Struttura del compito d'esame ed esempio

Struttura del compito

- Il compito d'esame è tipicamente composto da 2 esercizi
  - Esercizio sulla programmazione client-side (Javascript e Ajax)
  - Esercizio sulla programmazione server-side (JSP e servlet)
- Spesso gli esercizi fanno capo ad un tema comune
- Il tempo a disposizione è di 3 ore
- Si può consultare il materiale didattico
- Lo scorso anno c'era anche un esercizio su DTD e XSD che quest'anno non c'è

Esercizio 1 – Programmazione client-side - 1

- Si richiede di implementare una pagina Web in tecnologia AJAX che consenta di visualizzare giorno, ora e testo dei post inseriti dall'utente in un servizio di microblogging.
- La pagina è costituita da una form contenente:
  - Un campo di testo che consente all'utente di inserire il codice identificativo di un post.
  - Un secondo campo di testo che permette di indicare quanti post devono essere mostrati a partire da quello indicato nel campo sopra descritto
  - Un bottone che consente di inviare la request AJAX al server.

Esercizio 1 – Programmazione client-side - 2

- A seguito della pressione del bottone la pagina deve:
  - Verificare che i valori inseriti dall'utente contengano solo valori numerici
  - Inviare una chiamata Ajax (GET) al server per leggere un file XML che contiene tutti i post del microblog
  - Inserire i post che corrispondono alla richiesta dell'utente in una una lista puntata, situata sotto al form, in cui vengono riportati giorno, ora e testo di ogni post
- Per semplicità assumiamo che
  - sia sempre presente sul server un post con il codice specificato
  - i dati che questi ci restituisce siano in formato XML ed organizzati come segue

Formato dei post

```
<?xml version='1.0' encoding='UTF16'?>
<MyTwitter>
  <post>
    <data>
      <giorno>giorno</giorno>
      <ora>ora</ora>
    </data>
    <testo>post</testo>
  </post>
  <post>
    <data>
      <giorno>giorno</giorno>
      <ora>ora</ora>
    </data>
    <testo>post</testo>
  </post>
  ...
<MyTwitter>
```


Esercizio 1 – Programmazione client-side - 3

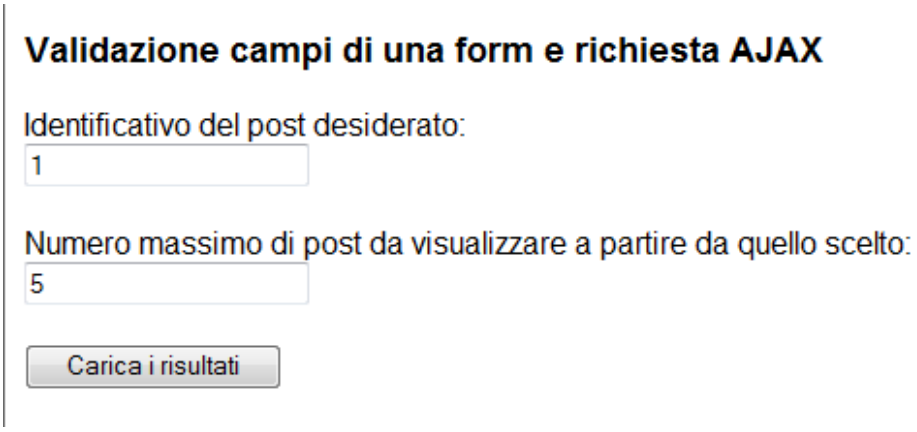
- Nel caso in cui il codice identificativo del post ed il numero massimo di post visualizzabili non siano stati immessi dall'utente in formato numerico si richiede che sia presentato un popup che segnala il problema all'utente

Esercizio 1 – Soluzione: pagina HTML

```
<html>
  <head>
    <title>Validazione campi di una form e richiesta AJAX</title>
    <meta http-equiv="Pragma" content="no-cache"/>
    <script type="text/javascript"
       src="scripts/validazione.js"></script>
    <script type="text/javascript"
       src="scripts/richiesta.js"></script>
  </head>
  <body>
    <h3>Validazione campi di una form e richiesta AJAX</h3>
    <form name="myForm">
      <p>Identificativo del post desiderato:<br>
         <input name="postId" type="text"/></p>
      <p>Numero massimo di post da visualizzare :<br>
        <input name="numberOfPosts" type="text"/></p>
      <p><input value="Carica i risultati" type="button"
        onclick="if (valida()) carica();"></p>
    </form>
    <div id="result">  </div>
  </body>
</html>
```


