débogage (**stop**)

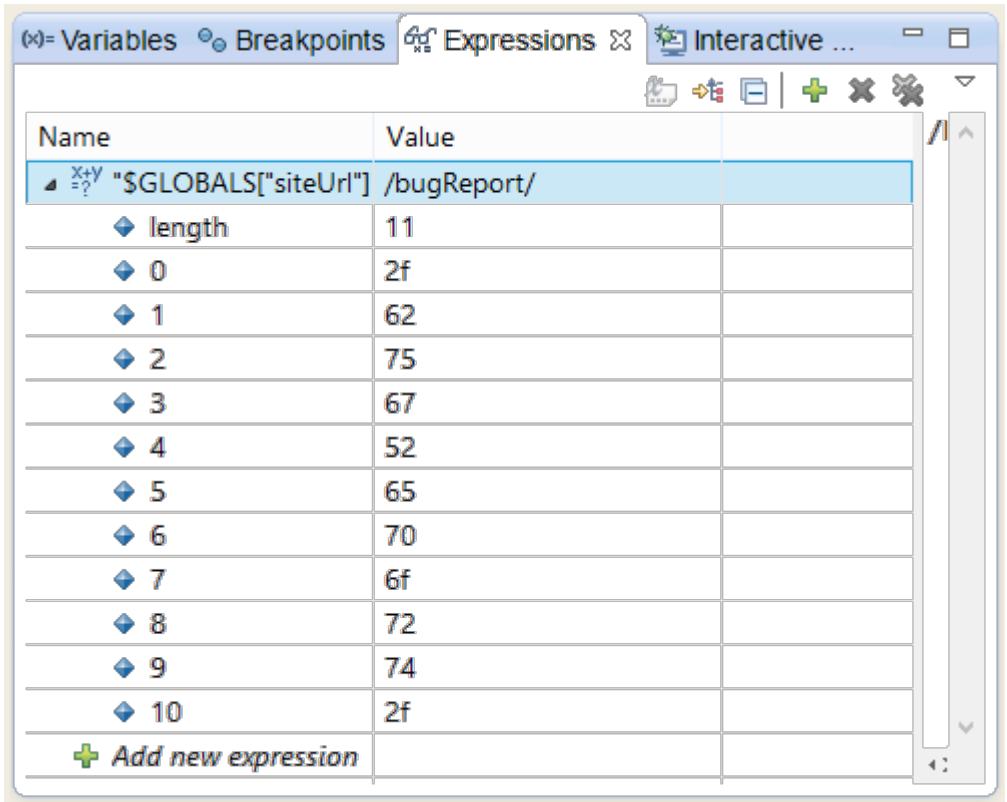
Inspection de variables

Il est alors possible :

- d'inspecter ponctuellement la valeur prise par une expression (Bouton droit de la souris sur l'expression à évaluer puis **Inspect** dans le menu) :

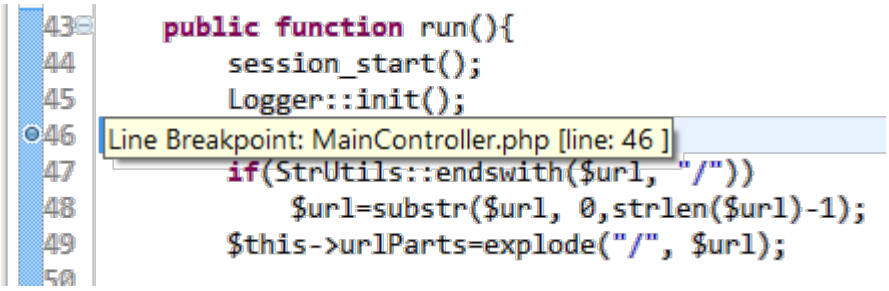


De maintenir la valeur de l'expression affichée (Bouton droit de la souris sur l'expression à évaluer puis **Watch** dans le menu :



