

Architectures N-Tiers

IMT Alès

2020

François Pfister - Connecthive

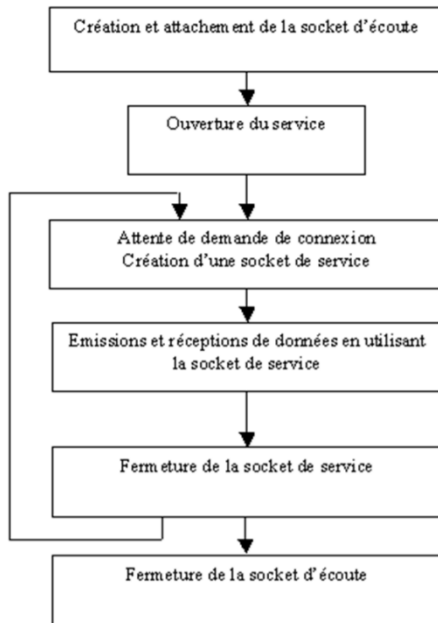
Planning

- 18/11 (6h): Architecture; Http, Html5, Javascript, Backend vs Front-end
- 30/11 (6h): Webapp; ORM – EJB, Frameworks, Deployment
- 2/12 (3h): Préparation de l'évaluation
- Sujet: application collaborative
- ~~Option A: jeu en réseau (genre bataille navale), interactions basées sur les websockets, dans le prolongement du sujet client-serveur.~~
- Option B: application collaborative, interactions basées sur les websockets: réaliser un clone de l'application Slack, coder le serveur et le client web.

Du serveur Socket vers le serveur HTTPD

29

Fonctionnement du serveur



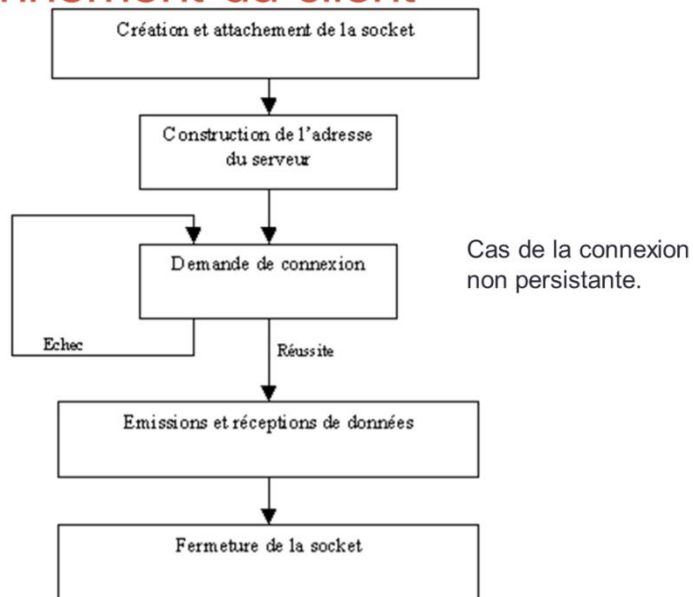
- Un serveur HTTPD est un serveur de sockets
- Les trames véhiculent le vocabulaire HTTP et HTML
- Le séquençage temporel répond à la spécification du protocole HTTP

