

[laboratori] Lab 03: Introducció a tecnologies Web. JavaScript, AJAX i JSON

Lab 03: Introducció a tecnologies Web. JavaScript, AJAX i JSON

Introducció

Continuarem aprenent noves formes de comunicació client/servidor amb [HTTP](#). Més concretament, en el costat del servidor tindrem de nou un [Java Servlet](#) executant-se en un servidor Apache Tomcat. A la part client, un script en JavaScript, valgui la redundància, executant-se en el nostre navegador. Aquest darrer farà les peticions al servidor via l'objecte XMLHttpRequest ([AJAX](#)). Les dades que s'intercanviaran estaran representades en [JSON](#).

Tasques a Realitzar

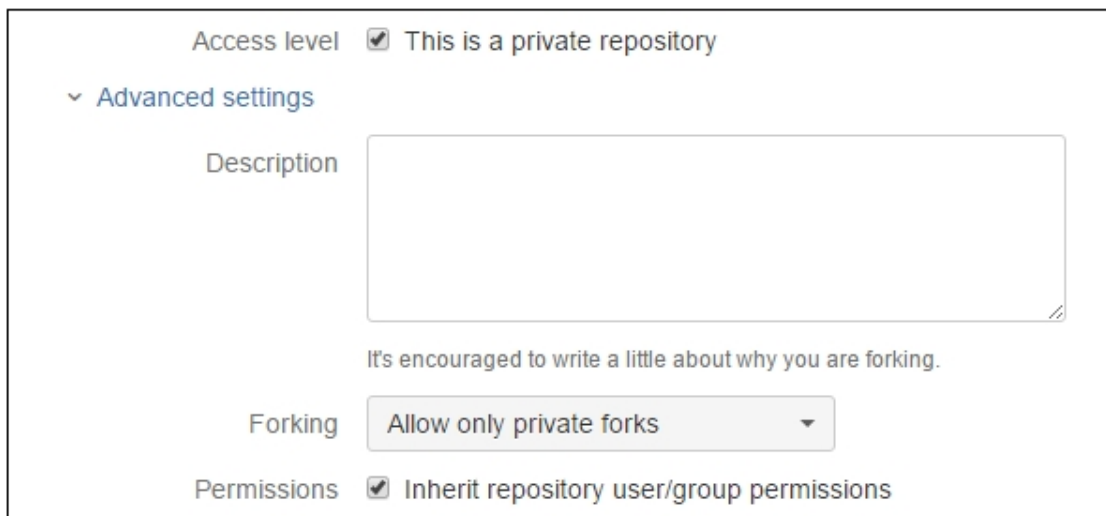
Prèvia

Se suposa que ja heu entrat als PCs escollint LINUX com a Sistema Operatiu. Les tasques s'han de dur a terme mantenint l'ordre seqüencial amb què són descrites a continuació.

Tasca #1 (15 punts si es completa abans de la fi de la sessió de laboratori)

1.1 Fork & Clone del repositori "waslab03"

1. Aneu al repositori https://bitbucket.org/fib_was/waslab03 i feu-ne un "Fork". Un dels dos membres de la parella s'haurà de "loguejar" amb les seves credencials de bitbucket.org. No cal que canvieu el nom del vostre repositori però sí que es necessari que les opcions **"This is a private repository"** (Access Level) i **"Inherit repository user/group permissions"** (Permissions) **estiguin seleccionades** tal com es mostra a la figura:



The image shows a screenshot of the Bitbucket repository settings page. At the top, there is a section for 'Access level' with a checked checkbox labeled 'This is a private repository'. Below this is a section titled 'Advanced settings' with a dropdown arrow. Under 'Advanced settings', there is a 'Description' label followed by a large text input field. Below the input field is a note: 'It's encouraged to write a little about why you are forking.' Below this is a 'Forking' section with a dropdown menu currently set to 'Allow only private forks'. At the bottom, there is a 'Permissions' section with a checked checkbox labeled 'Inherit repository user/group permissions'.

2. A la pàgina del vostre repositori nou, copieu l'URI que hi ha al costat del camp HTTPS
(https://el_vostre_username@bitbucket.org/el_vostre_username/waslab03.git)
3. A Eclipse, a la pestanya "Git repositories" seleccioneu l'opció "Clone a Git repository". A la finestra emergent copieu-hi l'URI del vostre repo, poseu-hi el vostre password a bitbucket.org i feu "Next >". A la següent pantalla no toqueu res i feu "Next >". I a l'altra pantalla seleccioneu l'opció "Import all existing projects ..." i feu "Finish". Us haureu d'esperar una estoneta i finalment veureu que al pane de l'esquerra ("Project Explorer") hi teniu un nou projectes: "waslab03 [waslab03 master]".

