Osnove web programiranja

jQuery, AJAX i JSON

Termin 12

Sadržaj

1. AJAX
   - ideja
   - osobine
   - Implementacija u JS
- JSON
- jQuery i Ajax

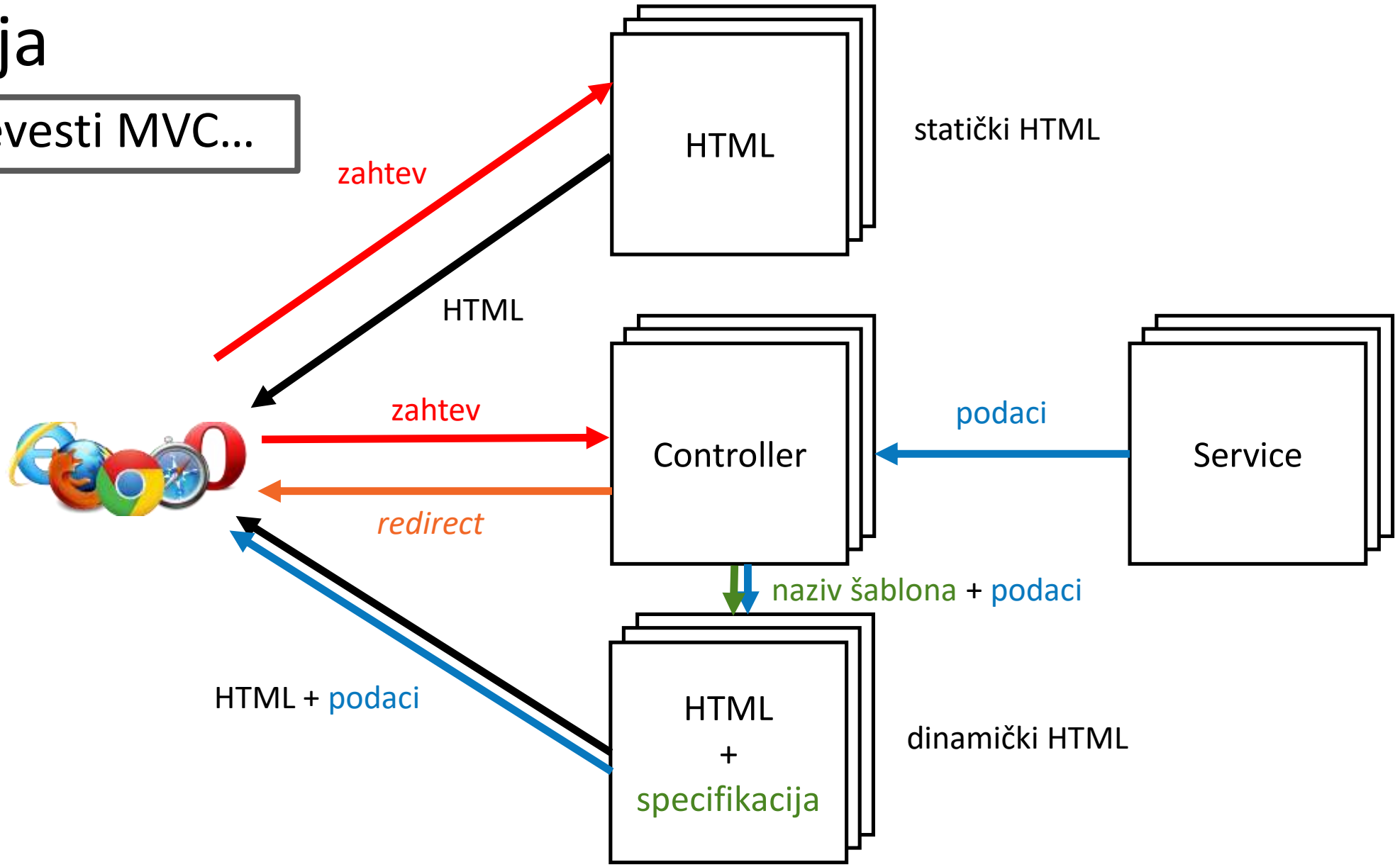
Nedostaci dosadašnjeg pristupa u generisanju dinamičkog HTML sadržaja

- za svaku i najmanju promenu u prikazu stranice (npr. poruka o greški), server je morao da generiše potpuno novu stranicu sa dodatkom te male promene
- potencijalni problem server postaje usko grlo
- ovakav pristup nije pogodan za izrazito interaktivne aplikacije i aplikacije bogate multimedijalnim sadržajem (npr. stranice kao što su *Google*, *Facebook*, *Youtube*, *Instagram* i sl.)
- učitavanje takvih stranica se broji u sekundama, pa bi čestim zahtevima za promenom, pri čemu bi se svaki put učitavala potpuno nova stranica, aplikacija postala spora (*unresponsive*)

