Lezione 17 - Javascript

Javascript è un linguaggio di scripting con sintassi simile a Java, da cui deriva il nome, che gira nel processo del browser (e quindi non in una Virtual Machine esterna come le Applet). Alcuni degli oggetti predefinit su cui opera Javascript sono le "parti" del browser, e alcuni metodi agiscono come le funzionalità del browser. Per esempio, sono automaticamente definiti gli oggetti `window` (finestre del browser), `document` (il documento visualizzato), `history` (la "storia" dei siti già visitati dal browser), etc. Gli usi possibili di Javascritp sono innumenrevoli tra i quali validazione dell'input e animazione delle pagine html sono i principali.

Javascript e validazione dell'input

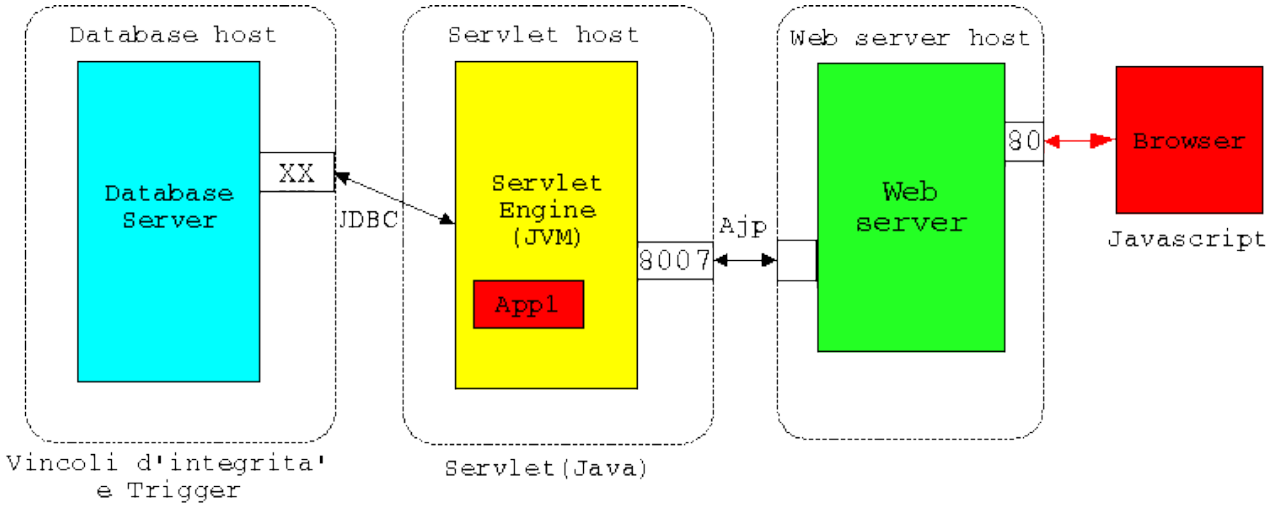
Per verificare la validita' dell'input nel form viene spesso usato javascript. In particolare esaminiamo il seguente codice:

```
<HTML>

<HEAD>
<TITLE>ESERCITAZIONE</title>
<script language="Javascript


<!--
function controlla()
{
log = document.Modulo.LOGIN.value;
if (log.length == 0)
{
alert("Inserire la LOGIN");
document.Modulo.LOGIN.select();
document.Modulo.LOGIN.focus();
return false;
}
if (log.length < 5 || log.length > 8)
{
alert("La LOGIN consiste in una stringa di almeno 5 caratteri e
massimo 8");
document.Modulo.LOGIN.select();
document.Modulo.LOGIN.focus();
return false;
}
numes = document.Modulo.NUMES.value;
if (numes.length == 0)
{
alert("Inserire il numero dell'esercitazione");
document.Modulo.NUMES.select();
document.Modulo.NUMES.focus();
return false;
}
ies = Number(numes);
if (String(ies) != numes)
{
alert("L'ESERCITAZIONE consiste di un numero");
document.Modulo.NUMES.select();
document.Modulo.NUMES.focus();
return false;
```

```
}

}
//-->

</script>
</HEAD>

<BODY bgColor="#8AC6F6">

<FORM ACTION="..." NAME="Modulo" onSubmit="return controlla();" >
<CENTER>
<BR>

LOGIN: <INPUT TYPE="TEXT" NAME="LOGIN" SIZE=8 MAXLENGHT=8 VALUE="">
NUMERO ESERCIZIO: <INPUT TYPE="TEXT" NAME="NUMES" SIZE=8 MAXLENGHT=8
VALUE=""><BR><BR>

<CENTER>
<P>
<INPUT TYPE="SUBMIT" VALUE="INVIA">
<INPUT TYPE="RESET" VALUE="CANCELLA">

</FORM>

</center>

</BODY>
</HTML>
```

Validazione input

Nello schema possiamo vedere dove intervengono i vari vincoli di integrità nel processo di elaboraione dei dati.