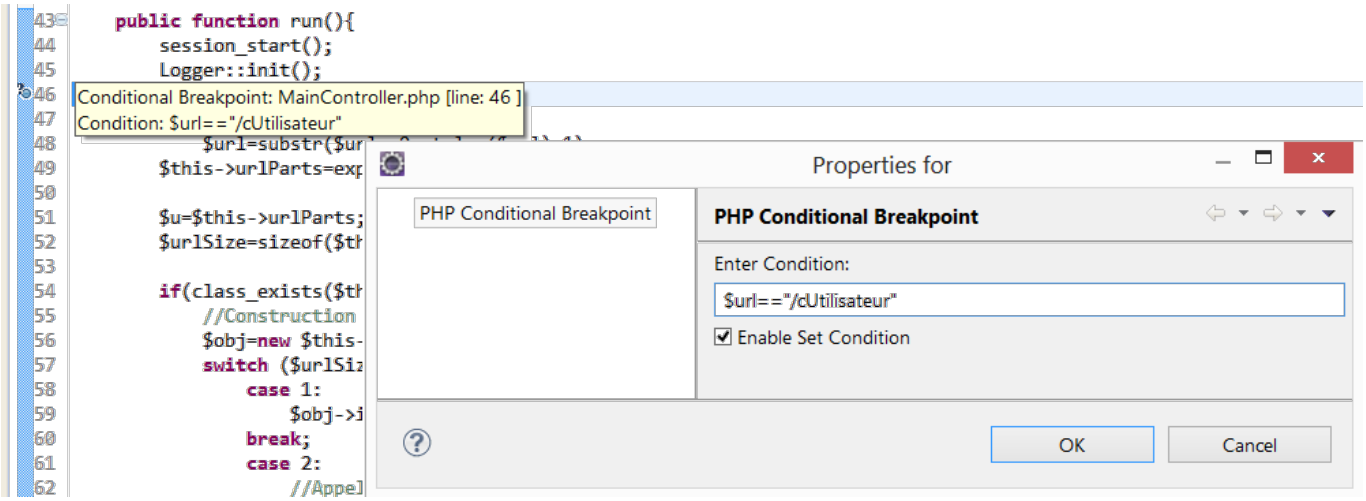
Poser un point d'arrêt

Double cliquer dans la barre verticale des numéros de ligne, au niveau de la ligne sur laquelle l'exécution doit s'interrompre :



Relancer ensuite l'exécution (F5 dans le navigateur par exemple).

Il est également possible de poser un point d'arrêt conditionnel. L'exécution ne sera interrompue dans ce cas que si la condition est vraie :



Poser le point d'arrêt normalement (double-clic), puis aller dans **breakpoint properties** en utilisant le menu contextuel.

-- Débogage côté client

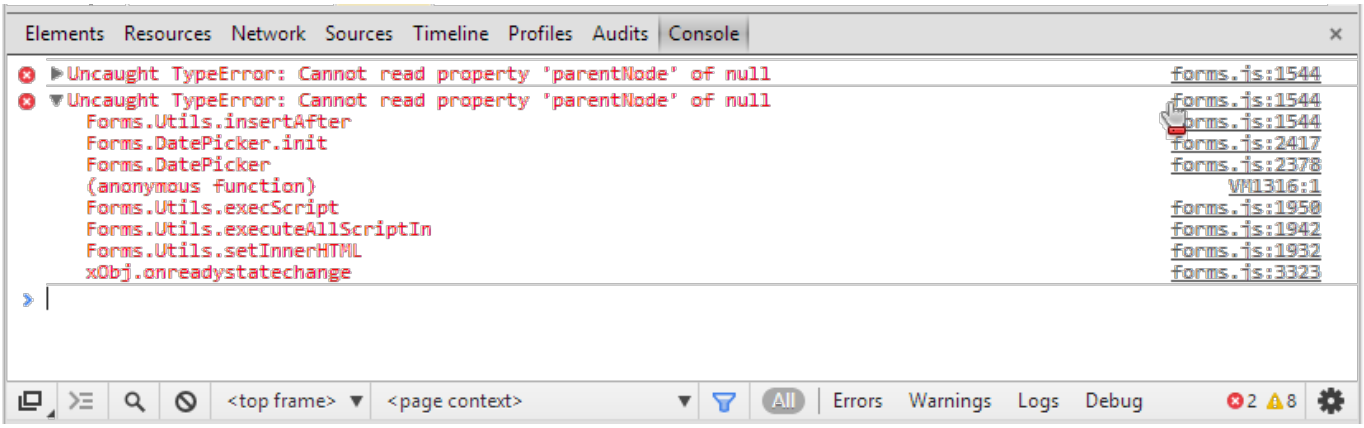
Difficile parfois de trouver les erreurs dans les scripts côté client et les requêtes ajax, il convient donc d'utiliser les outils de développement fournis avec les navigateurs, ou d'en installer en supplément.

