

BAB 14

AJAX

14.1 PENDAHULUAN

Hingga saat ini, aplikasi web mengikuti alur arsitektur berikut : satu – satunya cara dalam merepresentasikan content baru (sebagai contoh, dalam merespon interaksi antara user dengan halaman pada aplikasi) dilakukan dengan cara mengirimkan request kepada server dan menampilkan halaman baru pada user. Terdapat teknologi client side scripting (JavaScript, VBScript, dan sebagainya) yang memberikan gambaran pada user bahwa mereka telah membuka content baru dengan sebuah click pada tombol. Namun, teknologi scripting tersebut hanya dapat memodifikasi content yang terdapat pada sebuah halaman; teknologi tersebut juga hanya dapat bekerja terhadap informasi yang telah dikirimkan pada client.

Berbagai macam solusi telah dibuat sebelumnya, yang bertujuan untuk mengubah paradigma yang ada, namun beberapa diantaranya bersifat komersial (Microsoft memiliki sebuah solusi yang hanya berfungsi dengan Internet Explorer), sulit untuk diprogram, dan tidak dapat diakses dengan mudah.

Kemudian hadirlah AJAX sebagai penyelesaiannya.

14.2 AJAX

Menyerupai J2EE, AJAX adalah 2 hal yang digabungkan : merupakan sebuah teknologi, sekaligus arsitektir pemrograman.

14.2.1 AJAX sebagai sebuah teknologi

AJAX adalah singkatan dari “Asynchronous JavaScript and XML”, yang dibuat dari serangkaian teknologi dengan berbagai kemampuan : JavaScript, XML dan sebuah method komunikasi asinkron antara client dan server.

3 teknologi yang saling berinteraksi : JavaScript menangkap isyarat, gerak serta aksi. Sebagaimana pada situasi yang mungkin terjadi, JavaScript menggunakan jalur komunikasi pada server (object JavaScript dengan nama XMLHttpRequest) untuk memanggil method yang tersimpan pada server dan menggunakan XML sebagai mekanisme pengiriman data. Jika JavaScript pada client telah menerima respon dari server, maka JavaScript akan menggunakan kemampuannya untuk memanipulasi struktur DOM halaman untuk menambahkan content yang didapat dari server. Perubahan yang terjadi pada struktur DOM kemudian diterjemahkan oleh browser pada client, sehingga meningkatkan efek interaktivitas pada user.

14.2.2 AJAX sebagai sebuah Arsitektur

Pada awalnya AJAX dibuat sebagai sebuah rangkaian dari teknologi, namun kemudian AJAX mengalami perkembangan. Sebagai contoh, adanya aplikasi web yang tidak menggunakan XML dalam mentransfer data dari client ke server. Cara tersebut dilakukan dengan menggunakan object XMLHttpRequest.

Berdasarkan realita di atas, beberapa pihak mendefinisikan sebagai paradigma baru dalam pemrograman, disamping teknologi yang menyediakan fungsionalitas. Mari kita bahas lebih mendalam tentang arsitektur AJAX.

Pengembang telah mengembangkan web programming sebelum hadirnya AJAX : action dari user yang membutuhkan data dari server ditampilkan dalam halaman yang digunakan, dimana data request dari user dikirimkan menuju server. Setelah mengolah halaman tersebut, server menampilkan halaman baru bagi user yang mengandung hasil dari proses sebelumnya.

Permasalahan dari macam arsitektur tersebut adalah lambat dan cukup memakan waktu, terutama bila dibandingkan dengan aplikasi desktop. Aplikasi desktop mampu merespon cepat atas request dari user, aplikasi ini tidak memproses ulang masing – masing komponen interface yang akan ditampilkan sebagai respon.