AJAX

Ideja

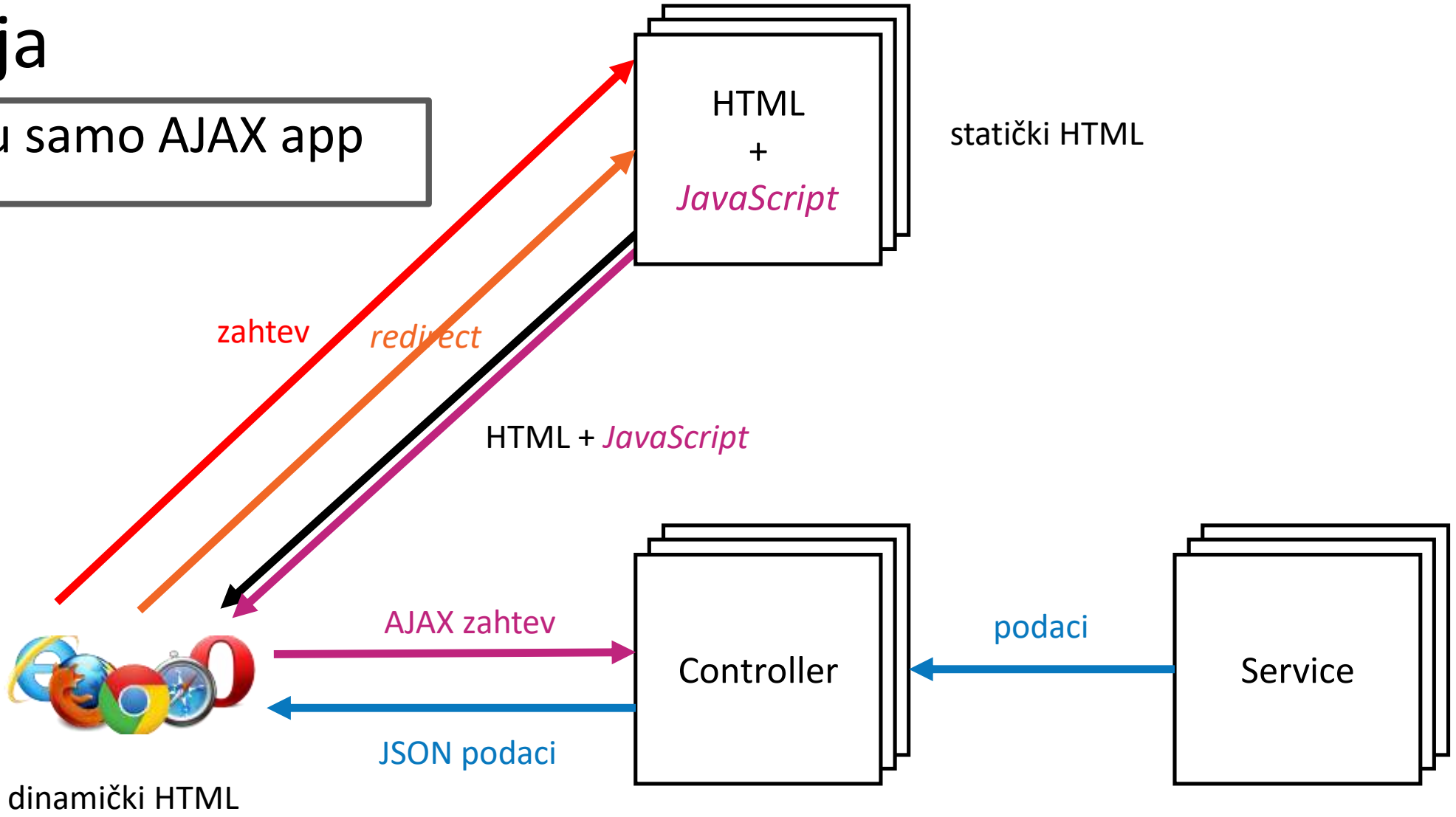
- prevesti MVC…



AJAX

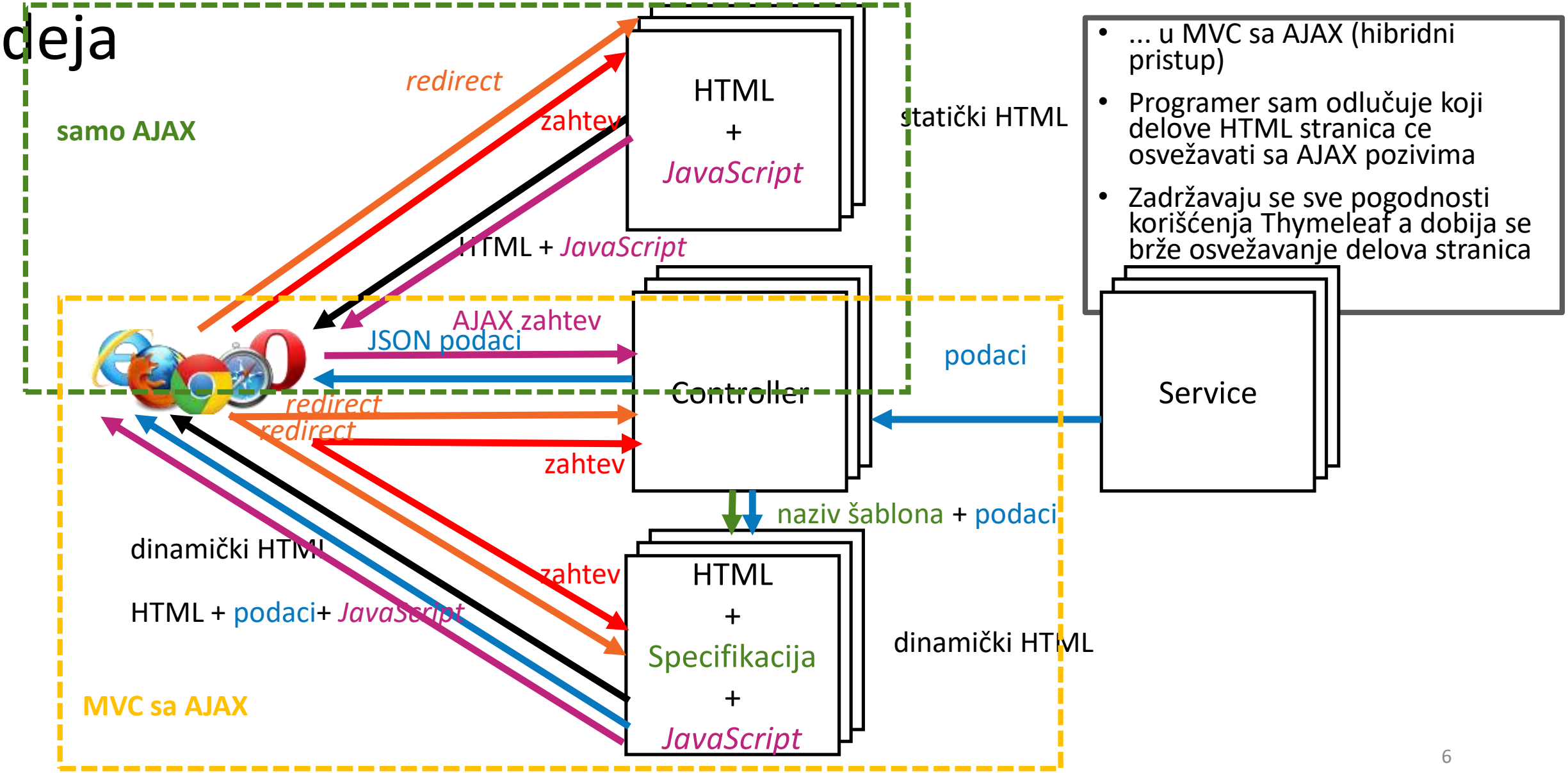
Ideja

- ... u samo AJAX app



AJAX

Ideja



- ... u MVC sa AJAX (hibridni pristup)
- Programer sam odlučuje koji delove HTML stranica ce osvežavati sa AJAX pozivima
- Zadržavaju se sve pogodnosti korišćenja Thymeleaf a dobija se brže osvežavanje delova stranica

AJAX

MVC

1. *web browser*:
   a) traži od servera statičku HTML stranicu
   b) šalje zahtev na *controller*
2. server:
   a) vraća statički HTML stranicu
   b) 1. čita podatke iz servisa, prosleđuje ih *template engine*-u i vraća dinamički HTML
   b) 2. vrši redirekciju na statički HTML ili na drugu metodu istog ili drugog *controller*-a

AJAX

1. *web browser*:
   a) traži od servera statičku HTML stranicu
2. server:
   a) vraća statički HTML stranicu sa pridruženim *JavaScript* programom
3. *web browser*:
   a) *JavaScript* program pravi jedan ili više uzastopnih AJAX zahteva ka *controller*-ima servera (inicijalno i na korisničke događaje)
4. server:
   a) vraća podatke *JavaScript* programu u vidu JSON objekata
5. *web browser*:
   a) *JavaScript program* ugrađuje podatke u HTML stranicu i menja je
   b) *JavaScript program* traži neku drugu statičku HTML stranicu (vrši *client-side* redirekciju)

