PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 1: Introducció a la programació web client

Joan Pons Tugores

Índex

Índex de continguts

Introducció

Quan hem de dissenyar una aplicació tenim diverses opcions segons les seves característiques. Per exemple, si és una aplicació que no necessita recursos externs a la pròpia aplicació no necessitam més que un sol programa que realitzi les totes les tasques de l'aplicació. Un exemple podria ser el bloc de notes.

En canvi hi ha d'altres aplicacions que necessiten recursos externs, per exemple una base de dades compartida per tots els ordinadors on s'executi aquesta aplicació. En aquest cas segurament tendrem que fer dos programes:

1. El servidor, que interactua amb l'altre programa, rep les seves peticions i li envia les respostes.

2. El client, que interactua amb l'usuari i segons les seves accions envia peticions al servidor i mostra les respostes del servidor a l'usuari.

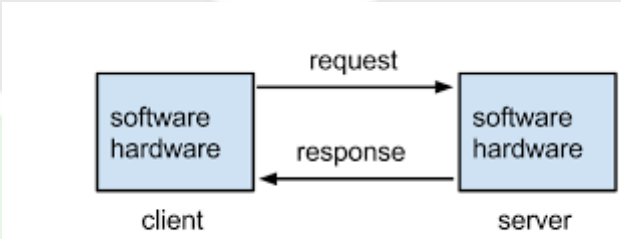

Figura 1: Autor: Lubaochuan

En un cas més general, el client no té perquè interactuar amb l'usuari, pot ser a la vegada un servidor que respon a peticions d'un altre client, ...

Actualment la majoria d'aplicacions utilitzen la web com a mitjà de connexió entre el client i el servidor. En aquest cicle ens ocuparem d'elles i en aquest mòdul en concret ens ocuparem de la programació d'aquestes aplicacions al costat del client, al navegador web.

Arquitectura client – servidor

Un parell de definicions per poder parlar amb propietat i que a més de ser uns experts o sembli.

- **Arquitectura**: una arquitectura de programari determina com s'identifiquen els components d'un sistema, com aquests components interactuen i quins protocols s'utilitzen per a la comunicació entre aquests components.
- **Protocol**: conjunt de regles i formats per establir com es comuniquen dues màquines entre si. En el cas de les aplicacions web utilitzen el protocol *http*.
- **Arquitectura client servidor**: el web és un exemple d'aquesta arquitectura:
  - **Servidor**: procés (o màquina) que proveeix un servei. Normalment espera una petició de servei, la resol i torna la resposta a qui l'ha sol·licitada. Per exemple un servidor web que espera peticions per servir les webs que allotja.
  - **Client**: procés (o màquina) que fa peticions de servei. Per exemple un navegador web que a través de les url demana recursos als servidors web.
- **Arquitectura multinivell**: la funcionalitat de l'aplicació està distribuïda entre diverses plataformes o ordinadors. Les més habituals són les de dos o tres nivells.

WWW o web

Encara que moltes vegades es prenguin per sinònims, no és el mateix internet que la web o www.

- **internet**: xarxa d'ordinadors basada en els protocols TCP/IP. Aquests protocols s'ocupen de connectar dos ordinadors a través de la xarxa i permetre que intercanviïn informació entre ells.
- **www**: Servei sobre internet, format per documents enllaçats entre si. Es basa en tres conceptes principals:
  - **HTTP (Hiper Text Transfer Protocol)**: Protocol de la capa d'aplicació utilitzat en les transaccions del www. És el que encapsula les peticions dels clients i les respostes dels servidors de manera que tant un com l'altra les puguin interpretar. Només es capaç de transportat text, no informació binària.
  - **HTML (Hiper Text MarkUp Language)**: Llenguatge de marques utilitzat per a escriure les pàgines web. Defineix l'estructura i el contingut del document i proporciona enllaços a d'altres pàgines.
  - **URI (Uniform Resource Identifier)**: Cadena de text curta que identifica unívocament un recurs a la xarxa, fins a una part del recurs. Cada pàgina, cada imatge de la web té un identificador únic.

    Per exemple, www.paucasesnovescifp.cat/logo.svg

Aplicacions basades en el web

Una aplicació client servidor ha de permetre comunicar-se el client i el servidor entre ells. Hi ha aplicacions que desenvolupen el seu propi protocol, però és més senzill utilitzar-ne un de ja existent. Un cas particular d'aplicacions client-servidor són les aplicacions que s'executen aprofitant l'arquitectura del web.

