XML u web aplikacijama

JS i XML (AJAX)

JavaScript, XML i AJAX

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX nije programski jezik.
- Predstavlja upotrebu objekta **XMLHttpRequest** i **JavaScripta** za **asinhroni** prikaz podataka.
- AJAX aplikacije mogu koristiti XML za prenos podataka, ali se u praksi još češće koriste obični tekstualni ili JSON formatirani podaci. Detaljnije o JSON formatiranju u narednom predavanju.
- AJAX omogćava da se web stranice ažuriraju asinhrono i to razmenom sa serverom i u pozadini. Ovo omogućava i izmenu dela stranice nezavisno od ostatka i tako značajno doprinosi efikasnosti u prikazu ali i boljem korisničkom iskustvu.

XMLHttpRequest

- Svi novi web čitači imaju ugrađeni **objekat XMLHttpRequest** kojim se zahtevaju podaci od servera.
- Objekat XMLHttpRequest obezbeđuje osnovu za razvoj modernih aplikacija. Primenom ovog objekta moguće je:
  - Ažurirati web stranicu bez ponovljenog učitavanja stranice.
  - Zahtevati dopremane podataka od server pošto je stranica učitana
  - Primiti podatke od servera pošto je stranica učitana
  - Poslati podatke serveru u pozadinskom procesu

Primer upotrebe XMLHttpRequest

```
<!DOCTYPE html>
<html>
<body>
    <h2>Primer 1: Slanje objekta XMLHttpRequest</h2>
    <div id="div1">
        <button type="button" onclick="loadXMLDoc()">Učitaj sadržaj fajla "info.txt"</button>
    </div>
    <script>
        function loadXMLDoc() {
          var xmlReq = new XMLHttpRequest();
          xmlReq.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {

document.getElementById("div1").innerHTML = this.responseText;
             }
          };
          xmlReq.open("GET", "info.txt", true);
          xmlReq.send();
        }
    </script>
</body>
</html>
```

Kreiranje objekta XMLHttpRequest

- Kreiranje ovog objekta se vrši primenom operatora new kao kod svih objekata u JavaScript-u.
-
```
var xmlReq = new XMLHttpRequest();
```
- Napomena:
- Svi noviji čitači podržavaju objekat XMLHttpRequest. Za slučaj starijih čitača (tj. IE5 and IE6) kreiranje ovog objekta se vrši posebno. Mi se nećemo baviti njima i ovaj deo nećemo pisati obavezno u kodu.

```
if (window.XMLHttpRequest) {
    xmlReq = new XMLHttpRequest();
} else {
    // stariji web čitači
    xmlReq = new ActiveXObject("Microsoft.XMLHTTP");
}
```

Slanje zahteva serveru

- Za slanje zahteva serveru koriste se metode `open` i `send`. Metoda open definiše vrstu zahteva i proceduru koja se izvšava. Metoda send izvršava slanje.
- Sintaksa je sledeća:
- `open(metod, url, asinhrono,kor,loz);`
  - *`method`* – tip zahteva: GET ili POST. GET je jednostavniji i brži zahtev ali POST obezbeđuje neke prednosti. Pomoću POST zahteva može se dobiti veća količina podataka bez ograničenja, ne koriste se keširani podaci i takođe obezbeđuje sigurni prenos.
  - *`url`* – putanja do fajla ili akcije na serveru.
  - *`asinhrono`* – da li je asinhroni zahtev: true ili false. Asinhroim slanjem zahteva serveru JavaScript kod se izvršava nečekajući vraćanje informacija.
  - *`kor`* – korisnčko ime (opciono)
  - *`loz`* – lozinka (opciono)

- `send(POST_Param);` – slanje zahteva serveru. Za GET metodu slanja ne postoje parametri dok se za POST koriste. Kod GET metode parametri se pridružuju parametru url koji je u open metodi.

Serverski odziv

- Objekat XMLHttpRequest (kao i *Document*) sadrži svojstvo **readyState** koje sadrži status učitavanja dokumenta. Moguće vrednosti odziva su:
  - 0 – zahtev nije inicijalizovan,
  - 1 – uspostavljena konekcija sa serverom,
  - 2 – zahtev prihvaćen,
  - 3 – zahtev se obrađuje,
  - 4 – zahtev završen i odgovor je spreman.
- Svojstvo **status** objekta XMLHttpRequest vraća kod u vidu broja kao odgovor pri dobijanju sadržaja. Ovi kodovi su standardni:
  - 200 - „OK“
  - 403 – „Forbidden“
  - 404 – „Not Found“
- Tekstualni opis statusa („OK“, „Forbidden“, ...) nalazi se u svojsvu **statusText**.