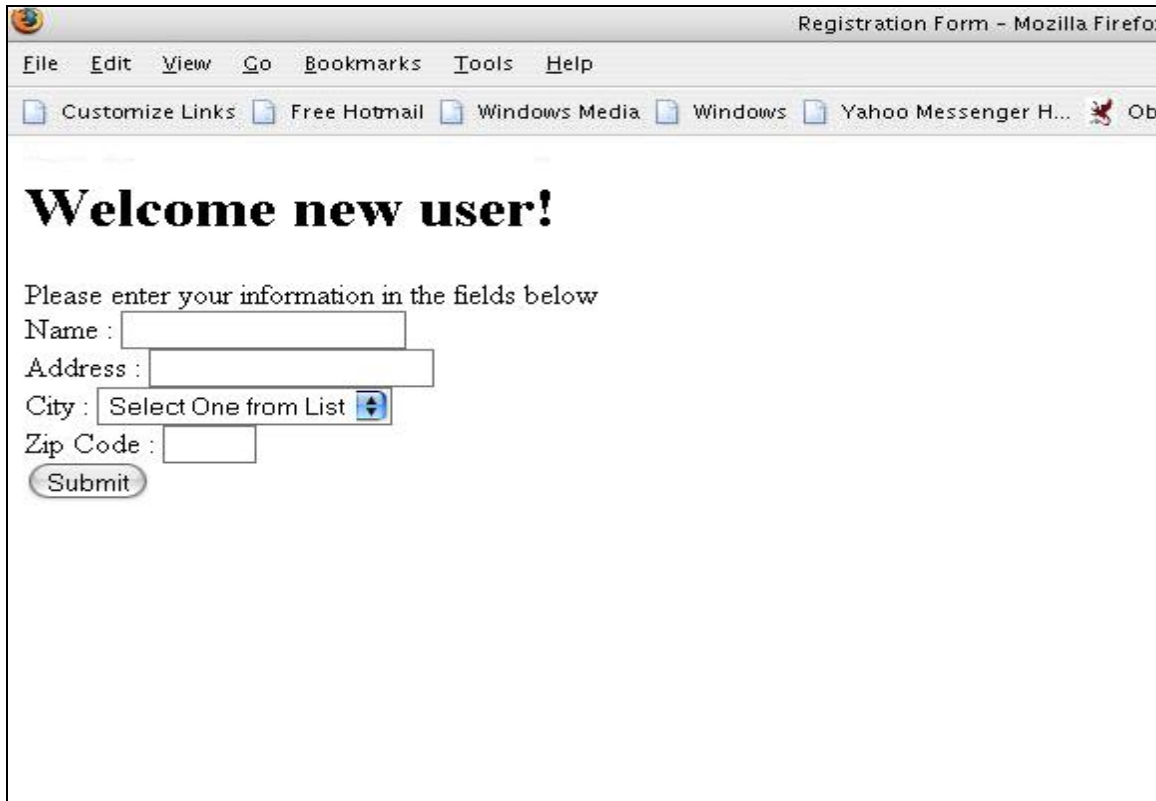
AJAX menggunakan arsitektur pemrograman tersebut pada aplikasi Web. Daripada memberikan sebuah halaman penuh pada server dan mendapatkan pula sebuah halaman penuh sebagai hasil operasi, AJAX memungkinkan kita untuk mengirimkan request dalam ukuran yang lebih kecil pada server. Halaman yang terpakai hanya termodifikasi untuk menampilkan hasil, bukan tergantikan dengan sebuah halaman baru.

Faktor penting yang lain dari arsitektur AJAX adalah request dan response dijalankan secara asinkron : AJAX tidak melarang user untuk melakukan proses lain pada halaman yang dipakai. User dapat mengisi dan menggunakan area lain pada halaman, sedangkan AJAX bekerja pada background.

Yang terakhir, AJAX memungkinkan user untuk berinteraksi dengan server sebagai respon terhadap seluruh hal yang dilakukan oleh user. Arsitektur yang ada sebelumnya hanya memungkinkan kita untuk berkomunikasi dengan server pada saat user menekan tombol atau link yang akan mengirim data pada halaman. AJAX memperbolehkan untuk me-request data baru dari server dalam bentuk mouseovers, keypress dan even lain yang dikenali oleh JavaScript.