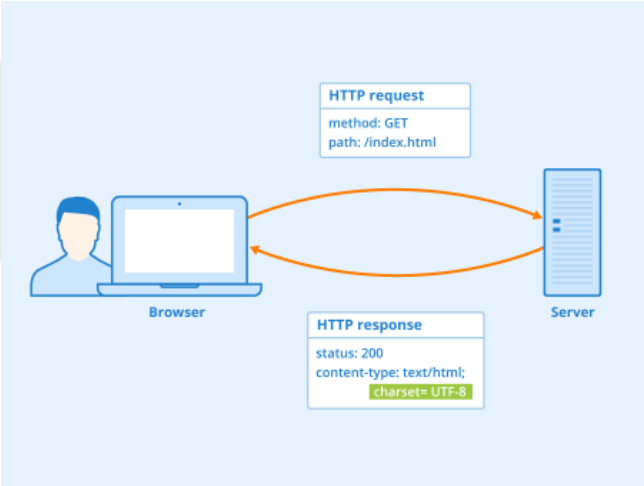

Figura 2: Author: Seobility - License: CC BY-SA 4.0

Aquestes aplicacions es basen en el fet de tenir part de la capacitat de processament en un servidor web (o conjunt de servidors) als quals s'accedeix des d'un navegador web. Quan l'usuari fa clic sobre un enllaç de la pàgina web que té al seu navegador, es genera una petició al servidor que conté la informació. El servidor, en rebre la petició, retorna la pàgina demanada si la petició era a una pàgina, o retorna el resultat d'executar una aplicació si l'enllaç corresponia a un codi a executar (per exemple, un Servlet, codi php, python, ...).

El navegador web proporciona a l'aplicació un entorn on presentar la informació. La comunicació entre client i servidor es fa utilitzant el protocol HTTP. El resultat que retorna el servidor al client s'envia en format HTML, de manera que per al client web és totalment transparent si accedeix a una pàgina web o a una aplicació que genera un resultat formatat en HTML.

Amb l'aparició de la programació asíncrona el client pot fer peticions al servidor demanant informació. Quan la rep, modifica la pagina de manera que l'usuari pugui veure el resultat. Les respostes a aquestes peticions no són HTML, sinó en algun llenguatge d'intercanvi d'informació com XML o, cada vegada més, JSON.

Les aplicacions basades en el web tenen l'avantatge que són accessibles des de qualsevol ordinador que disposi d'un navegador (la pràctica totalitat dels ordinadors connectats avui en dia a Internet) sense haver de tenir res més instal·lat a l'ordinador local.

L'ús d'aquestes arquitectures també facilita el disseny de les aplicacions, ja que no cal implementar la comunicació entre el client i el servidor (s'utilitza el protocol HTTP); i la part de presentació de l'aplicació es facilita molt pel fet de formatar el document en HTML i aprofitar les funcionalitats que proporciona el navegador (com l'intèrpret de javascript).

Mecanismes d'execució de codi en un navegador web

Inicialment el web era estàtic, és a dir, les pàgines no oferien interacció a l'usuari. Aquest només podia llegir-les i navegar entre elles utilitzant els enllaços que hi pogués haver. Les pàgines havien d'estar escrites prèviament. Aviat es va veure que no era suficient.

Una primera aproximació va ser elaborar dinàmicament les pàgines al servidor. Per exemple, per mostrar informació des d'una base de dades. El navegador és limitava a fer peticions al servidor i aquest construïa les pàgines amb la resposta i les torna va al navegador. Això tenia un cost en temps i diners (les tarifes planes no eren massa habituals en aquell temps) .

Per exemple, per incloure un conversor de temperatures a una pàgina es feia un formulari. Quan l'usuari pitjava el botó per fer el càlcul s'enviava al servidor i aquest tornava una pàgina nova amb el resultat.

Aviat es va pensar que es podria realitzar part d'aquest càlcul al client i varen sorgir dues línies d'actuació:

- Incloure d'alguna manera petites aplicacions dins la pàgina web, amb el seu propi entorn gràfic i d'execució, els **applets java**.
- Incloure a les pàgines qualque tipus de llenguatge que els navegadors poguessin interpretar i que pogués interactuar amb els altres elements de la pàgina, **els llenguatges de guions o scripts**.

Applets

Com hem dit abans els applets són "petites" aplicacions que s'incrusten dins una pàgina web. Estan escrites en Java i ofereixen pràcticament la mateixa funcionalitat que pugui tenir una aplicació d'escriptori.