Voir cours Applications Client-Serveur-2020.pptx

Du client socket vers le navigateur HTML

31

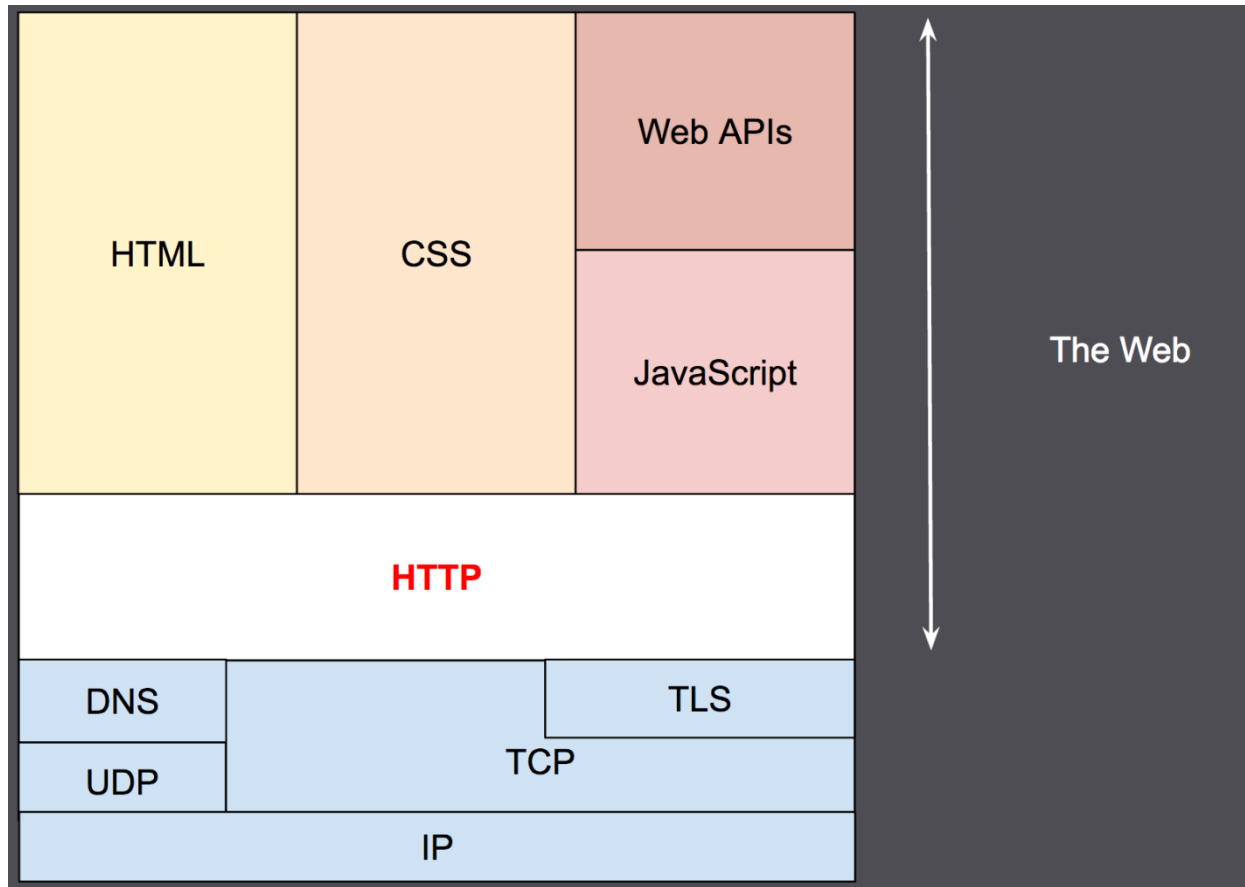
Fonctionnement du client



- Un navigateur est un client de sockets
- Les trames reçues du serveur véhiculent le vocabulaire HTTP et HTML
- Le navigateur décode le vocabulaire et produit un affichage cohérent avec la sémantique du vocabulaire HTML.
- De plus, le navigateur interprète des scripts par le biais du langage javascript.

Voir cours Applications Client-Serveur-2020.pptx

Le domaine WEB au-dessus du domaine SOCKET



Spécification, W3C

<https://www.w3.org/>

Le consortium W3C définit l'Open Web Platform, plate-forme WEB s'appuie sur des technologies telles que:

Semantic web, HTML5, XML, CSS, javascript, SVG, etc..



Web Design and Applications

Cette thématique concerne les normes de construction et de rendu des pages Web, y compris HTML, CSS, SVG, Ajax et d'autres technologies pour les applications Web (« WebApps »). Cette section contient également des informations sur la façon de rendre les pages accessibles aux personnes handicapées (WCAG), de les internationaliser, et les faire fonctionner sur des appareils mobiles.



Web Architecture

Web Architecture se concentre sur les technologies fondamentales et les principes qui maintiennent le Web, y compris les URI and HTTP.



Semantic web

En plus de la classique «Web des documents» W3C aide à construire une les technologies pour soutenir un «Web de données». Le but ultime du Web de données est de permettre aux ordinateurs de faire un travail plus utile et de développer des systèmes qui peuvent supporter des interactions fiables sur le réseau. Le terme «Web sémantique» se réfère à la vision du W3C du Web de données liées. Les technologies du Web sémantique permettent aux utilisateurs de créer des banques de données sur le Web, de construire des vocabulaires et écrire des règles pour gérer les données. Les données liées sont construites sur des technologies telles que RDF, SPARQL, OWL, SKOS ...

Spécification, W3C



XML Technology

les technologies XML comprenant XML, XML Namespaces, XML Schema, XSLT, Efficient XML Interchange (EXI), et les autres standards relatifs au XML.



Web of services

Web of Services fait référence aux services basés sur les messages fréquemment utilisés sur le Web et dans les logiciels d'entreprise. Le « Web of Services » est basé sur des technologies telles que HTTP, XML, SOAP, WSDL, SPARQL, et autres...



Web of devices

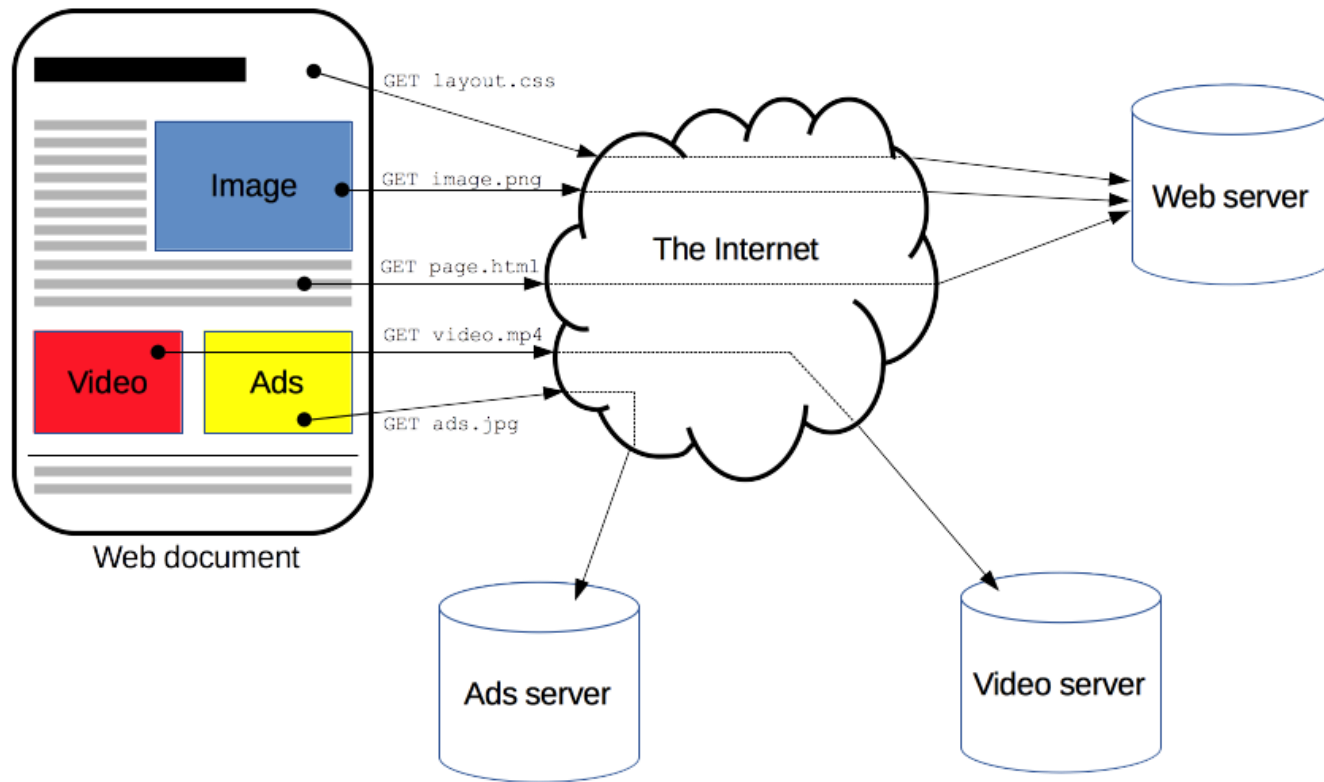
Le W3C se concentre sur des technologies permettant l'accès au Web de partout, n'importe quand et sur n'importe quel objet. Cela inclue l'accès web à partir de téléphones/smartphones, mais également à partir de produits d'électronique grand public, imprimantes, télévisions connectées et même des véhicules.