14.2.3 Proses yang dilakukan oleh AJAX

Mari kita perhatikan contoh nyata yang menunjukkan bagaimana AJAX memperkaya interaktivitas user dan bagaimana kita menggunakannya dalam aplikasi.



The screenshot shows a Mozilla Firefox browser window titled "Registration Form - Mozilla Firefo". The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, and Help. The address bar shows several open tabs: "Customize Links", "Free Hotmail", "Windows Media", "Windows", and "Yahoo Messenger H...". The main content area displays a registration form with the heading "Welcome new user!". Below the heading, there is a prompt: "Please enter your information in the fields below". The form contains four input fields: "Name :", "Address :", "City : Select One from List" (a dropdown menu), and "Zip Code :". A "Submit" button is located below the "Zip Code" field.

```
<html>
<head>
<title>Registration Form</title>
</head>
<body>
  <H1> Welcome new user! </H1>

  Please enter your information in the fields below
  <form action="/submit">
  Name : <input type="text" name="name"/> <br/>
  Address : <input type="text" name="address"/> <br/>
  City : <select name="city">
  <option>Select One from List</option>
  <option value="Quezon City">Quezon City</option>
  </select>
  <br/>
  Zip Code : <input type="text" name="zip" size="4"/> <br/>
  <input type="submit" value="Submit"/>
  </form>
</body>
</html>
```

Contoh di atas adalah contoh sederhana dari form registrasi. Namun di sini terdapat suatu permasalahan, tidak semua user mengetahui nomor zip code.

Kita gunakan contoh diatas sebagai titik awal perjalanan kita di AJAX. Akan dikenalkan teknik penggunaan AJAX yang akan membuat halaman mengisi sendiri field zip code berdasarkan City yang dimasukkan oleh user.

14.2.4 Menyiapkan halaman form.

Pertama-tama, kita harus mempersiapkan HTML yang akan digunakan oleh user. Jika kita akan menggunakan AJAX sebagai respon user input pada field City, perlu ditambahkan sebuah event listener pada field City. Akan ditambahkan juga atribut id pada elemen form – hal ini akan memudahkan pekerjaan kita.

Menggunakan event listener pada input field City merupakan hal yang mudah : hanya perlu mendaftarkan fungsi JavaScript yang akan dipanggil pada events. Dibawah ini adalah HTML form yang disempurnakan :

```
...
    Address : <input type="text" name="address"/> <br/>
    City : <select id="city" name="city" onchange="updateZip(this.value)">
    <option>Select One from List</option>
    <option value="Quezon City">Quezon City</option>
    </select>
    <br/>
    Zip Code : <input id="city" type="text" name="zip" size="4"/> <br/>
...
```

Dengan perubahan diatas, browser akan memanggil fungsi updateZip tiap kali user memilih value dari daftar City.

14.2.5 Membuat sebuah instance object XMLHttpRequest

Sekarang saatnya menuliskan kode fungsi updateZip yang akan menangani komunikasi terhadap server dan mengupdate value zip. Sebagaimana yang dibahas sebelumnya, JavaScript mengijinkan server untuk berkomunikasi dengan menggunakan sebuah object yang dikenal dengan XMLHttpRequest.

Permasalahan berikutnya adalah membuat object tersebut. Pembuatan object ini tidak semudah menambahkan baris kode :

```
var xmlRequest = new XMLHttpRequest();
```

Kode diatas hanya dapat bekerja pada Mozilla, FireFox, Safari, Konqueror, NetScape dan Opera, namun tidak pada Internet Explorer. IE adalah browser pertama yang menggunakan fungsi semacam ini, namun dibuat dari proprietary object ActiveX

(sebagai lawan dari JavaScript object). Terlebih lagi, method yang digunakan untuk membuat XMLHttpRequest berbeda pada IE 5.0+ dibandingkan dengan versi-versi sebelumnya.