Pagina HTML

- La pagina appare così:



Esercizio 1 – Soluzione: documento XML (posts.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<MyTwitter>
  <post>
    <data>
       <giorno>17/3/2007</giorno>
       <ora>11:00</ora>
    </data>
    <testo>testo del post</testo>
</post>
<post>
   <data>
     <giorno>17/3/2007</giorno>
     <ora>12:00</ora>
   </data>
   <testo>un altro post</testo>
</post>
...
</MyTwitter>
```


Esercizio 1 – Soluzione: valida.js /1

```
function isBlank(txt, msg)
{
  if (txt == "")
  {
    alert(msg);
    return true;
  }
  else return false;
}

function isNumber(txt,msg)
{
  if (isBlank(txt,msg))
    return false;
  var regex = /^\d+$/;
  if (txt.search(regex) == -1 )
  {
    alert(msg);
    return false;
  }
  return true;
}
```

Esercizio 1 – Soluzione: valida.js/2

```
function valida1()
{
  var n = document.myForm.numberOfPosts;
  return isNumber(n.value,"Controlla che il numero dei post sia in
formato corretto!");
}

function valida2()
{
  var n = document.myForm.postId;
  return isNumber(n.value, "Controlla che l'id del post desiderato
sia in formato corretto!");
}

function valida()
{
  return valida1() && valida2();
}
```


Esercizio 1 – Soluzione: richiesta.js/1

```
function calcolaStop(start,posts)
{
  var num = parseInt(document.myForm.numberOfPosts.value);
  var lastPost = start + num;
  if (lastPost > posts.length)
    return posts.length;
  else return lastPost;
}

function leggiContenuto(item, nomeNodo)
{
  return
item.getElementsByTagName(nomeNodo).item(0).firstChild.nodeValue;
}
```

Esercizio 1 – Soluzione: richiesta.js/2

```
function applicaContenuto(theXML, theElement)
{
  var theHTML = "<p><ul>";
  var posts = theXML.getElementsByTagName("post");
  var itemNodes = new Array();
  var start = parseInt(document.myForm.postId.value);
  var stop = calcolaStop(start,posts);
  for (var i = 0, p = start; p < stop; i++, p++)
  {
    itemNodes[i] = new Object();
    itemNodes[i].testo = leggiContenuto(posts[p],"testo");
    itemNodes[i].giorno = leggiContenuto(posts[p],"giorno");
    itemNodes[i].ora = leggiContenuto(posts[p],"ora");
  }
  if (itemNodes.length > 0)
    for( var c = 0; c < itemNodes.length; c++ )
      theHTML += '<li>'+itemNodes[c].giorno+', '+
        itemNodes[c].ora+' = '+itemNodes[c].testo+'</li>';
  else theHTML = “Nessun post trovato.<br/>";
  theHTML += "</ul></p>";
  theElement.innerHTML = theHTML;
}
```


Esercizio 1 – Soluzione: richiesta.js/3

