Ajax et services web

Alexandre Niveau

GREYC — Université de Caen

En partie adapté du cours de Jean-Marc Lecarpentier

Ajax et Web Services

- Ajax permet de faire des requêtes vers le serveur *indépendamment* de l'internaute.
- Peut-on l'utiliser pour faire des requêtes vers un autre serveur, par exemple un service web ?

  ➤ problème de la Same-origin policy [http://en.wikipedia.org/wiki/Same_origin_policy]

Same-origin policy

- Idée générale : les pages accédées sont isolées les unes des autres
  - Un script sur une page ne peut pas accéder au DOM d'une autre page
  - Si ce n'était pas le cas, on ne pourrait jamais visiter des sites en lesquels on n'a pas entièrement confiance, car ils pourraient récupérer les cookies des autres sites et les utiliser pour faire n'importe quoi (*cross-site request forgery*)
- En fait, ce n'est pas aussi extrême : les pages d'un même site peuvent accéder les unes aux autres

  ➤ Seules les pages *d'origine différente* sont inaccessibles
- Même origine = même protocole + même port + même hôte (domaine)
- *cross-origin request* = requête vers une origine différente de celle de la page

Ce qui est concerné par la *same-origin policy*

- Ce qu'il est interdit ou autorisé de faire en *cross-origin* n'est pas tout à fait intuitif !
- D'après MDN [https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy#Cross-origin_network_access], de façon générale :
  - *écrire* vers une origine différente est autorisé (liens, redirections, soumissions de formulaire)
  - *embarquer* du contenu d'une origine différente est autorisé (CSS, JS, images, vidéos, fichiers audio… et n'importe quel type de contenu, avec une `iframe`, sauf si le serveur l'interdit [https://developer.mozilla.org/en-US/

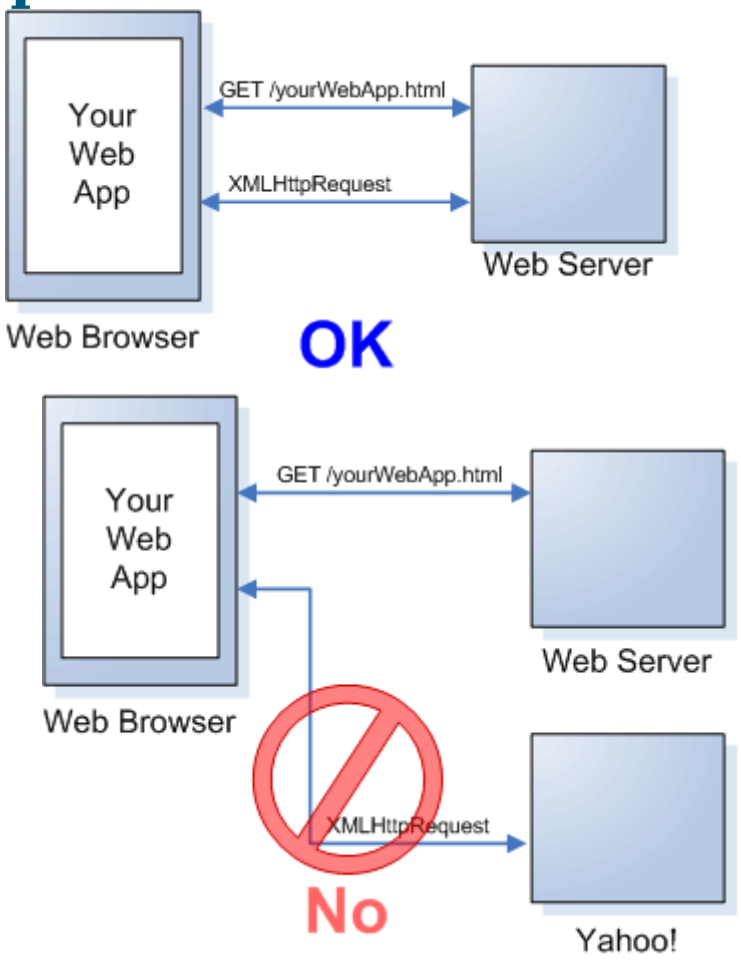
docs/Web/HTTP/X-Frame-Options])

- *lire* du contenu d'une origine différente est interdit (accès au DOM et aux cookies, XMLHttpRequest)

  ➤ sinon n'importe quelle page pourrait accéder à votre webmail, par exemple

- *modifier* du contenu d'une origine différente (méthodes HTTP PUT ou DELETE) est (assez logiquement) interdit