-- Chrome

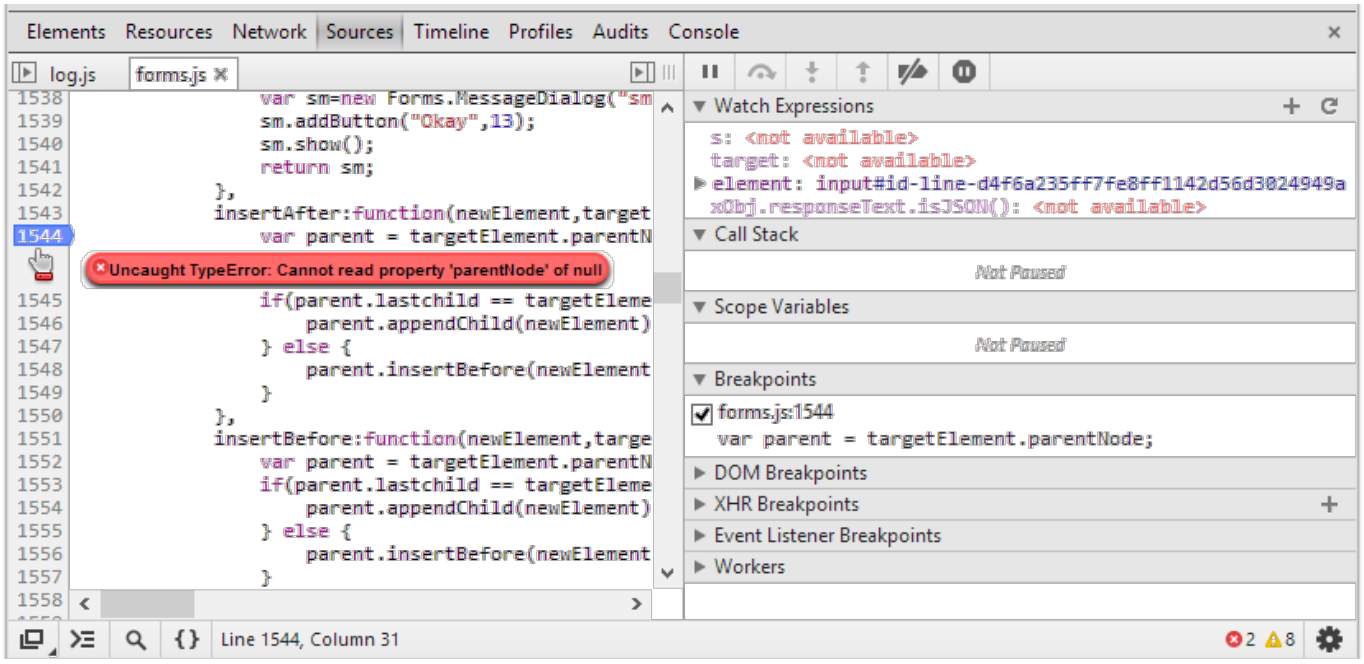
-- Débogage des scripts

Chrome (Version 31.0.1650.57 m) est fourni avec des outils de développement efficaces, accessibles par le menu : **Outils/Outils de développement**

La console d'erreur affichée par défaut montre les erreurs, warnings et/ou informations de log.