1.2 Posada en marxa de la web app "Wall of Tweets 3"

Només cal que cliqueu amb el botó dret del ratolí a sobre del projecte "waslab03 [waslab03 master]" i seleccioneu Run As>Run on Server. En principi us hauria de sortir com a preseleccionat el "Tomcat v7.0 Server at localhost". Llavors simplement premeu Finish. Veureu que en el pane central d'Eclipse s'obre una pestanya "Wall of Tweets 3" mostrant la pàgina web de l'aplicació. També la podeu veure des del vostre navegador si aneu a <http://localhost:8080/waslab03>. Hauríeu de veure quelcom semblant a [capture01.html](#).

1.3 Primer Commit

1. Aneu a la pestanya "Git repositories" i desplegueu el vostre repositori "waslab03 [master]". Dins de "Working Directory" hi ha el fitxer README.md. Feu-hi doble click per editar-lo i escriure-hi els vostres noms. Llavors salveu-lo.
2. De nou a la pestanya "Git repositories", cliqueu amb el botó dret del ratolí a sobre l'arrel del vostre repositori "waslab03 [master]" i seleccioneu "Commit ...". A la finestra emergent seleccioneu el fitxer README.md. On diu "Commit message" escriviu-hi "**tasca #1 finalitzada**" i premeu el botó "**Commit and Push**" (així també s'actualitzarà el repositori que teniu bitbucket.org)

Tasca #2 (20 punts si es completa abans de la fi de la sessió de laboratori)

Quan des del navegador demanem <http://localhost:8080/waslab03/>, el servidor ens retorna un html sense pràcticament cap mena de contingut, <http://localhost:8080/waslab03/index.html>, i que té associat un full d'estil, wallstyle.css i un fitxer amb codi JavaScript, wall-01.js. Aquest codi JavaScript serà l'encarregat d'interaccionar amb el servidor per obtenir-ne la informació necessària (tweets) i, fins i tot, actualitzar-la sense haver de recarregar la pàgina sencera.

Tanmateix, ara per ara, la informació sobre el tweets publicats a "Wall of Tweets 3" es mostra com un text pla:

```
[{"date":1457172961000,"author":"Sherlock","id":6,"text":"For every complex problem there is an answer that is clear, simple, and wrong.","likes":5}, {"date":1457168771000,"author":"Mycroft","id":5,"text":"A judge is a law student who marks his own examination papers.","likes":5}, {"date":1457084791000,"author":"Mycroft","id":4,"text":"Adultery is the application of democracy to love.","likes":4}, {"date":1457028431000,"author":"Sherlock","id":3,"text":"Before a man speaks it is always safe to assume that he is a fool. After he speaks, it is seldom necessary to assume it.","likes":8}, {"date":1457022225000,"author":"Mycroft","id":2,"text":"No married man is genuinely happy if he has to drink worse whisky than he used to drink when he was single.","likes":6}, {"date":1456926971000,"author":"Sherlock","id":1,"text":"A cynic is a man who, when he smells flowers, looks around for a coffin.","likes":5}]
```

La "responsable" d'això és la funció `getTweets` del fitxer `wall-01.js` (`waslab03>WebContent>wall-01.js`), i més concretament les línies [65-69](#): un cop rep la resposta de la seva petició HTTP GET a "<http://localhost:8080/waslab03/tweets>", aquesta funció escriu el text retornat (`req.responseText`) a la pàgina principal de "Wall of Tweets 3" tal qual. Ara bé, aquest text retornat no és altra cosa que un string JSON que codifica un array d'objectes JavaScript, on cadascun d'aquest objectes és un tweet amb els seus corresponents atributs. Afortunadament, ja tenim implementada una funció JavaScript, `getTweetHTML(tweet, action)` - `wall-01.js`, línia 53 - que, passant-li un tweet, retorna un text html més adient per ser mostrat per pantalla. Per tant, la vostra tasca consistirà en modificar el codi de la funció `getTweets` per tal de:

1. Transformar l'string JSON retornat a `req.responseText` en un array d'objectes utilitzant l'operació [JSON.parse](#).
2. Per a cada tweet tt inclòs en aquest array, invocar l'operació `getTweetHTML(tt, "like")`, de tal manera que els strings HTML obtinguts d'aquesta manera es vagin escrivint de manera acumulativa al seu lloc corresponent: `document.getElementById("tweet_list").innerHTML`

Si tot va bé, hauríeu d'obtenir quelcom semblant a [capture02.html](#). Llavors ja podreu fer el **Segon Commit**. Torneu a la pestanya "Git repositories", cliqueu amb el botó dret del ratolí a sobre l'arrel del vostre repositori "`waslab03 [master]`" i seleccioneu "`Commit ...`". Seleccioneu el fitxer que heu modificat: `waslab03/WebContent/wall-01.js`. On diu "Commit message" escriviu-hi "**tasca #2 finalitzada**" i premeu el botó "Commit and Push".