AJAX

Uloge

web browser (client):

- izvršava programsku logiku vezanu za prikaz
- izvršava programsku logiku vezanu za kontrolu toka
- zvršava *client-side* validaciju podataka gde je to moguće

server:

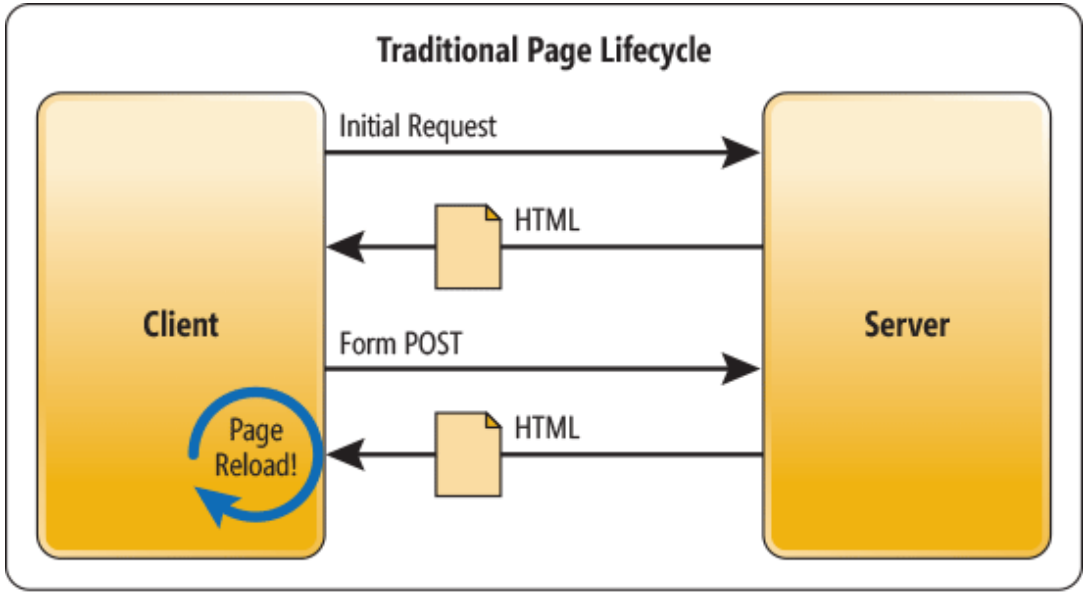
- obavlja ulogu repozitorijuma podataka: čuva integritet podataka i omogućuje da više klijenata konzistentno rukuju istim podacima
- i dalje može da čuva stanje korisničke sesije ili *context*-a aplikacije
- i dalje vrši *server-side* validaciju podataka
- i dalje obavlja poslovnu logiku (servisi)
- i dalje vrši perzistenciju podatka (DAL, baza)

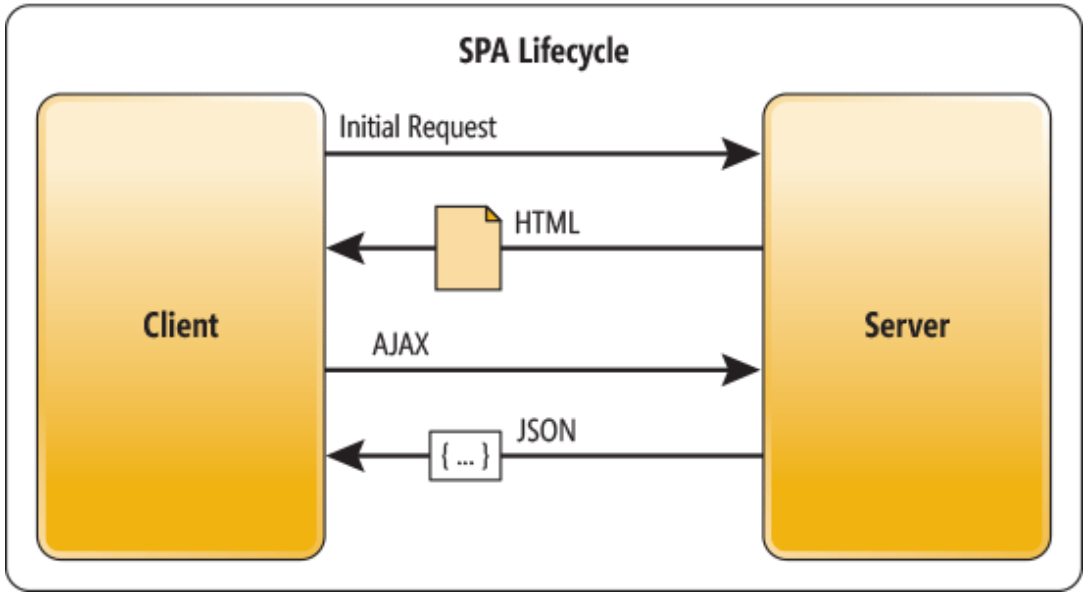
AJAX

- *Asynchronous JavaScript and* XML
- prevod: *JavaScript* program pravi asinhrone pozive ka serveru i sa njim razmenjuje XML podatke
- tehnika za kreiranje brzih interaktivnih dinamičkih web stranica
- u prvim inkarnacijama ovog koncepta, zaista su se razmenjivali XML podaci, dok ih je vremenom zamenio JSON format zbog jednostavnosti i činjenice da ga *JavaScript* ima ugrađenu podršku za njega
- asinhrona priroda zahteva podrazumeva da *web browser* nakon korisničkog događaja ne blokira interfejs i ne čeka odgovor servera da bi vratio korisniku kontrolu nad *web browser-om*, već u paralelnom toku (programskoj niti) inicira zahtev i čeka odgovor, a odgovor obrađuje tek kada on stigne;
- to može da potraje i nekoliko sekundi, a korisnik za to vreme može da nastavi da korisiti interfejs (**ne blokira se stranica, korisnik nije izgubio osećaj kontrole, nema louder sličice koja izluđuje korisnike**)

AJAX

- *Asynchronous JavaScript and* XML
- Kod klasičnih web stranica, potrebno je osvežiti čitavu web stranicu kako bi se promenio sadržaj.
- Umesto da se osvežava čitava stranica, učitavaju se podaci sa servera i osvežavaju delovi stranice
- web stranica se osvežava asinhrono, razmenom male količine podataka sa serverom. Ovo omogućava da se osvežavaju delovi stranice.