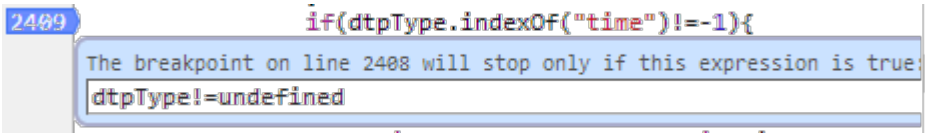
Il est ensuite possible d'ouvrir le fichier concerné (dans le cas présent **forms.js**), puis de poser un point d'arrêt sur une ligne en cliquant sur la barre de numérotation des lignes.



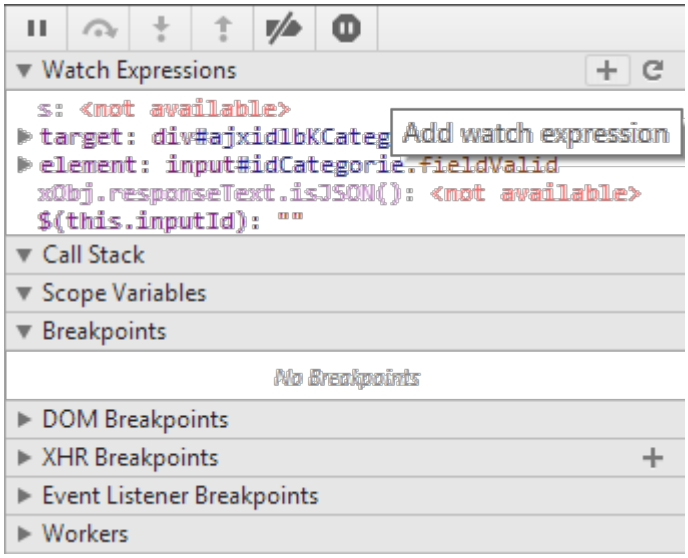
La nouvelle exécution du script permettra de s'arrêter au point d'arrêt, puis de reprendre l'exécution :

	en pas à pas d'instruction en instruction sans parcourir les fonctions appelées
	en pas à pas en parcourant les fonctions appelées

Le point d'arrêt peut être conditionnel (à poser avec le bouton droit de la souris) :

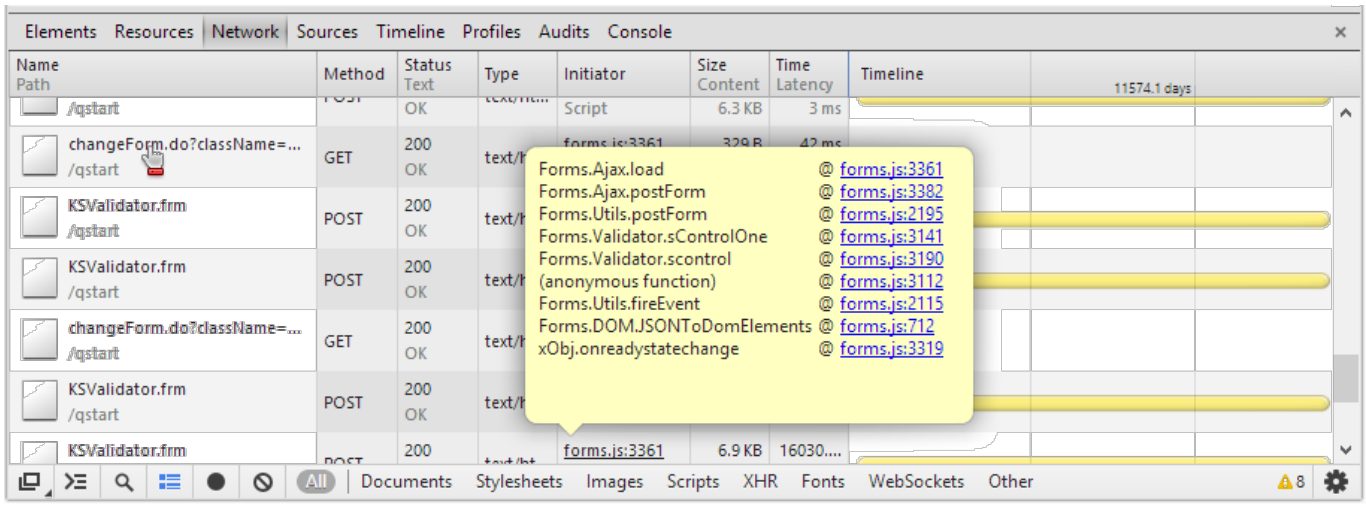


Il est également possible d'inspecter des variables, soit en les survolant à la souris, soit en les ajoutant aux addWatch :