Problème pour les services web



- Requête XMLHttpRequest impossible vers un domaine différent
- Impossibilité d'appeler directement des services tiers

  démo [demo/crossdomainXHR.html]

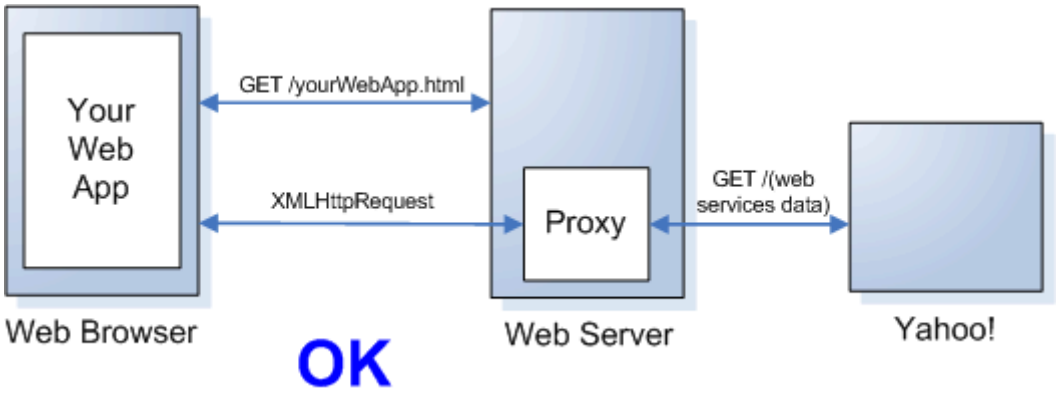
- Quelles solutions ?

Accès à tous les domaines

- Solutions :
  - Proxy (c'est le serveur qui interroge le service web)
  - JSONP (la réponse du service est un script à exécuter)
  - CORS (gestion des domaines autorisés dans l'en-tête HTTP)

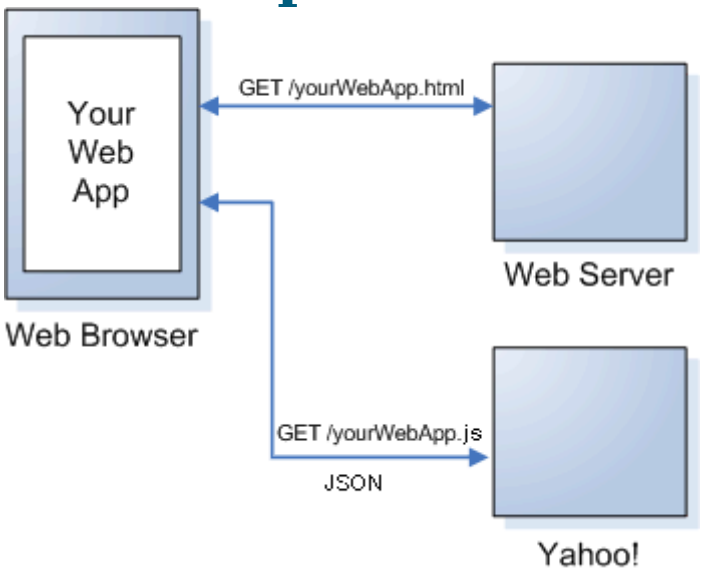
Utilisation d'un proxy

- On ne fait pas directement la requête Ajax vers le service web, on la fait à notre serveur qui s'occupe de la « transférer ».



- Deux options possibles pour le proxy :
  - simple transfert des données brutes (appli client lourde)
  - traitement des données reçues (appli client légère)

Utilisation d'un script



- Balise `<script>` non soumise à la *same origin policy*

  ➤ une page peut exécuter un script fourni par n'importe quelle autre page
- Si un service web fournit son résultat sous forme d'un script, on pourra donc toujours l'exécuter…
- Nécessite d'avoir confiance dans le service : on rend nos internautes vulnérables !
- Format : généralement JSONP

Principes de JSONP

- JSONP : JSON with Padding
- Données structurées en JSON, mais entourées d'un appel à une fonction

- Exécuter le code reçu revient à exécuter la fonction avec l'objet JSON en paramètre

JSONP : fonctionnement

- Appel : création d'une balise `<script>` vers l'URL du web service, qui prend des paramètres, en général :
  - les paramètres de la requête
  - un identifiant pour l'utilisation de l'API
  - le nom de la fonction *callback* qui sera utilisée pour le traitement des données

  Exemple :
  `http://monservice.com?key=x32b&search=flower&callback=toto`
- Réponse sous la forme `toto({ JSON data });`, donc l'exécution du script lance la fonction `toto`