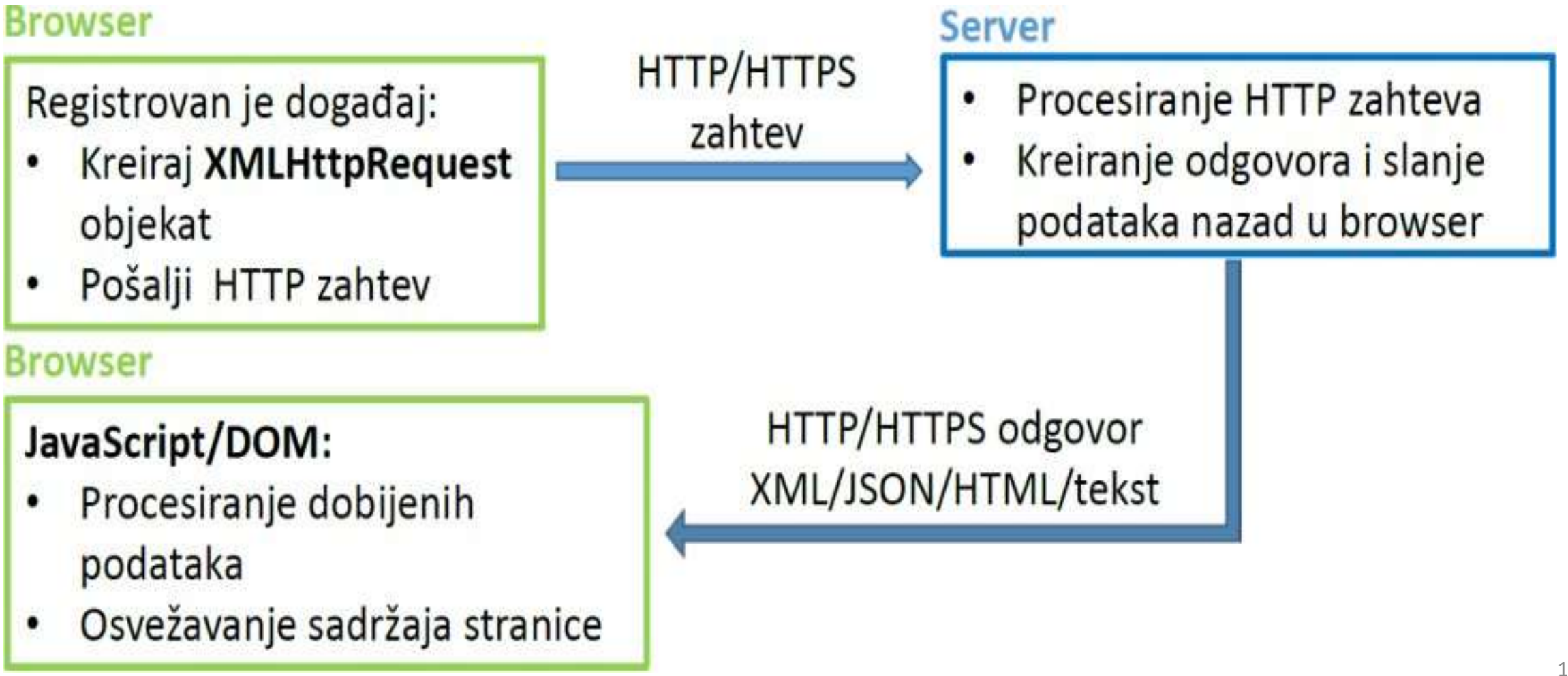


AJAX

- Nije nova tehnologija već je kombinacija postojećih tehnologija:
  - **XMLHttpRequest object** – asinhrona razmena podataka sa serverom
  - **JavaScript/DOM** – izmena strukture i sadržaja bez ponovnog učitavanja stranice
  - **CSS** – uređivanje izgleda stranice
  - **XML (često JSON)** – format podataka koji se razmenjuju

AJAX

- Povratna vrednost servera može biti: XML, JSON, HTML, ili običan tekst



AJAX

Prednosti

- Web stranica je prijatnija za korišćenje:
  - Ne gubi se sav operacioni sadržaj dok se stranica učitava (npr. Za kupovinu bioskopskih karti, postoji wizard sa više formi koje se preko ajax smenjuju, podaci iz prethodnih fomi se čuvaju u stranici);
  - Instant odgovor – korisnik ne mora da čeka sledeću stranicu;
  - Ne gubi se pozicija na koju je korisnik skrolovao;
  - Navigacija na stranici je lepša u odnosu na standardno backward/forward u browseru;
  - Osećaj desktop aplikacije.

AJAX

Prednosti

- ostaje se na istoj stranici koliko god je to potrebno (**moguće je i trajno i tada se aplikacija naziva *single-page web* aplikacija**); stranica se menja tek kada je potrebno značajno promeniti kontekst prikaza (npr. *Google, Gmail, Google Drive* i sl.
- programska logika koja se bavi prikazom se izmešta u *JavaScript* programe, a izvršava ih *web browser* i time se rasterećuje server (server više nije usko grlo)
- pri korisničkim događajima, stranica se prepravlja i dopunjuje umesto da se učitava nova.
- Data driven pristup (nasuprot klasičnim web aplikacijama koje su page-driven).

AJAX

Prednosti

- razmenjuju se JSON objekti, koji su tekstualne reprezentacije u obimu koji je daleko manji od koda kompletne HTML stranice, pa se time štedi mrežni kanal za prenos podataka
- uvećane performanse aplikacije i brzina odgovora.
- sva programska logika vezana za prikaz je sada iskazana isključivo *JavaScript* programima, što otvara mogućnost za profilisanje *developer*-a (*frontend*, *backend*)

AJAX

Izazovi

- *JavaScript* programe je teže *debug*-ovati od *Java* programa
- *JavaScript* programe je teže optimizovati od Java programa
- previše kompleksni *JavaScript* programi ne mogu da se izvršavaju na sporijim uređajima
- korisnik može da vidi *JavaScript* programe u svom *web browser*-u, pa može da njihove fragmente iskoristi za druge namene (*reverse engineering*), ili pokuša da *hack-uje* server jer ima više informacija o načinu funkcionisanja aplikacije
- validacija zahteva na serveru mora da bude besprekorna

AJAX

Izazovi

- Usled dinamičkog učitavanja sadržaja otežano je:
  - Registrovanje stanja u istoriji browsera (problem backward/forward navigacije);
  - Bookmark-ovanje nekog konkretnog stanja;
  - Indeksiranje od strane pretraživača (dinamički proizveden sadržaj u opštem slučaju nije vidljiv crawler-ima).
- Korisnici koji koriste browsere koji nemaju podršku ili imaju onesposobljen JavaScript neće biti u stanju da koriste funkcionalnost obezbeđenu AJAX-om.

AJAX

Primeri upotrebe

- Kaskadne drop-down liste (Primer:http://demo.tutorialzine.com/2011/11/chained-ajax-selects-jquery/).
- Glasanje, rejting i ostale instant akcije, kada korisnik očekuje instant odgovor.
- AutoComplete – automatski prikaz rezultata koji odgovaraju pretrazi (Facebook friend search).
- Paginacija – pretraga, sortiranje i organizacija podataka pristiglih sa servera.
- AutoSave – čuvanje sadržaja bez potrebe za čekanjem od strane korisnika (compose u Gmail).