Tasca #3 (20 punts si es completa abans de la fi de la sessió de laboratori)

Ara volem implementar la funcionalitat d'afegir nous tweets utilitzant el formulari que la pàgina principal proporciona per a aquest propòsit. Per fer-ho caldrà modificar l'script `wall-01.js` i el servlet `WallServlet.java`.

wall-01.js (waslab03>WebContent>wall-01.js):

Quan es prem el botó "Tweet!" es crida la funció `tweetHandler()`, que caldrà modificar. Es tracta d'utilitzar l'objecte `XMLHttpRequest` per tal de fer una petició HTTP POST al `WallServlet.java`. Comenceu copiant el codi de les [linies 35-42 de la funció `likeHandler`](#) i apliqueu-hi els canvis següents:

1. Em comptes de la variable `uri`, poseu la variable global `tweetsURI` (és a dir, fareu la petició POST a `http://localhost:8080/waslab03/tweets`).
2. Feu `req.setRequestHeader("Content-Type","application/json");` abans de `req.send(...)`
3. `req.send()` no pot tenir null com a paràmetre sinó que ha de ser un string JSON. Aquest string JSON l'obtindreu invocant l'operació [JSON.stringify](#), passant-li com a paràmetre un objecte JavaScript que tindrà, com a atributs, l'autor i el text del tweet que voleu afegir (mireu el codi d'exemple de [JSON.stringify](#), línies 5 o 10).
4. Finalment, modifiqueu el codi de la funció assignada a `req.onreadystatechange` (i que es crida quan el servidor retorna la resposta) per tal de 1) obtenir, via `JSON.parse` de `req.responseText`, el nou tweet `nt` que ha generat `WallServlet.java`; 2) cridar `getTweetHTML(nt, "delete");` i, 3), escriure l'html obtingut al principi de tot de la `tweet_list`.

WallServlet.java (waslab03>Java Resources>src>wallOfTweets>WallServlet.java):

Modifiqueu el mètode `doPost` tal com s'explica a continuació i a partir del lloc indicat en el mateix codi ([linia 63](#)). La variable string `body` ja conté l'string JSON que haurà enviat la funció JavaScript `tweetHandler`. A partir d'aquí, creeu un objecte [JSONObject](#), passant aquest `body` com a paràmetre. Un cop aquest objecte ja està creat, utilitzeu l'operació [getString](#) sobre aquest objecte per tal d'extreure'n els paràmetres necessaris (`author` i `text`). Llavors, ja podeu cridar l'operació `Database.insertTweet(author, text)`, ja implementada. Aquesta operació retorna un objecte `Tweet`, corresponent al nou tweet generat per la base de dades. Passant aquest mateix objecte `Tweet` com a paràmetre, creeu un segon objecte `JSONObject`. Finalment, crideu l'operació `toString` sobre aquest segon `JSONObject` per tal de

generar l'string JSON que s'ha de retornar com a resposta al client JavaScript. Per generar aquesta resposta, fixeu-vos en la [linia 36 del mètode doGet](#) per veure com es fa.

Un cop ja pugueu inserir tweets des del navegador, procediu a fer el **Tercer Commit**. Torneu a la pestanya "Git repositories", cliqueu amb el botó dret del ratolí a sobre l'arrel del vostre repositori "waslab03 [master]" i seleccioneu "Commit ...". Seleccioneu els dos fitxers que heu modificat: waslab03/WebContent/wall-01.js i waslab03/src/wallOfTweets/WallServlet.java. On diu "Commit message" escriviu-hi "**tasca #3 finalitzada**" i premeu el botó "Commit and Push".

Tasca #4 (20 punts si es completa abans de la fi de la sessió de laboratori)

Com podeu observar, els nous tweets que s'afegeixen a la pàgina a partir del formulari tenen un botó "delete". Ara es tracta, doncs, d'implementar aquesta funcionalitat. De nou, caldrà modificar l'script wall-01.js i el servlet WallServlet.java.

wall-01.js (waslab03>WebContent>wall-01.js):

Caldrà modificar la implementació de la funció deleteHandler(). Ara es tracta d'utilitzar l'objecte XMLHttpRequest per tal de fer una petició HTTP DELETE al WallServlet.java. Torneu a copiar les [linies 35-42 de la funció likeHandler](#) i apliqueu-hi els canvis següents:

1. Substituiu "POST" per "DELETE".
2. Feu que uri = tweetsURI+ "/" + tweetID (és a dir, fareu la petició DELETE a http://localhost:8080/waslab03/tweets/xx per esborrar el tweet amb id = xx).
3. La funció assignada a req.onreadystatechange (i que es crida quan el servidor retorna la resposta) ha d'esborrar el tweet de la pàgina HTML. Per esborrar un element d'una pàgina HTML cal utilitzar la sentència element.parentNode.removeChild(element);. Aquest element l'haureu obtingut prèviament a partir de document.getElementById("tweet_"+tweetID).