Browsers and Authoring Tools

L'utilité du web et sa croissance dépendent de son universalité. Nous devrions être capable de publier du contenu indépendamment du logiciel que nous utilisons, de l'ordinateur que nous avons, du langage que nous parlons, que nous soyons connecté en filaire ou sans fils... De même, nous devrions être capables d'accéder au Web à partir de n'importe quel matériel qui se connecte à l'Internet – fixe ou mobile, petit ou grand. Le W3C facilite cette vision via des standards web internationaux. Ces standards garantissent que toutes ces merveilleuses innovations digitales continuent à améliorer le Web accessible à tous.

Structure d'un document HTML



Navigateur, « user-agent »

Le navigateur Web assure le rôle d'"agent utilisateur" (user-agent); d'autres user-agent peuvent exister, il s'agit des clients http remplaçant le navigateur pour des usages de mise au point ou d'autres mécanismes spécifiques.

Le navigateur est toujours l'entité qui lance la demande. Ce n'est jamais le serveur; cependant des protocoles permettant au serveur de prendre l'initiative ont été définis (websockets)..

Pour présenter une page Web, le navigateur envoie une requête pour récupérer le document HTML qui représente la page. Il analyse ensuite ce fichier, effectuant des requêtes supplémentaires correspondant aux scripts d'exécution, aux informations de mise en page (CSS) à afficher et aux sous-ressources contenues dans la page (généralement des images et des vidéos). Le navigateur Web mélange ensuite ces ressources pour présenter à l'utilisateur un document complet, la page Web. Les scripts exécutés par le navigateur peuvent récupérer plus de ressources dans les phases ultérieures et le navigateur met à jour la page Web en conséquence.

Une page Web est un document hypertexte. Cela signifie que certaines parties du texte affiché sont des liens qui peuvent être activés (généralement par un clic de souris) pour récupérer une nouvelle page Web, permettant à l'utilisateur de diriger son utilisateur-agent et de naviguer sur le Web. Le navigateur traduit ces instructions en requêtes HTTP et interprète en outre les réponses HTTP pour présenter à l'utilisateur une réponse claire.

HTTP

Simple: lisible par l'homme

Extensible: par les en-têtes, des extensions peuvent être implémentées (par exemple websockets)

Sans état, mais le mécanisme des cookies permet de conserver un état pour la session en cours.

Fiable: HTTP s'appuie sur TCP (et non UDP), l'intégrité des messages est transmise.

Les connexions sont rendues persistantes grâce à un mécanisme de pipelining.

Autres fonctionnalités:

Mise en cache

Flux HTTP

```
1 | GET / HTTP/1.1
2 | Host: developer.mozilla.org
3 | Accept-Language: fr
```

Requête

```
1 | HTTP/1.1 200 OK
2 | Date: Sat, 09 Oct 2010 14:28:02 GMT
3 | Server: Apache
4 | Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
5 | ETag: "51142bc1-7449-479b075b2891b"
6 | Accept-Ranges: bytes
7 | Content-Length: 29769
8 | Content-Type: text/html
9 |
10 | <!DOCTYPE html... (here comes the 29769 bytes of the requested we
```

Réponse

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>

TP

Retour aux sockets

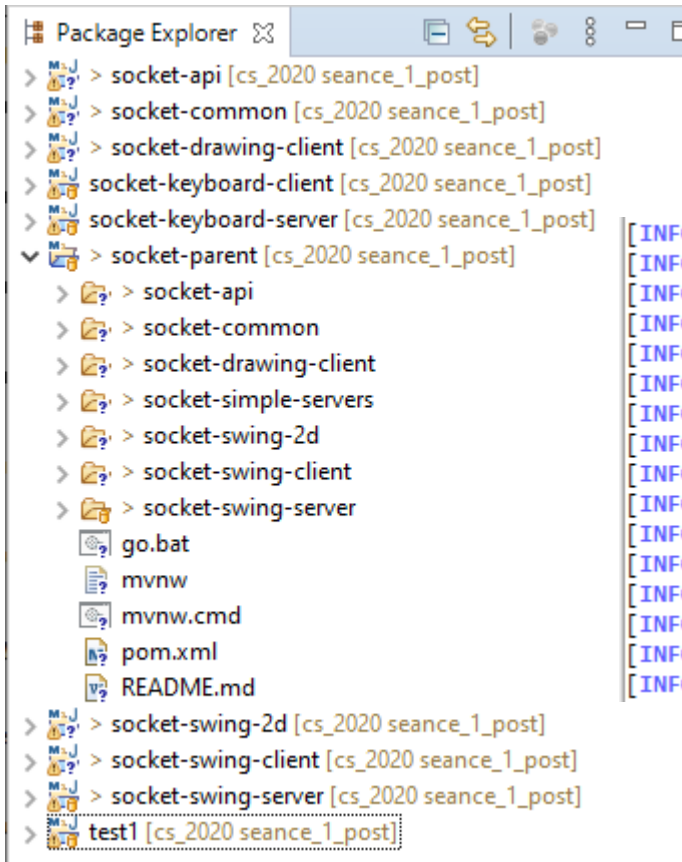
Les TP NTiers se font dans la continuité des TP client-serveur.