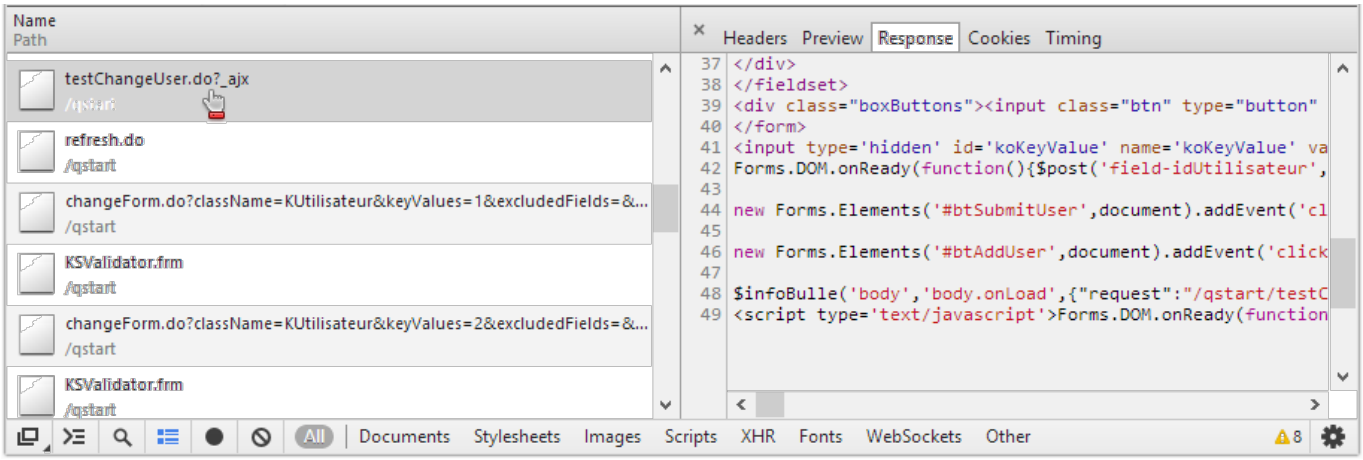


-- Inspection des requêtes

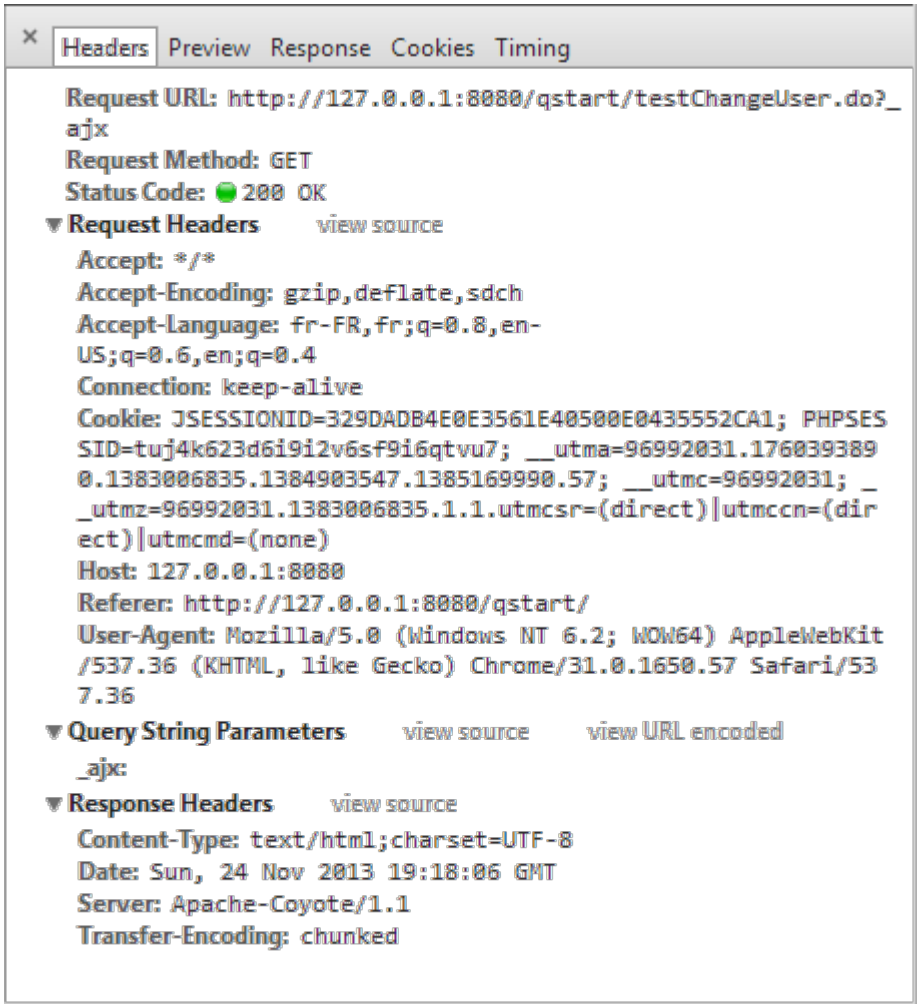
Afficher le volet réseau (**network**), cliquer sur la requête ajax à inspecter :