```
var xhr;
function callback()
{
  if (xhr.readyState == 4)
  {  var el = document.getElementById("result");
     if (xhr.status == 200 )
     { if ( xhr.responseXML )
         applicaContenuto(xhr.responseXML, el);
       else el.innerHTML = "L'XML non è valido.<br>";
     }
     else el.innerHTML = “Errore: " + xhr.statusText;
  }
}
function carica()
{
  var uri ='posts.xml';
  xhr = new XMLHttpRequest();
  xhr.onreadystatechange = callback;
  xhr.open("get", uri, true);
  xhr.setRequestHeader("connection","close");
  xhr.send(null);
}
```

Esercizio 2 – Programmazione server-side – 1

- Si richiede di implementare una sezione della applicazione Web per la gestione di un servizio di microblogging.
- In particolare si richiede che il candidato implementi due distinte JSPche consentano di inserire nuovi post (NuovoPost) e di leggere quelli precedentemente inseriti (LeggiPost).
- Le JSP hanno accesso ad un database comune costituito da una sola tabella con i campi ID, Giorno Ora, Testo
- ID è un codice numerico progressivo che identifica univocamente un post
- Giorno e Ora rappresentano rispettivamente il giorno e l'ora in cui il server ha inserito il messaggio nel data base.
- Testo rappresenta il testo del post (max 149 caratteri).

Esercizio 2 – Programmazione server-side – 2

- La JSP NuovoPost consente di inserire un nuovo post nel database.
- La JSP trasmette il testo del post tramite una GET.
- Una volta ricevuto il testo, la JSP incrementa il contatore dei messaggi ricevuti, ottiene giorno e ora e genera un record che verrà inserito nel data base.
- Il contatore verrà memorizzato nel campo ID del record, data e ora saranno memorizzati rispettivamente nei campi Giorno ed Ora e il parametro ricevuto dal client sarà memorizzato nel campo testo.
- Se l'inserimento del record avviene con successo la JSP mostra all'utente una conferma dell'avvenuto inserimento.
- In caso contrario viene presentata all'utente una pagina in cui si segnala l'errore.

Esercizio 2 – Programmazione server-side – 3

- La JSP LeggiPost consente di presentare all'utente una pagina che visualizza un insieme selezionato di post precedentemente registrati.
- A questo fine, la JSP richiede il passaggio di una richiesta HTTP GET che specifica due parametri: il codice del primo post da
- visualizzare (id_post), ed il numero dei post da visualizzare nella pagina (max_post).
- Una volta ricevuto il codice univoco del post la JSP controlla che questo sia presente nel database
- In caso positivo, presenta all'utente una pagina in cui è inclusa una tabella che include il post selezionato e al più i max_post che lo precedono.
- La relazione di precedenza è determinata sulla base dell'identificatore del post.

Esercizio 2 – Programmazione server-side – 4

- La tabella ha tre colonne: Giorno, Ora, Testo
- Ogni riga contiene le informazioni di un singolo post.
- In particolare la prima riga contiene le informazioni del post di codice id_post selezionato, mentre le righe successive presentano le informazioni sui post che lo precedono.
- Nel caso in cui non esista alcun post con il codice specificato viene mostrata all'utente una pagina che segnala l'errore.

Esercizio 2 – Struttura della soluzione

- Due pagine statiche
  - index.html pagina di partenza
  - notfound.html per gestire errori 404
- Quattro JSP
  - NuovoPost.jsp per l'inserimento di un post nel microblog
  - LeggiPost.jsp che permette di leggere un certo numero di post a partire da un ID indicato dall'utente
  - Pagine di errore: insertFailure.jsp e readFailure.jsp
- Un bean
  - Contatore.java
- Le classi per gestire il DAO
  - le omettiamo per brevità

Esercizio 2 – web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app ... >
  <display-name>Microblog</display-name>
  <!-- Context parameters -->
    <context-param>
      <param-name>daoFactory</param-name>
      <param-value>1</param-value>
    </context-param>
  <!-- Handle exceptions and errors -->
  <error-page>
    <error-code>404</error-code>
    <location>/errorpages/notfound.html</location>
  </error-page>
</web-app>
```

Esercizio 2 – index.html

```
<html>
  <head>
    <title>Microblog</title>
  </head>
  <body>
    <h3>Microblog...</h3>
    <ul>
      <li>
        <a href="NuovoPost.jsp">
          Inserimento di un nuovo post
        </a>
      </li>
      <li>
        <a href="LeggiPost.jsp">
          Lettura dei post esistenti
        </a>
      </li>
    </ul>
</body>
</html>
```


Esercizio 2 – Contatore.java

```
package bean;

public class Contatore
{
  private int count = 0;
  public void aumenta()
  {
    count++;
  }
  public int getValore()
  {
    return count;
  }
}
```

Esercizio 2 – PostDTO.java

```
package dao;

public class PostDTO
{
  private int id;
  private String giorno;
  private String ora;
  private String testo;

  public int getId() { return id; }
  public void setId(int i) {id = i; }

  public String getGiorno() { return giorno; }
  public void setGiorno(String g) {giorno = g; }
  public String getOra() { return ora; }
  public void setOra(String o) {ora = o; }
  public String getTesto() { return testo; }
  public void setTesto(String t) { testo = t; }
}
```


Esercizio 2 – NuovoPost.jsp/1

```
<%@ page import="java.text.*" %>
<%@ page import="java.util.*" %>
<%@page import="dao.*"%>

<%@ page errorPage="errorpages/insertFailure.jsp"%>

<html>
  <head>
    <title>Nuovo Post</title>

    <jsp:useBean id="contatore"
     class="bean.Contatore"
     scope="application"/>
…
```