Avantatges:

- Ofereixen una gran potència de càlcul.
- Poden disposar d'una interfície gràfica com la de les aplicacions d'escriptori.

Inconvenients:

- Necessiten la instal·lació del connector de Java a la màquina del client.
- Son pesats, per la descàrrega i per el temps d'inicialització que necessiten
- Tenen limitacions importants per motius de seguretat:
- No poden accedir a recursos del client com el sistema d'arxius locals, fitxers executables, el porta-retalls, ...
- No poden connectar-se a servidors que no siguin aquell del que els hem descarregat
- No poden carregar llibreries en codi natiu de la màquina client.
- No poden accedir a segons quines propietats del client
- ...

Per raons de seguretat els navegadors cada vegada han anat posant més pegues als applets. Avui en dia són pràcticament inexistents.

Scripts

L'altra mecanisme d'execució de codi al client són els guions, o scripts. Un script és un codi normalment interpretat, guardat dins un fitxer de text pla.

A la programació web s'entén per script una peça de codi inclosa dins l'HTML, escrita en un altre llenguatge. Depenent d'on s'executarà aquest guió podem distingir entre:

- **Scripts de servidor**: Els interpreta el servidor abans de lliurar la resposta al client. Pel client son transparents, mai li haurien d'arribar. Per exemple, PHP, ASP, JSP, ...
- **Scripts de client**: Aquests scripts han de ser executats per el client, pel navegador. Permeten modificar la pàgina HTML, interactuar amb l'usuari, ... Exemples d'aquest tipus de guions poden ser: JavaScript, Jscript, VBScript, ...

Formes d'inclusió de guions als documents HTML

Els guions del costat del client poden estar inclosos dins el document HTML, *embedded scripts*, però també els podem trobar en fitxers separats, en tal cas els anomenarem scripts externs, inclosos al document per una etiqueta.

Utilitzarem el primer tipus, els guions incrustats, quan el codi que inclouen es molt particular d'aquella pàgina i no el reutilitzarem, i, així i tot, si és de dimensions reduïdes.

```
<script type="text/javascript">
        document.write("<p>Contingut afegit</p>");
</script>
```

En canvi els external scripts ens donen la possibilitat de reutilitzar el codi en altres pàgines i a més ajuden a reduir el temps de descàrrega.

```
<script type="text/javascript" src="/js/codigo.js"></script>
```

El contingut que hi pugui haver dins l'element s'ignora. Tant en un cas com en l'altra, el guió s'inclou a la pàgina utilitzant l'etiqueta <script>

Un cas especial de guió incrustat és el d'aquell codi que s'inclou dins d'un altre element, normalment per programar una resposta a qualque esdeveniment. **Prohibit!**

```
<p onclick="alert('Hola!!!')">Un paràgraf simpàtic</p>
```

Possibilitats i limitacions

JavaScript es va fer molt aviat molt popular, ja que obria un munt de noves possibilitats als programadors web. L'aparició de Flash li va suposar un gran cop, ja que permetia realitzar algunes accions totalment impossibles per JavaScript. Però la truita va girar amb l'aparició d'AJAX i la possibilitat de comunicar asíncronament el client amb el servidor.

JavaScript va ser dissenyat de manera que s'executàs en un entorn molt limitat que permetés als usuaris confiar en l'execució dels scripts:

- Els scripts no es poden comunicar amb recursos que no pertanyin al mateix domini d'on es va descarregar l'script.
- No poden tancar finestres que no hagin obert.
- Les finestres que es creen no poden ser massa grosses ni massa petites ni crear-se fora de la vista de l'usuari.
- No poden accedir als arxius de l'ordinador de l'usuari ni llegir o modificar les preferències del navegador.
- Si l'execució d'un script consumeix massa recursos el navegador dóna l'opció a l'usuari de tancar-lo.

Existeixen alternatives per a saltar-se algunes d'aquestes restriccions, principalment signar digitalment els scripts i sol·licitar permís a l'usuari.

Pel que fa a les possibilitats que ofereix Javascript es poden resumir amb una: dotar de comportament als documents HTML. És a dir, fer que els documents HTML puguin interaccionar amb l'usuari:

- Respondre als esdeveniments: pitjar botons, pasar sobre objectes, ...
- Modificar el contingut de la pàgina: Crear, eliminar elements, modificar el seu contingut, modificar la seva aparença, ...
- Fer peticions al servidor i processar la seva resposta sense haver de recarregar la pàgina.
- ...