Chrome affiche le contenu de la réponse :

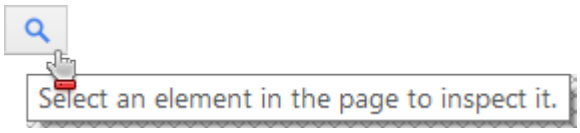


Il est possible d'inspecter également les en-têtes et paramètres de la requête.



-- Inspection DOM

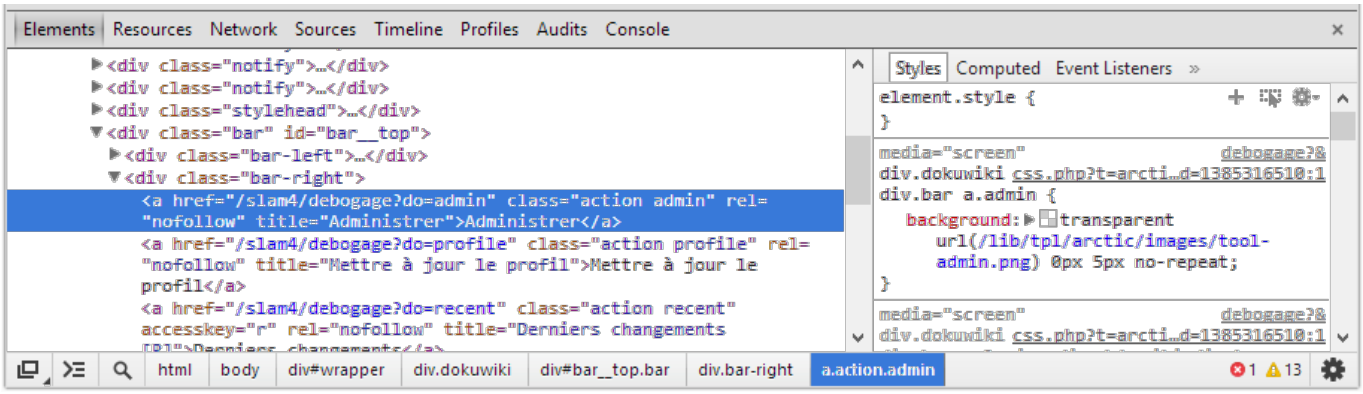
L'outil d'inspection DOM permet d'obtenir des informations sur l'arborescence DOM de la page :



Il suffit de cliquer ensuite sur l'élément DOM à inspecter



Pour obtenir la structure à laquelle il appartient, ou les informations de style :



From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/debogage?rev=1385342324

Last update: **2019/08/31 14:39**