  ➤ on a pris soin de déclarer une fonction `toto` qui prend en paramètre l'objet JSON contenant les données structurées

Démo : interrogation de l'API Google Books au format JSONP [demo/books_JSONP.html]

JSONP : conclusion

- Principe simple pour interroger des APIs
- Tout s'appuie sur JSON et un appel de fonction
- Attention à la sécurité des données et des applications : on exécute du code inconnu
- Voué à disparaître avec le développement de CORS

CORS : exemple introductif

- GMail ne veut pas autoriser les requêtes XHR depuis d'autres sites que `mail.google.com`

  ➤ c'est prévu : les navigateurs l'interdisent par le biais de la *same-origin policy*
- En revanche, Google Books est un *service web* : Google voudrait que n'importe quelle page puisse faire des requêtes XHR vers `www.googleapis.com/books`

  ➤ cette fois la *same-origin policy* est un obstacle
- Solution : que chaque serveur puisse indiquer au client si la *same-origin policy* doit être respectée ou non

  ➤ en-têtes HTTP spéciaux, définis dans le standard CORS [http://www.w3.org/TR/cors/] (*Cross-Origin Resource Sharing*, partage de ressources entre origines différentes). Maintenant largement supporté [http://caniuse.com/#feat=cors]

CORS : utilisation basique

- Client ajoute l'en-tête HTTP `Origin` dans ses requêtes
- Serveur répond avec l'en-tête HTTP `Access-Control-Allow-Origin` pour indiquer les domaines autorisés
- Exemples

```
Access-Control-Allow-Origin: null
Access-Control-Allow-Origin: *
Access-Control-Allow-Origin: http://foo.fr
Access-Control-Allow-Origin: http://foo.fr http://bar.net
```

CORS et XHR2

- XHR2 gère le CORS basique automatiquement : pas besoin d'ajouter un en-tête `Origin` manuellement

Démo : récupération d'un fragment HTML avec CORS [demo/XHR_CORS.html]

On a vu au début que la même démo sans CORS ne fonctionne pas [demo/crossdomainXHR.html] ; comparer les messages HTTP dans les deux cas avec la console web.

Démo : interrogation de l'API Google Books avec CORS [demo/books_XHR.html]

CORS avancé

- Si besoin d'utiliser des méthodes moins classiques que GET et POST (PUT, DELETE…), ou des en-têtes personnalisés, ça ne suffit pas
- Objectif : éviter de créer des failles de sécurité dans les serveurs existants. En effet :
  - avant CORS, impossible de faire un DELETE depuis un domaine autre que le sien, donc les serveurs ne se sont pas nécessairement protégés contre cette possibilité
  - Si CORS rendait ça possible, cela pourrait donc créer des failles de sécurité pour les serveurs qui s'appuyaient sur la *same-origin policy* pour interdire PUT ou DELETE, et qui mettraient en place CORS sans se rendre compte des implications sur les requêtes autres que GET

Preflight

- Solution : *preflight* des requêtes pour savoir si autorisation accordée
- Exemple : le client veut utiliser PUT
  - il commence par demander si c'est possible : première requête avec

`Access-Control-Request-Method: PUT`

- le serveur répond avec `Access-Control-Allow-Methods: PUT`
- le client peut ensuite envoyer sa véritable requête PUT

- Intérêt :
  - le serveur peut autoriser les clients à utiliser des méthodes « avancées » comme PUT ou DELETE : il lui suffit d'implémenter le protocole ci-dessus
  - **mais** par défaut ces méthodes sont interdites, puisque le serveur doit les autoriser explicitement

Conclusion

- Avec CORS, les fournisseurs de services peuvent les rendre utilisables depuis du code client, sans casser la *same-origin policy* sur laquelle s'appuient tous les autres sites
- Utilisation très simple pour le client (transparent avec XHR2), et beaucoup plus sécurisé que JSONP (on n'exécute pas de code inconnu)

Spécifications et normes

- Cross-Origin Resource Sharing [http://www.w3.org/TR/cors/] (W3C Rec)

Références et guides

- Same-origin policy [https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy] (MDN)

Tutoriels

- DOM access control using CORS [http://dev.opera.com/articles/view/dom-access-control-using-cross-origin-resource-sharing/] sur dev.opera
- Cross-site XMLHttpRequest with CORS [http://hacks.mozilla.org/2009/07/cross-site-xmlhttprequest-with-cors/]

Lectures complémentaires

- Article introduisant JSONP [http://bob.ippoli.to/archives/2005/12/05/remote-json-jsonp/]