AJAX

- Nije nova tehnologija već je kombinacija postojećih tehnologija:
  - **XMLHttpRequest object** – asinhrona razmena podataka sa serverom
  - **JavaScript/DOM** – izmena strukture i sadržaja bez ponovnog učitavanja stranice
  - **CSS** – uređivanje izgleda stranice
  - **XML (često JSON)** – format podataka koji se razmenjuju

AJAX

XMLHttpRequest

- Koristi se za slanje HTTP i HTTPS zahteva iz skritpe ka serveru i za slanje odgovora sa servera u skriptu.
- Radi nezavisno od stranice
  - pozovemo metodu **send** i ona uputi HTTP zahtev nezavisno od glavne stranice
- Koristi se za
  - Ažuriranje web stranice bez potrebe da se ona ponovno učita
  - Zatraživanje podataka sa servera – nakon što je stanica učitana
  - Prihvatanje podataka sa servera – nakon što je stanica učitana
  - Slanje podataka serveru – u pozadini
- Povratna vrednost servera može biti: **XML, JSON, HTML, ili običan tekst**

AJAX

XMLHttpRequest

- Atribut **readyState** sadrži informaciju o stanju XMLHttpRequest
- Funkcija definisana **onreadystatechange** se svaki put poziva kada **readyState** atribut promeni vrednost
- Tokom slanja zahteva serveru **readyState** atribut se menja od 0 do 4
  - 0: zahtev nije inicijalizovan
  - 1: konekcija se serverom ostvarena established
  - 2: zahtev primljen na serveru
  - 3: procesiranje zahteva
  - 4: zahtev obrađen i odgovor je spreman
- Funkcija definisana **onload** poziva se samo kada se XMLHttpRequest tansakcija uspešno završi tj. kada odgovor stigne

AJAX

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous)
send()	Sends a request to the server (used for GET)
send(*string*)	Sends a request string to the server (used for POST)
onreadystatechange	A function to be called when the readyState property changes
readyState	The status of the XMLHttpRequest 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: OK 404: Page not found
responseText	The response data as a string
responseXML	The response data as XML data

Implementacija sa XMLHttpRequest

```
//Tipična upotreba bi bila
var xhttp = new XMLHttpRequest();
// poziva se svaki put kada readyState menja vrednosti
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
       // Action to be performed when the document is read;
    }
};
//true označava asinhronu komunikaciju, deault je true
xhttp.open("GET", "URL", true);
//pozivom open readyState dobija vrednost 1
// konekcija se serverom established
xhttp.send();
//posle poziva send, readyState menja vrednosti u 2,3,4
```

AJAX

Implementacija sa XMLHttpRequest

https://httpbin.org/html

primer01-JavaScript.html

BioskopVebAplikacijaT12-SamoAJAX/ajax.html

TestAjaxHTMLController

```
//Tipična upotreba bi bila
var xhttp = new XMLHttpRequest();
// poziva se samo kada odgovor stigne
xhttp.onload = function() {
    if (this.status == 200) {
       // Action to be performed when the document is read;
    }
};
//true označava asinhronu komunikaciju, deault je true
xhttp.open("GET", "URL", true);
xhttp.send();
```

JSON

Uvod

- JavaScript Object Notation
- Jednostavan format za razmenu podataka predstavljenih pomoću teksta
- Danas primarni format podataka korišćen za asinhronu komunikaciju između klijenta i servera

JSON

Prednosti

- Nezavisan od konkretnih tehnologija sa kojima se koristi;
- Lako se razume
- Kod za generisanje i parsiranje JSON-a je dostupan u većini programskih jezika
- Sintaktički identičan kodu za kreiranje objekata u JavaScriptu

JSON

Sintaksa

- Nizovi ili kolekcije su predstavljene kao vrednosti odvojene zarezom koje su obuhvaćene simbolima '[' i ']'
- Stringovi su pod navodnicima, primitivni tipovi nisu
- Vitičastim zagradama '{' i '}' obeležavamo blok ili strukturu

JSON

Sintaksa

- Podaci su predstavljeni kao parovi **ključ:vrednost**
- Podaci su odvojeni zarezima
- Podaci mogu biti:
  - Brojevi (celi ili razlomljeni)
  - Stringovi (pod navodnicima)
  - Boolean vrednosti (true ili false)
  - Nizovi ili kolekcije (u uglastim zagradama)
  - Objekti (u vitičastim zagradama)
  - null

JSON

Primer

```
{
    "studenti": [
        {
            "id": 1,
            "ime": "Pera",
            "prezime": "Perić",
            "email":"pera@gmail.com"
        },
        {
            "id": 2,
            "ime": "Steva",
            "prezime": "Stević",
            "email": null
        },
        {
            "id": 3,
            "ime": "Jova",
            "prezima": "Jović",
            "email":"jova@gmail.com"
        }
    ]
}
```

JSON

Primer

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

JSON

Primer

```
{ "users":[
        {
            "firstName":"Ray",
            "lastName":"Villalobos",
            "joined": {
                "month":"January",
                "day":12,
                "year":2012
            }
        },
        {
            "firstName":"John",
            "lastName":"Jones",
            "joined": {
                "month":"April",
                "day":28,
                "year":2010
            }
        }
    ]}
```


```
[
    "Rail Booking": {
        "reservation": {
            "ref_no": 1234567,
            "time_stamp": "2016-06-24T14:26:59.125",
            "confirmed": true
        },
        "train": {
            "date": "07/04/2016",
            "time": "09:30",
            "from": "New York",
            "to": "Chicago",
            "seat": "57B"
        },
        "passenger": {
            "name": "John Smith"
        },
        "price": 1234.25,
        "comments": ["Lunch & dinner incl.", "\"Have a nice day!\""]
    }
]
```

JSON

JSON i JavaScript

- Kada se razmenjuju sa Web serverom, podaci se razmenjuju kao string.
- **JSON.parse(tekst)** parsira podatke (string) u JavaScript objekat.
- **JSON.stringify(objekat)** konvertuje JavaScript objekat u string.
- **JSON.stringify(objekat):**
  - Konvertuje sve datume u stringove;
  - Uklanja sve funkcije iz JavaScript objekta.

JSON i JavaScript - na klijentskoj stani

```
//kreira se JSON string od objekta
//da bi se preko AJAX poslao serveru
var jsonString = JSON.stringify(objekat);

//server je vratio JSON string
//koji se konvertuje u objekat
var objekat = JSON.parse(jsonString);
```

Implementacija sa JSON

- u čistom *JavaScript*-u se oslanja na *XMLHttpRequest* objekte

```
var naziv = … // pročitan iz forme za pretragu

var url = "Zanrovi?naziv=" + naziv
console.log(url)

var request = new XMLHttpRequest(); // kreiranje zahteva
request.open("GET", url); // podešavanje tipa i URL-a zahteva
// konverzija string-ovnog odgovora u JSON formatu u JSON objekat
request.responseType = "json";

// funkcija koja će se izvršiti kada odgovor stigne
// function called when an XMLHttpRequest
// transaction completes successfully.
request.onload = function() {
    console.log(request.response) // ispis odgovora
}
request.send() // slanje zahteva
console.log("GET: Zanrovi")
```

```
Zanrovi?naziv=ja

GET: Zanrovi
{…}
    status: "ok",
    zanrovi: (2) […]
        0: Object { id: 2, naziv: "akcija" }
        1: Object { id: 3, naziv: "komedija" }
```

AJAX

Implementacija sa JSON

- XMLHttpRequest.response Read only
  Returns an ArrayBuffer, Blob, Document, JavaScript object, or a DOMString, depending on the value of XMLHttpRequest.responseType, that contains the response entity body.