Podem dir que Javascript és un dels tres pilars en els que es basa la part de client d'una aplicació web:

- HTML: Per el contingut i l'estructura del document.
- CSS: Per la seva aparença.
- Javascript: Per la interacció amb l'usuari.

Entorn de desenvolupament

JavaScript és un llenguatge de guions interpretat. L'únic que necessitam per escriure el codi font és un editor que pugui generar fitxers en text pla. L'únic que necessitam per executar-lo és insertar-lo dins d'un document HTML i obrir aquest document des del navegador.

En principi no es necessari tenir cap servidor web instal·lat, encara que sempre és recomanable executar l'aplicació web a través d'un d'ells. Així evitam sobre tot errades amb les rutes dels recursos de l'aplicació, ...

Encara que sigui possible desenvolupar una aplicació web completa amb el bloc de notes, jo recoman utilitzar algun entorn de desenvolupament. N'hi ha bastants de gratuïts molt complets. Per exemple, netbeans, visual studio code, eclipse, atom ide, ...

Pel que fa a la depuració del codi, fins fa poc era un tema un mica complicat i els programadors acabaven fent el debug mostrant missatgets així com el codi s'anava executant. Afortunadament per voltros les coses han anat canviant i els navegadors proporcionen eines per a millorar la vida dels programadors. Per exemple, Chrome, Edge i

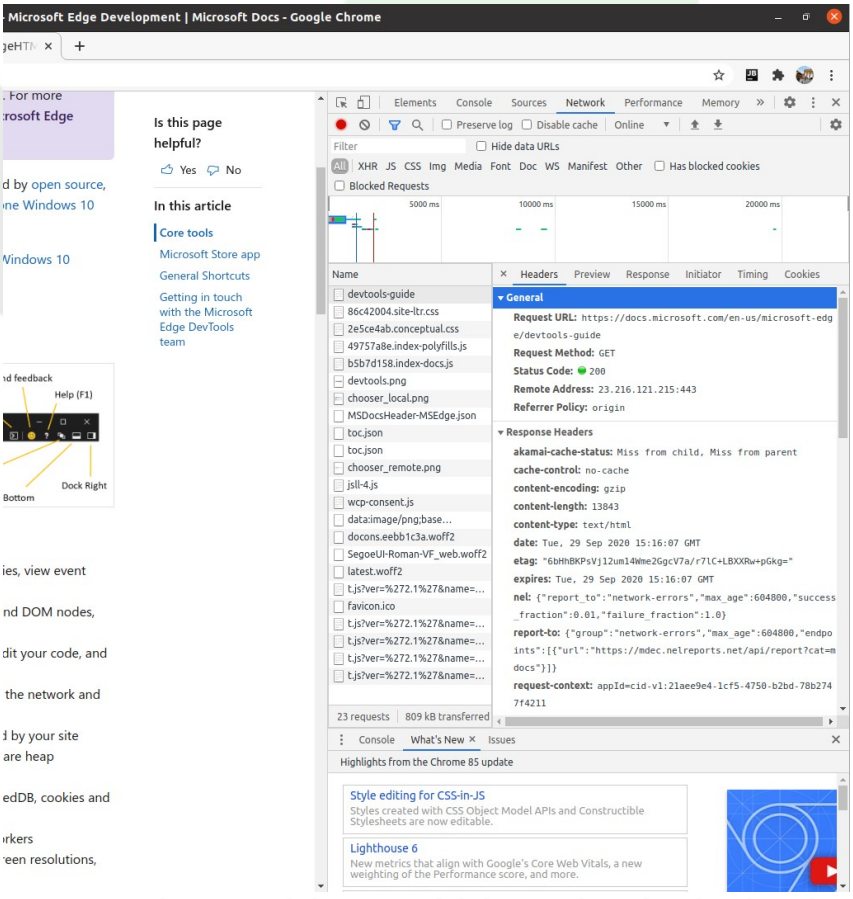

Figura 3: Chrome amb les eines del desenvolupador desplegades (F12)

Firefox ofereixen les Eines per a desenvolupadors, que inclouen visualització de variables, d'elements HTML, execució de codi pas a pas ...

Per la seva banda, els IDEs incorporen nombroses eines: molts d'ells tenen funcions de LiveEdit, és a dir, mentre s'escriu el codi es pot veure simultàniament el resultat al navegador, depuració del codi des del propi IDE, servidor web local a través del qual s'executen els documents HTML, ...