Supaya dapat digunakan oleh browser-browser yang ada, gunakan kode berikut ini :

```
function createRequestObject() {
    var xmlRequest;
    try {
        xmlRequest = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (error1) {
        try {
            xmlRequest = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (error2) {
            if (typeof XMLHttpRequest != 'undefined') {
                xmlRequest = new XMLHttpRequest();
            }
        }
    }
    return xmlRequest;
}
```

14.2.6 Menggunakan object XMLHttpRequest untuk berkomunikasi dengan Server

Jika telah membuat instance object dari XMLHttpRequest, selanjutnya object tersebut dapat digunakan untuk membuat fungsi updateZip. Pada dasarnya, kita mengirimkan request kepada server berupa input City yang diberikan user. Sebagai resource, kita dapat mengimplementasikan servlet.

```
<script type="text/javascript">
    var xmlRequest;

    function updateZip(cityValue) {
        xmlRequest = createRequestObject();
        xmlRequest.open("GET",
            "http://ourServer/ourApplication/getZIPCode?city=" +
            cityValue);
        xmlRequest.send(null);
    }
</script>
```

Terdapat beberapa cara untuk menampung respon dari server. Untuk menampung respon sebagai String sederhana, gunakan atribut responseText. Untuk menampungnya sebagai dokumen XML, gunakan atribut responseXML. Namun, value dari atribut – atribut tersebut tidak dapat diakses secara langsung.

Ingat kembali akronim dari AJAX. A berarti Asynchronous yang berarti seluruh method yang membuat request pada server tidak mengandung respon dari server

langsung setelah eksekusi method. Respon dari server akan tiba dalam waktu yang tidak dapat ditentukan.

14.2.7 Fungsi Callback

Untuk mengatasi situasi semacam ini, XMLHttpRequest mengijinkan kita untuk menggunakan fungsi JavaScript sebagai Callback handler : callback adalah fungsi yang akan digunakan oleh XMLHttpRequest setelah menerima respon dari server.

Dibawah ini adalah fungsi JavaScript yang digunakan untuk meregistrasi sebuah fungsi dengan nama processServerData sebagai fungsi callback :

```
<script type="text/javascript">
  var xmlRequest;

  function updateZip(cityValue) {
    xmlRequest = createRequestObject();
    xmlRequest.onreadystatechange=processServerData;
    xmlRequest.open("GET",
    "http://ourServer/ourApplication/getZIPCode?city=" + cityValue);
    xmlRequest.send(null);
  }
```

Pada saat menuliskan kode fungsi callback, perhatikan bahwa event onreadystatechange muncul tiap kali terdapat perubahan keadaan pada object XMLHttpRequest. Pada contoh, kita hanya ingin menangani data setelah menerima respon, maka dibuatlah statement kondisional untuk memeriksa apakah XMLHttpRequest pada kondisi yang sesuai :

```
function processServerData() {
  if (xmlRequest.readyState == 4) {
    var data = xmlRequest.responseText;
    document.getElementById("zip").value = data;
  }
}
```

Dibawah ini adalah kondisi – kondisi yang mungkin terjadi :

<i>Value readystate</i>	<i>Indikasi</i>
0	Uninitialized. Method send() belum dipanggil.
1	Loading. Request sedang dikirim menuju server.
2	Loaded. Respon dari server telah diterima.
3	Interactive. Respon sedang diproses.
4	Completed. Respon telah diproses. Siap untuk digunakan.

Hal lain yang perlu diperhatikan pada saat membuat fungsi callback adalah kemungkinan error pada server. Kode yang kita gunakan diatas akan berjalan sebagaimana fungsinya, diasumsikan jika tidak ada permasalahan dari sisi server, namun kemungkinan kesalahan selalu akan terjadi.