Il faut faire un checkout du dépôt git (branche master):

https://bitbucket.org/francois_pfister/cs_2020

Faites le dans un nouveau workspace (et conservez l'existant)

TP retour aux sockets



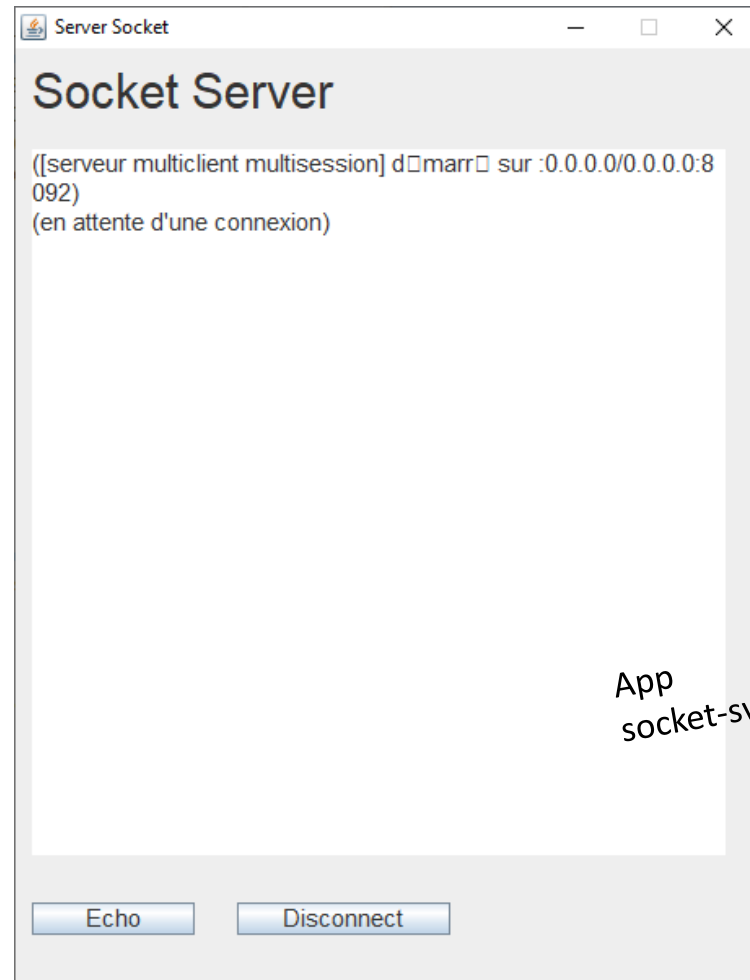
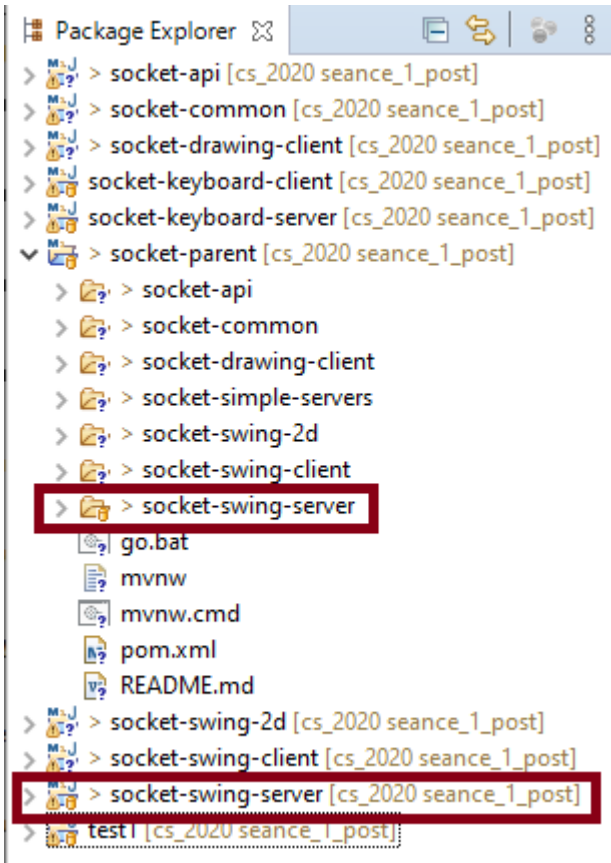
The screenshot shows an IDE window with a Package Explorer on the left and a console output on the right. The Package Explorer displays a Maven project structure for 'socket-parent'. The console output shows the build process for 'socket labs 0.0.1-SNAPSHOT', listing various modules and their build times, all of which completed successfully.

```
Package Explorer [cs_2020 seance_1_post]
> > socket-api [cs_2020 seance_1_post]
> > socket-common [cs_2020 seance_1_post]
> > socket-drawing-client [cs_2020 seance_1_post]
> > socket-keyboard-client [cs_2020 seance_1_post]
> > socket-keyboard-server [cs_2020 seance_1_post]
▼ > socket-parent [cs_2020 seance_1_post]
  > > socket-api
  > > socket-common
  > > socket-drawing-client
  > > socket-simple-servers
  > > socket-swing-2d
  > > socket-swing-client
  > > socket-swing-server
  go.bat
  mvnw
  mvnw.cmd
  pom.xml
  README.md
> > socket-swing-2d [cs_2020 seance_1_post]
> > socket-swing-client [cs_2020 seance_1_post]
> > socket-swing-server [cs_2020 seance_1_post]
> test1 [cs_2020 seance_1_post]
```

```
[INFO] -----
[INFO] Reactor Summary for socket labs 0.0.1-SNAPSHOT:
[INFO]
[INFO] socket labs ..... SUCCESS [ 0.498 s]
[INFO] socket api ..... SUCCESS [ 2.170 s]
[INFO] socket common ..... SUCCESS [ 0.605 s]
[INFO] socket swing 2D ..... SUCCESS [ 2.284 s]
[INFO] socket swing client ..... SUCCESS [ 1.052 s]
[INFO] socket swing server ..... SUCCESS [ 0.821 s]
[INFO] socket drawing client ..... SUCCESS [ 0.761 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 8.434 s
[INFO] Finished at: 2020-11-17T20:43:16+01:00
[INFO] -----
```