Esercizio 2 – NuovoPost.jsp/2

```
<% boolean inserimento = false;
  String testo = request.getParameter("testo");
  if ( testo != null && ! testo.equals("") )
  {
    if ( testo.length() > 149 )
      throw new Exception("Testo troppo lungo");
    Date d = new Date();
    int n =
      Integer.parseInt(application.getInitParameter("daoFactory"));
    String giorno =
      new SimpleDateFormat("EEE, MMM d, ''yy").format(d);
    String ora = new SimpleDateFormat("h:mm a").format(d);
    PostDTO postDTO = new PostDTO();
    postDTO.setId(contatore.getValore());
    postDTO.setGiorno(giorno);
    postDTO.setOra(ora);
    postDTO.setTesto(testo);
    PostDAO postDAO = DAOFactory.getDAOFactory(n).getPostDAO();
    postDAO.createPost(postDTO);
    contatore.aumenta();
    inserimento = true;
  } %>
```


Esercizio 2 – NuovoPost.jsp/3

```
<link rel="stylesheet“type="text/css"
 href="<%= request.getContextPath()%>/styles/default.css" />
</head>
  <body>
    <h1>Inserimento</h1>
     <% if (inserimento)
       { %>
         <p>Inserimento avvenuto con successo.</p>
         <p>Clicca <a href="LeggiPost.jsp">qui</a>
         per vedere tutti i post inseriti.</p>
    <% } %>
    <form method="get">
      <table>
        <tr>
          <td>Inserisci post:</td>
          <td><input type="text" name="testo"/></td>
        </tr>
      </table>
      <input type="submit" value="invia!"/>
    </form>
  </body>
</html>
```

Esercizio 2 – LeggiPost.jsp/1

```
<%@ page import="java.text.*" %>
<%@ page import="java.util.*" %>
<%@page import="dao.*"%>
<%@ page errorPage="errorpages/readFailure.jsp"%>
<%!
  String printTableRow(PostDTO postDTO)
  {
    return "<tr><th>"+
       postDTO.getGiorno()+
       "</th><th>")+
       postDTO.getOra()+
       "</th><th>")+
       postDTO.getTesto()+
       "</th></tr>";
  }
%>
```


Esercizio 2 – LeggiPost.jsp/2

```
<html>
  <head>
    <title>Leggi Post</title>
    <link rel="stylesheet"
      href="<%= request.getContextPath() %>/styles/default.css"
      type="text/css"/>
  </head>
  <body>
    <h1>Lettura</h1>
    <div id="query">
    <form id="form">
      Inserire di seguito:<br/>
      <br/>
      Identificativo del post desiderato:<br/>
      <input name="postId" type="text"/><br/>
      <br/>
      Numero massimo di post da visualizzare:<br/>
      <input name="numberOfPosts" type="text"/><br/>
      <br/>
      <input value="leggi!" type="submit" /><br/>
    </form>
  </div>
```

Esercizio 2 – LeggiPost.jsp/3

```
<% String postId_as_string = request.getParameter("postId");
   if (postId_as_string != null && !postId_as_string.equals(""))
   {
     int pid = Integer.parseInt( postId_as_string );
     int np = Integer.parseInt(request.getParameter("numberOfPosts"));
     int n =  Integer.parseInt(application.getInitParameter("daoFactory"));
     PostDAO postDAO =
      DAOFactory.getDAOFactory(n).getPostDAO();
     PostDTO postDTO = postDAO.readPost(postId);
     if (postDTO == null )
       throw new Exception("Non esiste alcun post con l'ID indicato));
%>
  <table>
  <tr><th>Giorno</th><th>Ora</th><th>Testo</th></tr>
  <%= printTableRow(postDTO) %>
  <% for (int i=postId-1; (i >= 0 && (postId-i)<numberOfPosts); i-- )
     {
       try postDTO = postDAO.readPost(i);
       catch ( Exception e ) break;
       if (postDTO==null ) break;
      else out.print(printTableRow(postDTO));
     } %>
  </table>
<% } // end if %>
</body>
</html>
```


Esercizio 2 – readFailure.jsp

```
<%@ page contentType="text/html; charset=US-ASCII" %>
<%@ page isErrorPage="true" %>
<html>
  <head>
    <title>Errore in lettura</title>
    <link rel="stylesheet" href="..." type="text/css"/>
  </head>
  <body>
    <h1>Errore</h1>
    <p>Si sono verificati errori durante la lettura.</p>
    <p>Clicca <a href="LeggiPost.jsp">qui</a> per tornare indietro.</p>
    <p> <%= exception.toString() %> </p>
    <p>
      <b>Exception message is:</b><br/>
      <%= exception.getMessage() %>
    </p>
    <p><b>Stacktrace is:</b><br/>
    <%
      // this will send trace to the browser's screen
      exception.printStackTrace(new java.io.PrintWriter(out));
      // this will send it to the log file
      exception.printStackTrace();
    %></p>
  </body>
</html>
```