Validazione dei dati

	Browser	Application Server	Database
Eludibilità	sì	no	no

Risposta	immediata	intermedia	lenta
Linguaggio	Javascript	Java	SQL
Visibilità codice	sì	no	no
Computazione	distribuita		centralizzata
Internazionalizzazione	sì	sì	no

Problemi Javascript

Anche se Javascript può essere la soluzione di molti problemi è bene non abusarne perché:

1. il browser potrebbe averlo disattivato;
2. versioni diverse di browser si comportano in maniera leggermente diversa in visualizzazione, ma leggere differenze di funzionamento possono portare malfunzionamenti del codice;

Pushlet

Un uso leggermente diverso di Javascript lo vediamo tramite le Pushlet. Sappiamo che non c'é modo da parte del server di comunicare al browser un'informazione se la connessione è stata già chiusa. Sfruttano Javascript possiamo procedere nel seguente modo:

1. il browser richiede la pagina alla Servlet/JSP;
2. la Servlet/JSP fornisce la risposta con i codice HTML completo (quindi compreso il tag </HTML>);
3. la connessione viene mantenuta attiva e i dati eventualmente ancora da inviare vengono forzatamente inviati;
4. il browser è in grado di visualizzare la pagina HTML completa, ma resta ancora in attesa di dati dalla connessione;
5. la Servlet/JSP quando vuole far caricare una pagina, invia nella vecchia connessione un codice Javascript che forza il carcamento della pagina ;
6. il browser esegue il codice Javascript non appena lo riceve e in questo caso ricarica la pagina quando vuole il server.

Vediamo qui di seguito il codice completo di una servlet molto semplice che forza il ricaricamento di se stessa dopo un tempo casuale.

```
import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

import java.util.*;

public class Pushlet extends HttpServlet

{
```

```
 private int c=0;

 /**Process the HTTP Get request*/

 public void doGet(HttpServletRequest request,

          HttpServletResponse response) throws ServletException,

  IOException

 {

  response.setContentType("text/html");

  response.setBufferSize(0);

  PrintWriter out = response.getWriter();

  out.println("<html><body><b>count "+c++ +"</b></body></html>");

  out.flush();

  response.flushBuffer();

  try

  {

  Thread.sleep((int)(1000*Math.random()*10));

  }

  catch( InterruptedException e)

  {

  }

  out.println("<script language=\"JavaScript\">");

  out.println("document.location='http://localhost:8080/servlet/Pushlet';");

  out.println("</script>");

  out.close();

 }

}
```

AJAX Asincronus-JAvascript-Xml

Con Javascrit possiamo sostanzialmente simulare la comunicazione Applet/Servlet. Infatti grazie al'uso di richieste Asincrone al server scritte in javascript possiamo sostanzialmente ottenere lo stesso effetto dell'applet. In più, e questo lo possiamo fare solo con Javascript, sfruttando l'API DOM per Javascript, possiamo modificare singole parti della pagina html sostituendala con “pezzi” di pagina inviati dal server. Questi pezzi di pagina sono inviati tramite XML. In conclusione sfruttando richieste Asincone scritte in Javascript con cui otteniamo risposte in XML (AJAX), possiamo ottenere applicazioni WEB con interattività molto simile alle applicazione Desktop.

Ecco la sequenza di azioni del browser in un accesso AJAX:

1. il browser richiede la pagina html
2. il server invia la pagina (statica o dinamica)
3. il cliente visualizza la pagina, se la pagina contiene richieste AJAX queste vengono inltrate al server con un thread secondario
4. l'attività nel browser continua normalmente (nel caso inltrando altre richieste AJAX o interagendo con l'utente)
5. quando il server invia la risposta al cliente, il thread secondario visualizza il risualtato nella pagina HTML modificandola tramite il DOM.

Di solito è conveniente utilizzare una libreria che semplifica la programmazione in AJAX, in questo modo non dobbiamo occuparci delle differenze tra i vari browser e dei dettagli di più basso livello. Per esempio una libreria molto usato è Prototype che potete scaricare e trovare documentazione ed esempio al link: http://www.prototypejs.org/

Esempio:

```
<html>
<head>
<script type="text/javascript" src="prototype.js"></script>
</head>
<body>
<script language="javascript">
Event.observe(window,'load',init);

function init(){
$('greeting-submit').style.display = 'none';
Event.observe('greeting-name','key-up',greetings);
}
function greetings(){
var myAjax = new Ajax.Updater('greeting','servlet',{method:'get',parameters:{ par:
$F('greeting-name') }});
return false;
}
</script>
<form method="get" action="servlet" id="greeting-form" onSubmit="return
greetings();">
<div>
<label for ="greeting-name">Enter your name:</label>
<input id="greeting-name" name="greeting-name" type="text" />
<input id="greeting-submit" name="greeting-submit" type="submit" value="Greet me!" />
</div>
<div id="greeting"></div>
</form>
</body>
</html>
```