Restructuration avec un projet socket-parent, un seul build du projet parent est effectué.

Lancement du serveur

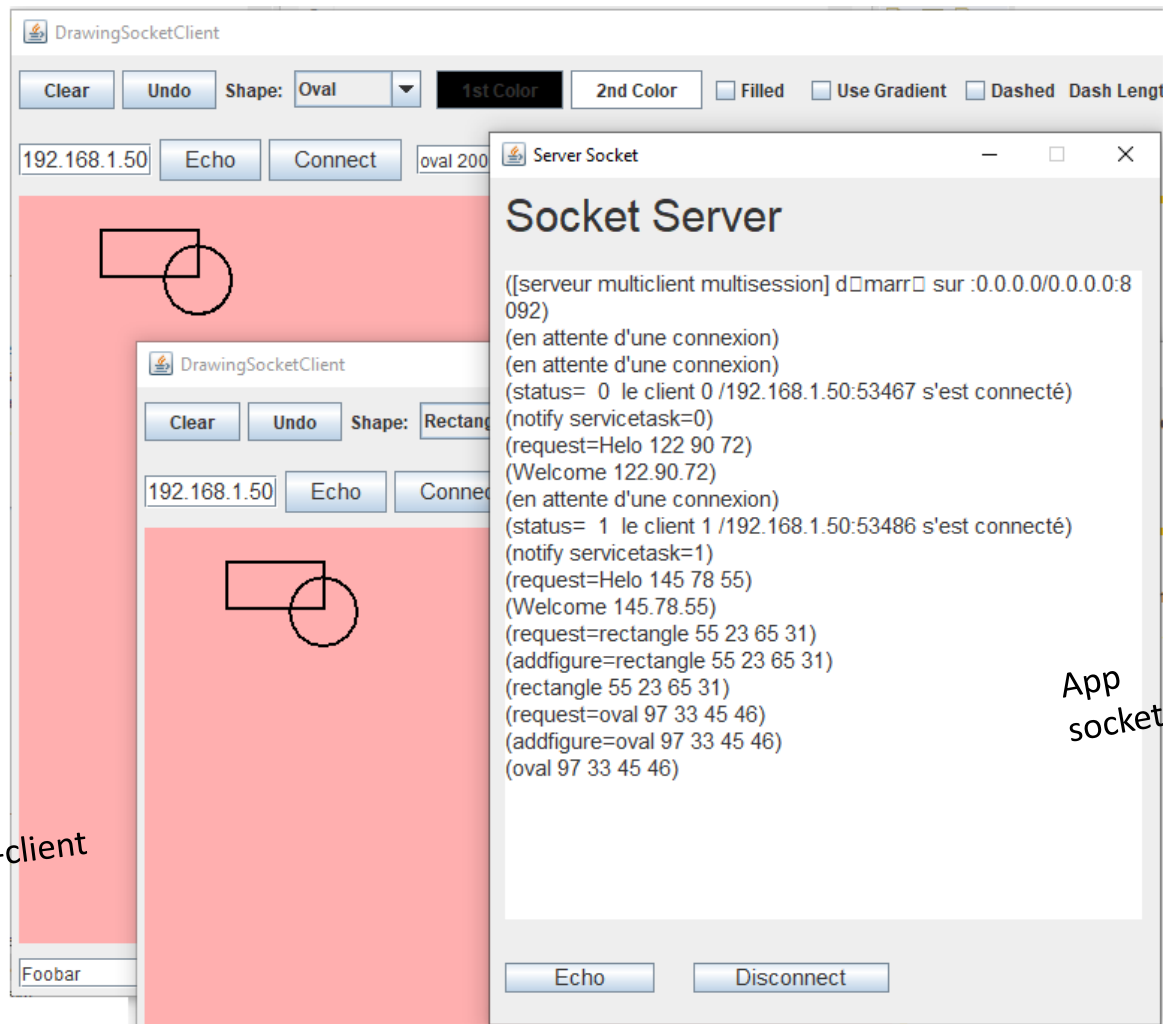


App
socket-swing-server

Lancement de 2 clients Draw2d

Le vocabulaire
Draw2d est
interprété

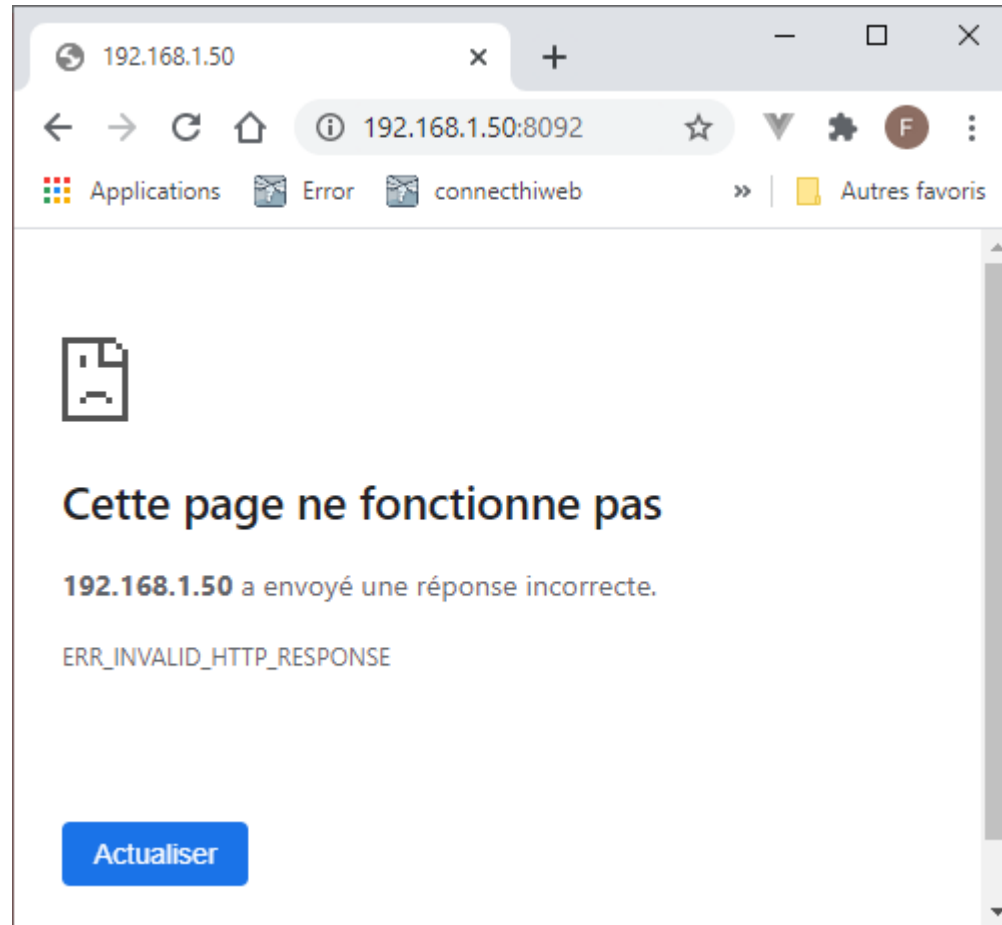
App
socket-drawing-client



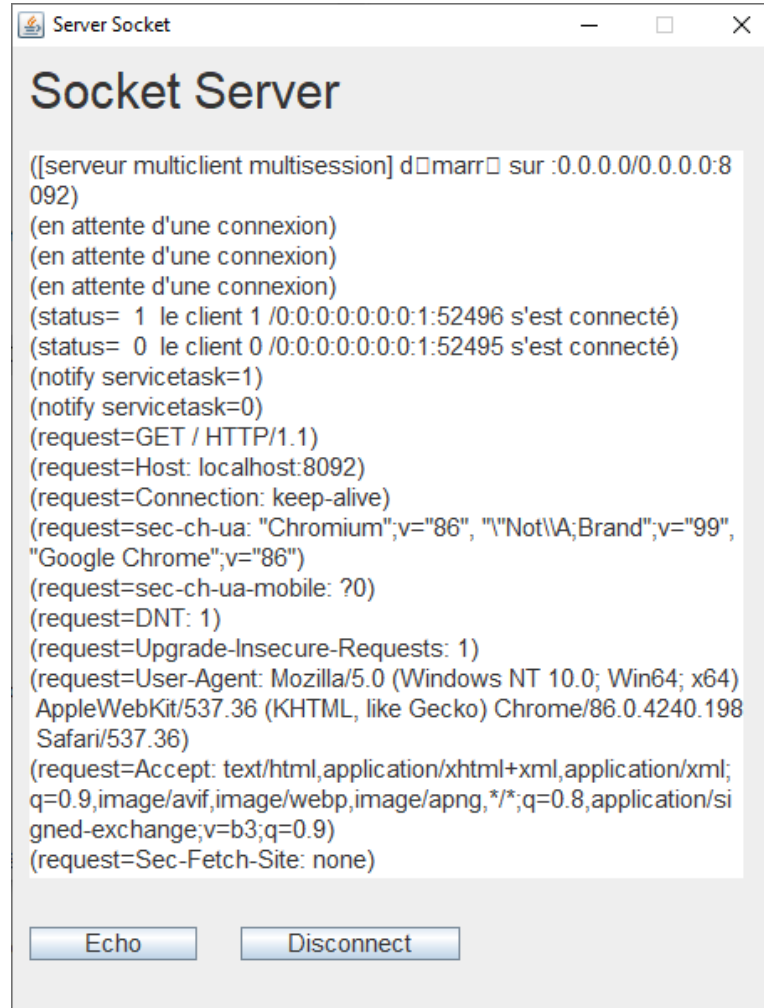
App
socket-swing-server

Lancement d'un client HTTP

Le vocabulaire
Http est
inconnu



Le vocabulaire HTTP n'est pas connu



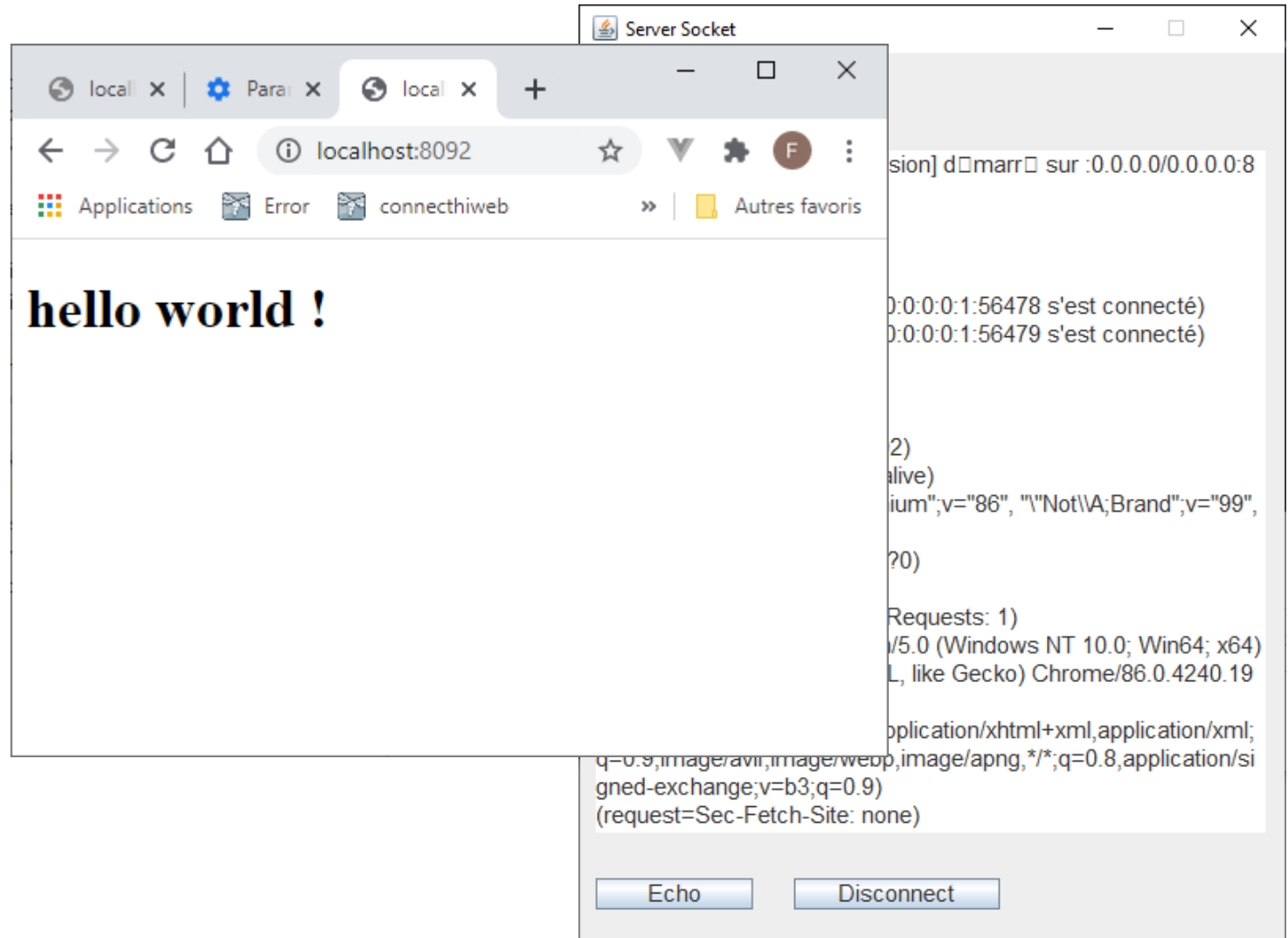
Réponse HTTP

public class HTTPHandler