WallServlet.java (waslab03>Java Resources>src>wallOfTweets>WallServlet.java):

Modifiqueu el mètode `doDelete` tal com s'explica a continuació. Obtingueu l'id del tweet que s'ha d'esborrar a partir de la `RequestURI` de manera semblant a com el mètode `doPost` obté l'id `xx` del tweet quan es fa un POST `http://localhost:8080/waslab03/tweets/xx/likes`. Un cop teniu aquest valor, ja podeu cridar l'operació `Database.deleteTweet(id)`, ja implementada. No cal retornar cap resultat al client. Si hagués alguna mena d'error (al parsejar la `RequestURI` o si l'operació `Database.deleteTweet` retornés fals) llavors caldria llançar una `ServletException`.

Un cop ja pugueu esborrar tweets des del navegador, procediu a fer el **Quart Commit**. Torneu a la pestanya "Git repositories", cliqueu amb el botó dret del ratolí a sobre l'arrel del vostre repositori "waslab03 [master]" i seleccioneu "Commit ...". Seleccioneu els dos fitxers que heu modificat: `waslab03/WebContent/wall-01.js` i `waslab03/src/wallOfTweets/WallServlet.java`. On diu "Commit message" escriviu-hi "**tasca #4 finalitzada**" i premeu el botó "Commit and Push".

Tasca #5 (Fins a 25 punts si es completa com a molt tard a les 22:00 del dilluns 14/03/2016)

Els tweets que tenen el botó "delete" són aquells que es creen des de la mateixa pàgina. Si després d'afegir un nou tweet feu un reload de la pàgina, veureu que el botó "delete" es converteix en "like" i aquell tweet ja no es pot esborrar des de la pàgina. Això és així perquè a la Tasca #2 heu programat a la funció `getTweets` perquè "pinti" els tweets d'aquesta manera.

Per una altra banda, ja heu vist a la tasca anterior que per esborrar un tweet només cal saber l'id del tweet en qüestió. Per tant, és fàcil implementar o utilitzar un client REST, per exemple, que faci peticions `delete` que esborrin qualsevol dels tweets de l'aplicació, independentment de qui els hagi creat.

Per tal de poder esborrar els tweets després de fer un reload de la pàgina i, inclús, després de tancar i obrir el navegador, i, alhora, evitar que cap altre client pugui esborrar tweets que no hagi creat ell prèviament, modificareu

degudament la vostra implementació seguint una estratègia similar a la utilitzada a la tasca #5 del primer laboratori. Però aquest cop no utilitzareu cookies sinó el [localStorage](#) del vostre navegador. La idea és la següent:

1. Quan es creï un nou tweet,
 - WallServlet.java afegirà un nou atribut "token" al tweet (JSON) que retorna al navegador. El valor d'aquest atribut no serà altra cosa que un contingut **encriptat** generat a partir de l'id del tweet en qüestió, utilitzant una funció de hash criptogràfica (MD5, SHA-256, ...).
 - wall-01.js es guardarà l'id i el token del nou tweet al localStorage.
2. Quan wall-01.js hagi de mostrar tots els tweets per pantalla, aquells tals que el seu id estigui al localStorage apareixeran amb el botó "delete". La resta, amb "like".
3. Quan s'esborri un tweet,
 - wall-01.js inclourà també el token, emmagatzemat fins aleshores al localStorage, a la petició DELETE.
 - WallServlet.java comprovarà que el token hi sigui i sigui el correcte abans d'esborrar el tweet de la BD.
 - si tot va bé, wall-01.js esborrarà també la info del tweet del localStorage.

Commits. En aquesta darrera tasca no us limiteu a fer un únic "Commit & Push" final, sinó, al contrari, heu de fer "Commit & Push" cada cop que modifiqueu parts de codi, tot i que no estigueu segurs de si aquell canvi funciona o no. Això em permetrà a mi de disposar d'una traçabilitat de com heu anat desenvolupant aquesta tasca.

Lliurament de la Pràctica

Si heu creat el vostre repositori de bitbucket.org tal com s'indicava a l'apartat 1.1, jo també hi tinc accés. Per tant, el repositori en sí és el vostre "lliurament". Això sí, **aneu comprovant de tant en quant que els vostres commits es propaguen correctament al vostre repositori de bitbucket.org**.

Avaluació

Al principi de cada tasca ja s'especifica quants punts val si es realitza en el termini establert. En el cas de les tasques #1-4, si no s'acaben dins de la sessió de laboratori, es poden finalitzar amb posterioritat però, en aquest cas, **valdran 5 punts** cadascuna.