- XMLHttpRequest.responseText Read only
  Returns a DOMString that contains the response to the request as text, or null if the request was unsuccessful or has not yet been sent.

jQuery i Ajax

Uvod

nastavak naredni cas

- jQuery pojednostavljuje AJAX funkcionalnosti
- Različiti brauzeri podržavaju različitu sintaksu AJAX poziva
- jQuery rešava ovaj problem, pa se AJAX poziv svodi na jednu liniju koda
- Više na: http://api.jquery.com/category/ajax/

jQuery i Ajax

Uvod

Uz pomoć *jQuery* biblioteke, oslanja se na poziv jedne funkcije

- $(*selector)*.load("URL" [, data] [, *callback*])
- $.get("URL" [, data] [, *callback*])
- $.post("URL" [, data] [, *callback*])
- *$*.ajax(url [, settings])

```
var naziv = "ja"; // pročitan iz forme za pretragu

// parametri kao JSON objekat
var params = {
    naziv: naziv
}
console.log(params)
// URL, parametri i handler koji će da obradi odgovor
// kada stigne
$.get("Zanrovi", params, function(odgovor) {
    console.log(odgovor) //ispis odgovora
})
console.log("GET: Zanrovi")
```

može se izostaviti

!!!

function(responseTxt, statusTxt, xhr)

Callback funkcija ima dodatne opcione parametre

- responseTxt - sadrži odgovor servera ako poziv metode uspe
- statusTxt - sadrži status poziva metode("success", "notmodified", "error", "timeout", or "parsererror")
- xhr - sadrži XMLHttpRequest objekat

```
Object { naziv: "ja" }
GET: Zanrovi
{…}
    status: "ok",
    zanrovi: (2) […]
        0: Object { id: 2, naziv: "akcija" }
        1: Object { id: 3, naziv: "komedija" }
```

jQuery i Ajax

$(selector).load

- Asinhrono učitavanje sadržaja sa servera i smeštanje tog sadržaja u selektovani element
- $(*selector)*.load("URL" [, data] [, *callback*])
- $(*selector)*.load("URL" [, data] , function(responseTxt, statusTxt, xhr))
- data – opcioni podaci koji se šalju serveru
- Dodati parameti *callback* funkcije:
  - responseTxt - sadrži odgovor servera ako poziv metode uspe
  - statusTxt - sadrži status poziva metode("success", "notmodified", "error", "timeout", or "parsererror")
  - xhr - sadrži XMLHttpRequest objekat

jQuery i Ajax

$(selector).load

- Primer:

```
$("div#div2").load('https://httpbin.org/html');

$("#myDiv").load('TestAjaxHTML', {"tekst": "Juhu"}, function (responseTxt, statusTxt, xhr)
{
        alert(responseTxt);
        if (statusTxt == "success")
                alert("External content loaded successfully!");
        if (statusTxt == "error")
                alert("Error: " + xhr.status + ": " + xhr.responseTxt);
});
```

- Pitanje za sve koja se HTTP metoda poziva kod jQuery load funkcije?

jQuery i Ajax

$(selector).load

- Pitanje za sve koja se HTTP metoda poziva kod jQuery load funkcije?
  - Ako se zada drugi parametar data tada se poziva Post metoda u suprotnom je Get

POST

```
$("#result").load("AjaxPages/Page.html", { "name": "hajan" }, function () {
    ////callback function implementation
});
```

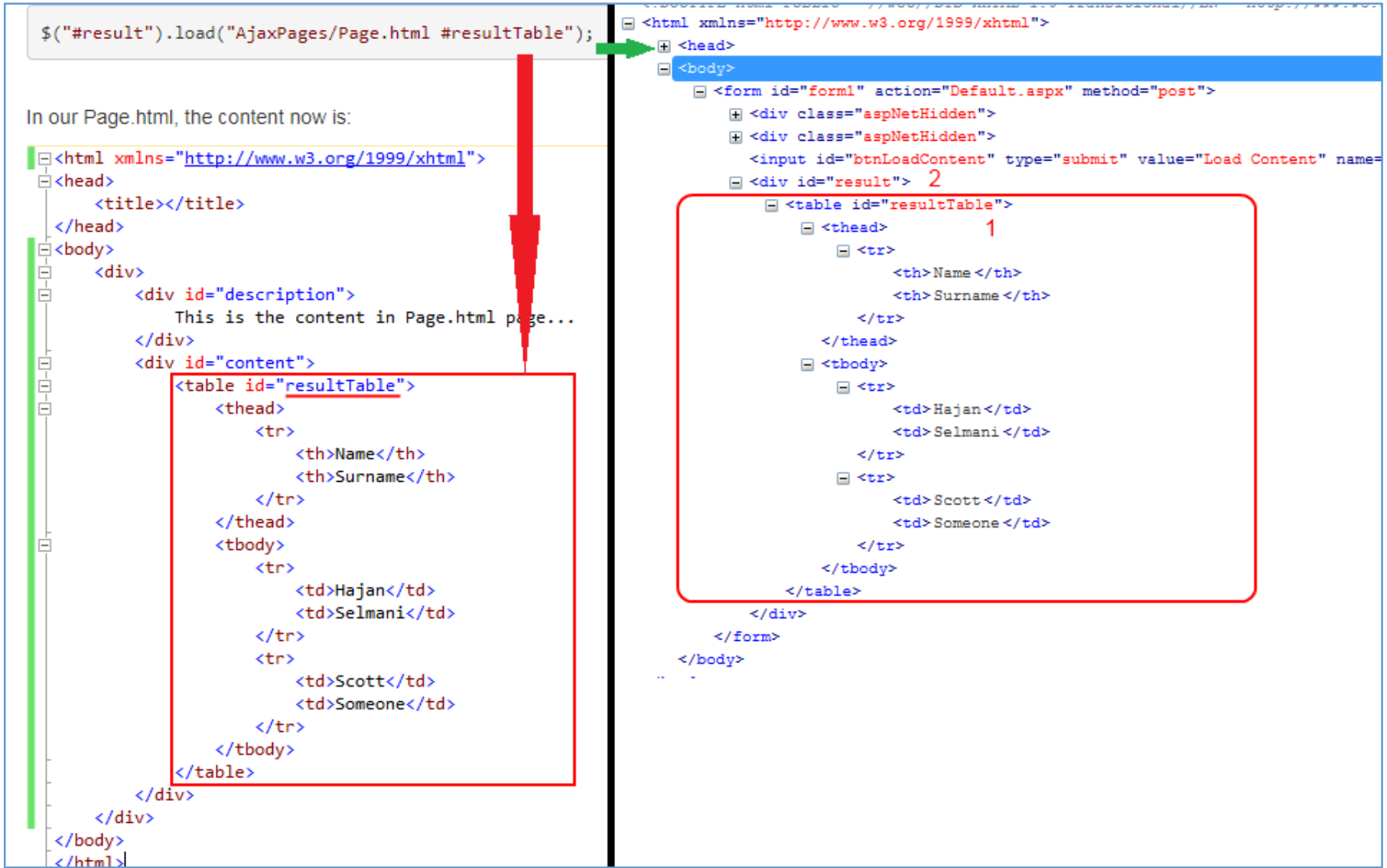
GET

```
$("#result").load("AjaxPages/Page.html", function () {
    ////callback function implementation
});
```