```
private void writeHttpResponse(String fileRequested, String method, PrintStream out, BufferedOutputStream dataOut)
    throws IOException {
    if (fileRequested.endsWith("/")) {
        fileRequested += INDEX;
    }
    File file = new File(ROOT, fileRequested);
    int fileLength = (int) file.length();
    String content = getContentType(fileRequested);
    if (method.equals("GET")) {
        byte[] fileData = readFileData(file, fileLength);
        out.println("HTTP/1.1 200 OK");
        out.println("Server: Java HTTP Server from Connective : 1.0");
        out.println("Date: " + new Date());
        out.println("Content-type: " + content);
        out.println("Content-length: " + fileLength);
        out.println(); // blank line between headers and content, mandatory
        out.flush();
        dataOut.write(fileData, 0, fileLength);
        dataOut.flush();
    }
}
```

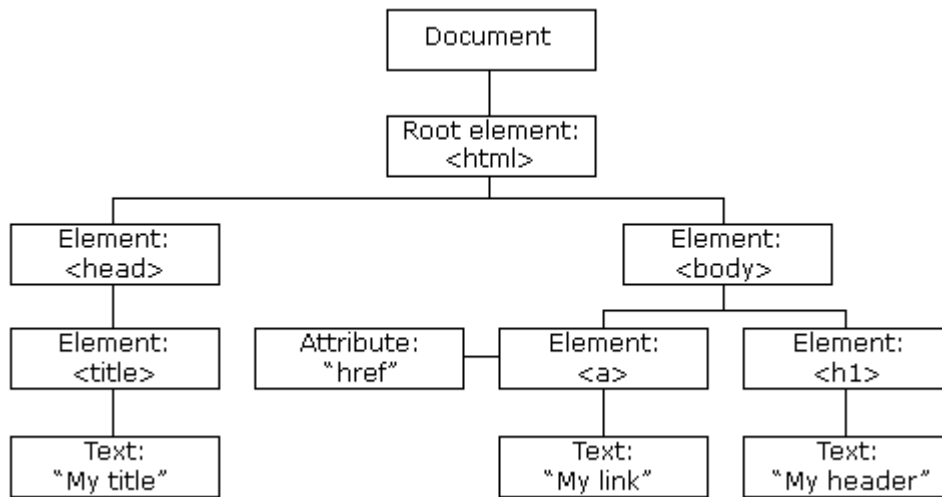
Du serveur Socket au serveur HTTP

Le vocabulaire
Http est
maintenant
connu et traité



HTML5

Document Object Model (DOM)



```
var myElement = document.getElementById("intro");
```