Povratna funkcija

- Povratna funkcija (engl. Callback function) je funkcija koja se prosleđuje u vidu parametra drugoj funkciji. Na ovaj način je moguće istom funkcijom obraditi više AJAX zadataka prosleđujući različite metode za različite zadatke. Pogledajte primer 2

```
<!DOCTYPE html>
<html>
<body>
    <h1>Primer 2: Više AJAX metoda za zajedničku obradu</h1>
    <button type="button" onclick="loadXMLDoc(prikazFajla, 'info.txt', 'div1')">Učitaj sadržaj fajla 1 "info.txt"</button>
    <button type="button" onclick="loadXMLDoc(prikazFajla, 'proba.txt', 'div2')">Učitaj sadržaj fajla 2 "proba.txt"</button>
    <br/>
    <button type="button" onclick="loadXMLDoc(prikazFajlaA, 'info.txt', 'div1')">Prikazi sadržaj fajla 1 "info.txt"</button>
    <button type="button" onclick="loadXMLDoc(prikazFajlaA, 'proba.txt', 'div2')">Prikazi sadržaj fajla 2 "proba.txt"</button>
    <div id="div1"></div>
    <div id="div2"></div>
```

Nastavak koda na sledećem slajdu:

```
    <script>
        function loadXMLDoc(fja, url, divID) {
            var xmlReq = new XMLHttpRequest();
            xmlReq.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
                    fja(this, divID);
                }
            };
            xmlReq.open("GET", url, true);
            xmlReq.send();
        }
        function prikazFajla(xmlReq,divID) {
            document.getElementById(divID).innerHTML = xmlReq.responseText;
        }
        function prikazFajlaA(xmlReq, divID) {
            alert(xmlReq.responseText);
        }
    </script>
</body>
</html>
```

U gornjem primeru koristi se zajednička funkcija za prikaz sadržaja fajla *loadXMLDoc*. Različitim parametrima istu metodu koristimo za prkaz različitih fajlova u različitim odeljcima. Takođe koristimo dve metode za različite prikaze.

Svojstvo responseXML

- Objekat XMLHttpRequest sadrži ugrađeni XML parser. Svojstvo **responseXML** vraća odziv kao DOM model objekat. Nad ovim objektom mogu se primeniti svojstva i metode tipične za obradu DOM objekata. U narednom primeru izdvajaju se podaci iz XML fajla.

```
<html>
<body>
    <h1>Primer 3: XML parsiranje</h1>
    <button type="button" onclick="loadXMLDoc('simple.xml')">Učitaj sadržaj fajla 1 "simple.xml"</button>
    <div id="div1"></div>
    <script>
        function loadXMLDoc(url) {
            var xmlReq = new XMLHttpRequest();
            xmlReq.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
                    prikazFajla( parsiranje(xmlReq) );
                }
            };
            xmlReq.open("GET", url, true);
            xmlReq.send();
        }
```

Nastavak koda na sledećem slajdu:

```
        function parsiranje(xmlReq) {
            var rez = "";
            var xmlDoc = xmlReq.responseXML;
            var lst_food = xmlDoc.getElementsByTagName("food");
            for (i = 0; i < lst_food.length; i++) {
                rez = rez + lst_food[i].getElementsByTagName("name")[0].childNodes[0]. nodeValue + "<br>";
            }
            return rez;
        }
        function prikazFajla(rez) {
            document.getElementById("div1").innerHTML = rez;
        }
    </script>
</body>
</html>
```

XML DOM model

- DOM (engl. Document Object Model) je jezički nezavisna platforma definisana **interfejsom** koja omogućava rad sa podacima, strukturom i stilovima dokumenta. HTML DOM definiše standardni način pristupa i manipulacije HTML dokumentima.
- XML DOM definiše standard za pristup i manipulaciju podacima XML dokumenta.

Dobijanje vrednosti XML elementa

- Na primer:
- `r = xmlDoc.getElementsByTagName("name")[0].childNodes[0].nodeValue;`
- Na ovaj način se prvo formira lista objekata koji sadrže elemente `name`, zatim se uzima privi u toj listi sa indeksom `[0]`, zatim se pristupa podčvorovima, i to prvom podčvoru i uzima njegova vrednost. Za prikaz vrednosti podčvora koristili smo svojstvo `nodeValue`. Postoje i druge mogućnosti koje navodimo ukratko:
- `nodeValue` - vrednost čvora.
- `innerHTML` – parsira se HTML sadržaj (duže traje).
- `textContent` – koristi se čist tekstualni sadržaj bez parsiranja.
- `innerText` – koriste se stilovi (npr: neće se prikazati skriveni tekst).