jQuery i Ajax

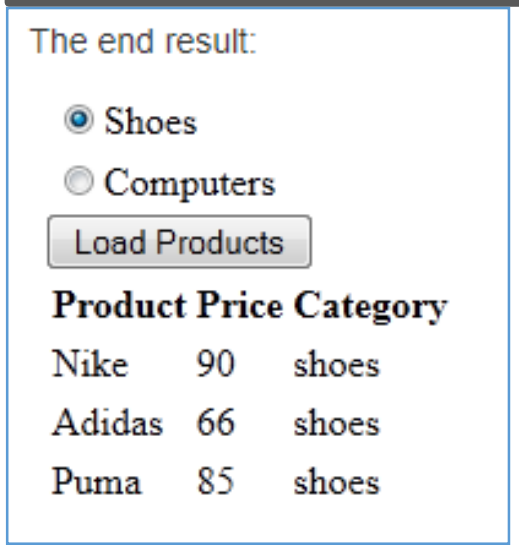
$(selector).load

- Ne mora se očitavati cela stranica već samo deo stranice može da se učita



$(selector).load

- Kako još možemo da iskoristimo fukciju load
- Npr. U zavisnosti od odabranog radio dugmeta možemo da prikažemo različite kategorije za kupovinu
  - Definiše se funkcija onClick za dugme koja poziva load JQuery funkciju
  - Load funkcija gađa akciju Kontrolera ili statičku HTML stranicu
  - Sadržaj vraćene HTML stranice će se prikazati ispod dugmeta očitati odabrane




```
//call GetProducts.aspx with the category query string for the selected category in radio button list
//filter and get only the #tableProducts content inside #products div
$("#products").load("AjaxPages/GetProducts.aspx?category=" + selectedRadioButton + " #tableProducts");
```

jQuery i Ajax

$(selector).load

- Više o funkciji
- https://www.w3schools.com/jquery/jquery_ajax_load.asp
- https://api.jquery.com/load/

https://httpbin.org/html

primer01.html

BioskopVebAplikacijaT12-SamoAJAX/ajax.html TestLoadAjaxJQuery inside form

jQuery i Ajax

$.get() i $.post()

- Šalju poziv na server i dobijaju odgovor
- Cela operacija se odvija nezavisno od brauzera
- Rezultat se dobija preko callback funkcije
- get() - zahteva resurs sa servera
- $.get(URL,callback);
- post() - slanje podataka na server
- $.post(URL,data,callback);
- https://www.w3schools.com/jquery/jquery_ajax_get_post.asp

jQuery i Ajax

$.get()

BioskopVebAplikacijaT12-SamoAJAX/ajax.html TestLGetAjaxJQuery outside form

- Asinhrono zahteva resurs sa servera kroz HTTP get metodu
- $.get("URL" [, data] [, callback] [, dataType])
- *http://api.jquery.com/jQuery.get/*
- Primer: https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_ajax_get

Description: *Load data from the server using a HTTP GET request.*

jQuery.get(url [, data] [, success] [, dataType]) — **version added: 1.0**

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

success
Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
A callback function that is executed if the request succeeds. Required if `dataType` is provided, but you can use `null` or `jQuery.noop` as a placeholder.

dataType
Type: String
The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).

jQuery.get([settings]) — **version added: 1.12/2.2**

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All properties except for `url` are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) for a complete list of all settings. The type option will automatically be set to `GET`.

jQuery i Ajax

$.post()

BioskopVebAplikacijaT12-SamoAJAX/ajax.html TestLPostAjaxJQuery outside form

- Asinhrono zahteva resurs sa servera kroz HTTP post metodu
- $.post("URL" [, data] [, callback] [, dataType])
- http://api.jquery.com/jQuery.post/

Description: *Load data from the server using a HTTP POST request.*

jQuery.post(url [, data] [, success] [, dataType]) — **version added: 1.0**

url
Type: String
A string containing the URL to which the request is sent.

data
Type: PlainObject or String
A plain object or string that is sent to the server with the request.

success
Type: Function(PlainObject data, String textStatus, jqXHR jqXHR)
A callback function that is executed if the request succeeds. Required if `dataType` is provided, but can be `null` in that case.

dataType
Type: String
The type of data expected from the server. Default: Intelligent Guess (xml, json, script, text, html).

jQuery.post([settings]) — **version added: 1.12/2.2**

settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All properties except for `url` are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) for a complete list of all settings. Type will automatically be set to `POST`.

jQuery i Ajax

$.post()

```
var jsonObj = {
    "custname": "pera",
    "custtel": "0601234567",
    "custemail": "abc@def",
    "size": "small",
    "topping": ["bacon", "cheese"],
    "delivery": "11:30",
    "comments": "abc"
};

$.post('https://httpbin.org/post', JSON.stringify(jsonObj), function (resul
t, status) {
    if (status != "success")
        alert("Result: " + result + "\nStatus: " + status);
    if (status == "success")
        $("#div1").text(JSON.stringify(result));
```

https://httpbin.org/post

primer02.html

jQuery i Ajax

$.post()

https://jsonplaceholder.typicode.com/posts

primer02.html

```
var jsonObj = {
      "id": $("input#vrednostAtID").val()
};
$.get('https://jsonplaceholder.typicode.com/posts', jsonObj, fu
nction (result, status) {
if (status != "success")
      alert("Result: " + result + "\nStatus: " + status);
if (status == "success") {
      var tekst = '<table border="1">';
      for (i = 0; i < result.length; i++) {
            tekst += '<tr>';
            tekst += '<td>userId: ' + result[i].userId + '</td>';
```

jQuery i Ajax

$.ajax()

- AJAX zahtevi opšte svrhe(viši nivoi apstrakcije), GET ili POST
- Slično kao $.get() i $.post() funkcija $.ajax() omogućuje da se asinhrono zahteva resurs sa servera
- $.ajax(url [, settings])
- http://api.jquery.com/jquery.ajax/

Obrada događaja

- Moguće je uvezati fukcije koje rade određenu funkcionalost prilikom ajax poziva
- beforeSend()
  - Funkcija koja će se pozvati pre slanja zahteva;
  - Metoda može da vrati false, čime će se zahtev ukinuti.
- done()
  - Funkcija koja se poziva u slučaju da je zahtev serveru uspešno razrešen.
- fail()
  - Funkcija koja se poziva u slučaju da je zahtev serveru razrešen neuspehom.
- complete()
  - Funkcija koja se poziva kada se zahtev izvrši(bez obzira na ishod), nakon izvršavanja done() i fail().