XMLHttpRequest menyediakan cara untuk menentukan HTTP status code yang diberikan oleh server. Jika kode menyatakan dalam kondisi 200, berarti request telah berhasil diproses :

```
function processServerData() {
    if (xmlRequest.readyState == 4) {
        if (xmlRequest.status == 200) {
            var data = xmlRequest.responseText;
            document.getElementById("zip").value = data;
            document.getElementById("zipError").innerHTML = "";
        } else {
            document.getElementById("zipError").innerHTML = "Error in
retrieving ZIP code";
        }
    }
}
```

Potongan kode JavaScript diatas menggunakan cara yang sederhana namun sangat bermanfaat untuk menampilkan pesan pada user : diasumsikan bahwa terdapat elemen HTML (<div>) dengan id dari zipError. Proses yang sukses dilakukan dari sisi server akan membiarkan elemen ini kosong. Sebaliknya, sebuah error pada server akan menampilkan pesan kesalahan sesuai yang ditentukan pada kode.

Satu hal yang patut untuk diperhatikan, atribut innerHTML didukung oleh sebagian besar, namun tidak seluruh browser. Alternatif lain yang lebih kompatibel bagi browser adalah sebagai berikut :

```
function processServerData() {
    if (xmlRequest.readyState == 4) {
        var zipErrorDiv = document.getElementById("zipError");
        if (xmlRequest.status == 200) {
            var data = xmlRequest.responseText;
            document.getElementById("zip").value = data;
            zipErrorDiv.replaceChild(document.createTextNode(""),
            zipErrorDiv.childNodes[0]);
        } else {
            zipErrorDiv.replaceChild(document.createTextNode("Error
            in retrieving ZIP code"),
            zipErrorDiv.childNodes[0]);
        }
    }
}
```

Berikut ini adalah kode lengkap dari halaman yang telah dibuat :

```
<html>
  <head>
    <title>Registration Form</title>
    <script type="text/javascript">
      var xmlRequest;
      function updateZip(cityValue) {
        xmlRequest = createRequestObject();
        xmlRequest.onreadystatechange=processServerData;
        xmlRequest.open("GET",
          "http://ourServer/ourApplication/getZIPCode?city=" + cityValue);
        xmlRequest.send(null);
      }
      function processServerData() {
        if (xmlRequest.readyState == 4) {
          if (xmlRequest.status == 200) {
            var data = xmlRequest.responseText;
            document.getElementById("zip").value = data;
            document.getElementById("zipError").innerHTML = "";
          } else {
            document.getElementById("zipError").innerHTML = "Error in
              retrieving ZIP code";
          }
        }
      }
      function createRequestObject() {
        var xmlRequest;
        try {
          xmlRequest = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (error1) {
          try {
            xmlRequest = new ActiveXObject("Microsoft.XMLHTTP");
          } catch (error2) {
            if (typeof XMLHttpRequest != 'undefined') {
              xmlRequest = new XMLHttpRequest();
            }
          }
        }
        return xmlRequest;
      }
    </script>
  </head>
  <body>
    <H1> Welcome new user! </H1>
    Please enter your information in the fields below
    <form action="/submit">
      Name : <input type="text" name="name" id="name"/> <br/>
      Address : <input type="text" name="address" id="address"/> <br/>
      City : <select id="city" name="city" onchange="updateZip(this.value)">
        <option>Select One from List</option>
        <option value="Quezon City">Quezon City</option>
      </select>
      <br/>
      Zip Code : <input type="text" name="zip" id="zip" size="4"/> <div
        id="zipError"></div>
      <br/>
      <input type="submit" value="Submit"/>
    </form>
  </body>
</html>
```

Berikut ini adalah contoh implementasi dari pemanggilan resource dari sisi server oleh AJAX :

```
public ZIPRetrievalServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
        response)
        throws ServletException, IOException {
        String city = request.getParameter("city");
        ZIPService service = new ZIPService();
        String zipCode = service.getZipForCity(city);
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(zipCode);
        out.close();
    }
}
```