- Primeri:
- `el.textContent='<a href="http://google.com">google</a>'`
- `Output: <a href="http://google.com">google</a> (without spaces)`

- `el.innerHTML='<a href="https://google.com">google</a>'`
- `Output: google`

- Neka tipična DOM svojstva koja proizilaze iz strukture su:
  - `x.nodeName` – naziv čvora x
  - `x.nodeValue` – vrednost čvora x
  - `x.parentNode` – roditeljski čvor za x
  - `x.childNodes` – deca čvorovi od x
  - `x.attributes` – čvorovi atributa od x.

Učitavanje XML fajla

- Učitavanje XML fajla je prikazano u već korišćenim primerima. Samo učitavanje vrši se pomoću objekta XMLHttpRequest i identično je učitavanju bilo kog drugog tekstualnog fajla.
- Ono po čemu se XML fajl razlikuje od drugih tekstualnih fajlova je da se nakon učitavanja može koristiti svojstvo **responseXML** objekta XMLHttpRequest koje sadrži XML DOM model dokumenta.

Primer:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        table, td {
            border: 1px solid black;
            border-collapse: collapse;
        }
        td {
            padding: 5px;
        }
        .tblHeader {
            color: red;
            font-weight: bold;
        }
    </style>
</head>
<body>
```

Nastavak koda na sledećem slajdu:

```
<h1>Primer 4. Prikaz CD kataloga</h1>

    <button type="button" onclick="loadXMLDoc()">Prikaz kataloga</button>
    <br><br>
    <table id="div1"></table>

    <script>
        function loadXMLDoc() {
            var xmlReq = new XMLHttpRequest();
            xmlReq.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
                    prikazFajla(formiranjeTabele(xmlReq));
            }
          };
            xmlReq.open("GET", "cd_catalog.xml", true);
            xmlReq.send();
        }
        function formiranjeTabele(xmlReq) {
          var xmlData = xmlReq.responseXML;
          var tableHTML = "<tr><td class='tblHeader'>Izvođač</td><td class='tblHeader'>Naslov</td></tr>";
          var cds = xmlData.getElementsByTagName("CD");
          for (let i = 0; i < cds.length; i++) {
              let cd = cds[i];
              tableHTML += "<tr><td>" +
              cd.getElementsByTagName("ARTIST")[0].childNodes[0].data + "</td><td>" +
              cd.getElementsByTagName("TITLE")[0].childNodes[0].data + "</td></tr>";
          }
          return tableHTML;
        }
        function prikazFajla(rez) {
            document.getElementById("div1").innerHTML = rez;
        }
    </script>
</body>
</html>
```

Nalaženje odgovarajućih čvorova

Pristup do čvorava moguće je uraditi na više načina.

1. **Koristeći `getElementsByTagName()`**

- Kao u gornjem primeru: `xmlData.getElementsByTagName ("CD");`
- Ova metoda vraća listu objekata koji predstavljaju sve elemente zadatog imena. Metoda se primenjuje na dokument (čvor dokumenta) ili na sam čvor koji je rezultat ove metode. U primeru se vidi da na ovaj rezultat vršimo dalje izdvajanje čvora primenom iste metode.

2. **Iterativno, prelazeći sve čvorove kroz petlju**

```
var cds = xmlData.getElementsByTagName("CD");
for (let i = 0; i < cds.length; i++) {
    let cd = cds[i];
    tableHTML += "<tr><td>" +
    cd.getElementsByTagName("ARTIST")[0].childNodes[0].textContent +
    </td><td>" +
    cd.getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +
    "</td></tr>";
}
```

3. Navigacijom kroz stablo dokumenta

U ovom načinu koriste se strukturne relacije između elemenata. Na primer:

```
function navigacija(xmlReq) {
    var x, y, i, xlen, xmlDoc, txt;
    xml = xmlReq.responseXML;
    x = xml.getElementsByTagName("CD")[0];
    brPodelemenata = x.childNodes.length;
    cvor = x.firstChild;
    txt = "";
    for (i = 0; i < brPodelemenata; i++) {
        if (cvor.nodeType == 1) {
            txt += i + " " + cvor.nodeName + "<br>";
        }
        cvor = cvor.nextSibling;
    }
    return txt;
}
```

Tipovi čvorova koji se spominju u primeru

Tip čvora	Opis	Deca čvorovi
1	Predstavlja element	Element, Tekst, Komentari, Instrukcije obrade, CDATA sekcija, entitet
2	Predstavlja atribut	Text, EntityReference
3	Tekstualni sadržaj elementa ili atributa	None
4	Predstavlja jednu CDATA sekciju	None
5	Referencu na entitet	Element, instrukcija obrade, komentar, Tekst, CDATA sekcija, entitet
6	Predstavlja jedan entitet	Element, instrukcija obrade, komentar, Tekst, CDATA sekcija, entitet
7	Jedna instrukcija obrade	None
8	Jedan komentar	None
9	Ceo dokument	Element, ProcessingInstruction, Comment, DocumentType