https://www.w3schools.com/js/js_htmlDOM.asp

HTML5

Javascript (SCRIPT)

```
1 <html>
2   <head>
3     <script>
4       // run this function when the document is loaded
5       window.onload = function() {
6
7         // create a couple of elements in an otherwise empty HTML page
8         const heading = document.createElement("h1");
9         const heading_text = document.createTextNode("Big Head!");
10        heading.appendChild(heading_text);
11        document.body.appendChild(heading);
12      }
13    </script>
14  </head>
15  <body>
16    </body>
17 </html>
```

https://www.w3schools.com/js/js_htmlDOM.asp

HTML5

Spécification

DOM



Living Standard — Last Updated 9 November 2020

Participate:

[GitHub whatwg/dom](#) (new issue, open issues)

[IRC: #whatwg on Freenode](#)

Commits:

[GitHub whatwg/dom/commits](#)

[Snapshot as of this commit](#)

[@thedomstandard](#)

Tests:

[web-platform-tests dom/](#) (ongoing work)

Translations (non-normative):

[日本語](#)

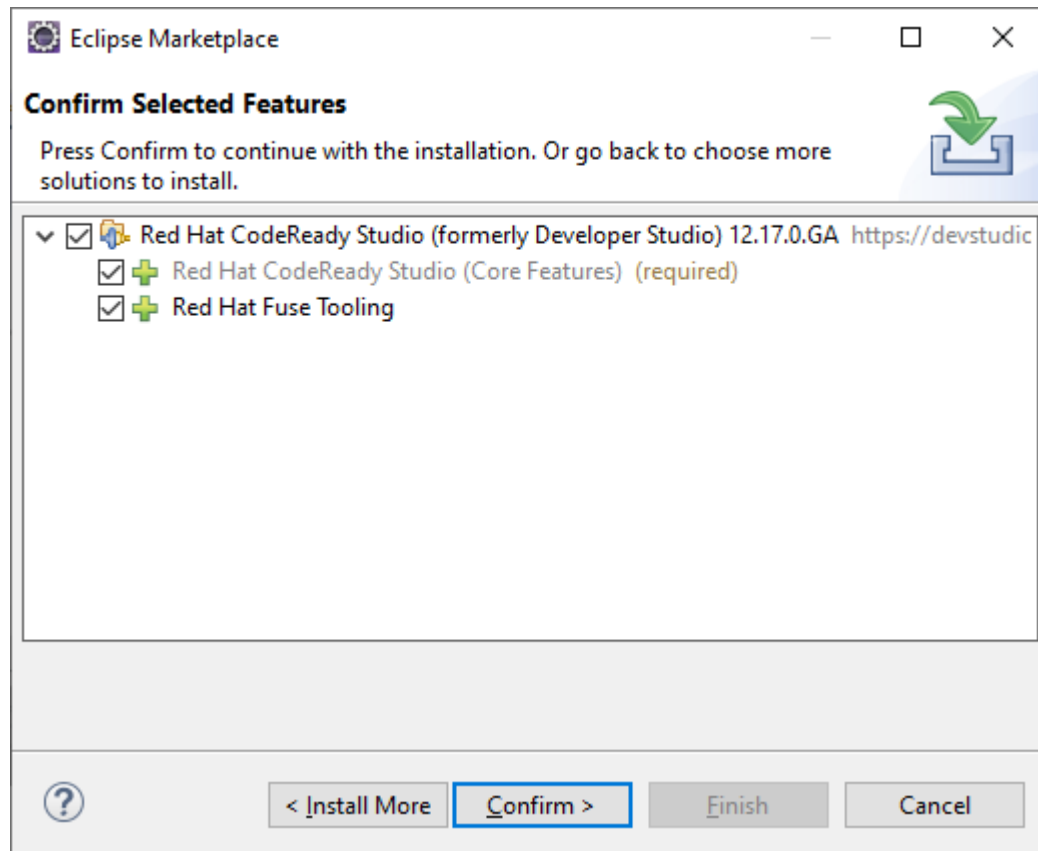
Abstract

[File an issue about the selected text](#) DOM defines a platform-neutral model for events, aborting activities, and node trees.

Tout est là: <https://dom.spec.whatwg.org/>
ou là: <https://www.w3.org/TR/DOM-Level-3-Core/introduction.html>

Quel serveur ?

<https://www.wildfly.org/downloads/>



Sujet

L'application à développer est un clone de Slack (<https://slack.com>). Il s'agit d'un environnement de travail collaboratif dont la base est une messagerie de groupe. Les utilisateurs s'inscrivent à des canaux, ces canaux comportant éventuellement des espaces de travail. Les messages sont émis par l'utilisateur vers un canal et les utilisateurs partenaires de ces canaux reçoivent les messages.

Il est possible d'ajouter des plugins à Slack, ce qui transforme la messagerie en socle d'applications distribuées. Un exemple de plugin est le tableau blanc. On peut y ajouter la visio-conférence, etc...

Travail à faire:

- Instancier un modèle métier basé sur EJB.
- Créer un jeu de données pour le test.
- Créer un serveur backend avec API REST (Jax-RS) pour les données et les règles de gestion.
- Créer un serveur web front-end (Html5 + vanilla javascript)
- Gérer les interactions en temps-réel en utilisant la technologie websocket.

Modèle métier