Obrada događaja

```
$.ajax({
        method: "POST",
        url: "some.php",
        data: { name: "John", location: "Boston" }
})
        .done(function( msg ) {
                alert( "Data Saved: " + msg );
        })
        .fail(function( jqXHR, textStatus) {
                alert( "Request failed: " + textStatus );
        });
```

primer03.html

$.post("http://localhost:8080/url", …)
.fail(function(data, status){alert("Request failed!");}) .done(function(data, status){alert("Reqeust successful");});

$.getJSON() i $.getScript()

- $.getJSON(URL, data, successCallback)
  - Učitavanje JSON objekta pomoću GET metoda.
- $.getScript(URL, successCallback)
  - Učitavanje JavaScript-a sa servera, skripta se automatski izvršava

jQuery i Ajax

Uvod

Uz pomoć *jQuery* biblioteke, oslanja se na poziv jedne funkcije

- $(*selector)*.load("URL" [, data] [, *callback*])
- $.get("URL" [, data] [, *callback*])
- $.post("URL" [, data] [, *callback*])
- *$*.ajax(url [, settings])

```
var naziv = "ja"; // pročitan iz forme za pretragu

// parametri kao JSON objekat
var params = {
    naziv: naziv
}
console.log(params)
// URL, parametri i handler koji će da obradi odgovor
// kada stigne
$.get("Zanrovi", params, function(odgovor) {
    console.log(odgovor) //ispis odgovora
})
console.log("GET: Zanrovi")
```

može se izostaviti

!!!

function(responseTxt, statusTxt, xhr)

Callback funkcija ima dodatne opcione parametre

- responseTxt - sadrži odgovor servera ako poziv metode uspe
- statusTxt - sadrži status poziva metode("success", "notmodified", "error", "timeout", or "parsererror")
- xhr - sadrži XMLHttpRequest objekat

```
Object { naziv: "ja" }
GET: Zanrovi
{…}
    status: "ok",
    zanrovi: (2) […]
        0: Object { id: 2, naziv: "akcija" }
        1: Object { id: 3, naziv: "komedija" }
```

Asinhroni pozivi

```
// handler koji obrađuje događaj
$("form").submit(function() {
    var naziv = $("input[name=naziv]").val()

    var params = {
        naziv: naziv
    }
    console.log(params)
    // handler koji obrađuje odgovor
    $.get("Zanrovi", params, function(odgovor) {
        console.log(odgovor)
        …
    })
    console.log("GET", "Zanrovi")

    // sprečiti da submit forme promeni stranicu
    return false
});
```

glavni tok

paralelni tok

Asinhroni pozivi

```
@Component
@RequestMapping(value="/Zanrovi")
public class ZanroviController {

    @GetMapping
    @ResponseBody
    public Map<String, Object> index(@RequestParam(required=false, defaultValue="") String naziv) {
        // čitanje
        List<Zanr> zanrovi = zanrService.find(naziv);

        Map<String, Object> odgovor = new LinkedHashMap<>();
        odgovor.put("status", "ok");
        odgovor.put("zanrovi", zanrovi);
        return odgovor;
    }
```

ponovo je potrebno;
nalaže *Spring*-u da objekat koji
metoda vrati konvertuje u *string*-ovnu
JSON reprezentaciju

JSON

mapa prirodno podržava parove
(ključ, vrednost)

Asinhroni pozivi

```
// handler koji obrađuje događaj
$("form").submit(function() {
    var naziv = $("input[name=naziv]").val()

    var params = {
        naziv: naziv
    }
    console.log(params)
    // handler koji obrađuje odgovor
    $.post("Zanrovi", params, function(odgovor) {
        console.log(odgovor)
        …
    })
    console.log("GET", "Zanrovi")

    // sprečiti da submit forme promeni stranicu
    return false
});
```

glavni tok (odavno završen)

paralelni tok

Učitavanje početnih podataka

```
// po učitavanju stranice
$(document).ready(function() {
    …

    var params = …
    console.log(params)
    // handler koji obrađuje odgovor
    $.get("URL", params, function(odgovor) {
        console.log(odgovor)
        …
    })
    console.log("GET", "URL")

    …
});
```

Povezani zahtevi

```
…

var params = …
console.log(params)
$.get("URL1", params, function(odgovor) {
    console.log(odgovor)
    …

    var params = …
    $.get("URL2", params, function(odgovor) {
       console.log(odgovor)
       …
    })
    console.log("GET", "URL2")
    …
})
console.log("GET", "URL1")

…
```

Povezivanje stranica

- HTML stranice se povezivaju statički:

```
<ul>
    <li><a href="zanrovi.html">žanrovi</a></li>
    <li><a href="filmovi.html">filmovi</a></li>
    <li><a href="projekcije.html">projekcije</a></li>
    <li><a href="korisnici.html">korisnici</a></li>
</ul>
```

```
<a href="film.html?id=1">Avengers: Endgame</a>
```

- *client-side* redirekcija:

```
window.location.replace("filmovi.html")
```

Case study – CRUD bioskop veb aplikacija sa AJAX

- USE CASE korišćenje AJAX za BIoskop web aplikaciju
- *BioskopVebAplikacijaT12-SamoAJAX*
  - *com.ftn.PrviMavenVebProjekat:*
    - *index.html (index.js), IndexController*
    - *prijava.html (prijava.js), KorisnikController*
    - *filmovi.html (filmovi.js),* ***dodavanjeFilma.html (dodavanjeFilma.js)****, film.html(film.js), FilmoviController, ZanroviController*
    - *zanrovi.html (zanrovi.js), dodavanjeZanra.html (dodavanjeZanra.js), zanr.html(zanr.js), ZanroviController*
  - *Neostaju sve html staranice i js fajlovi za rad sa korisnicima i kompletan rad sa projekcijama (Kontroleri, html stanice i js fajlovi)*

- *BioskopVebAplikacijaT12-MVCsaAJAX*
  - *com.ftn.PrviMavenVebProjekat:*
    - *filmovi.html (filmovi.js), FilmoviController -* ***pretraga filmova***
    - ***sugestija****, pri CUD operacijama za entitete se AJAX može koristiti za redirekciju sa klijentske strane i ispis poruka, i za sve slučaje gde je neophodno osvežiti deo stranice*

Dodatni materijali

- https://www.w3schools.com/xml/ajax_intro.asp
- https://www.w3schools.com/jquery/jquery_ajax_intro.asp
- https://www.tutorialspoint.com/jquery/jquery-ajax.htm
- https://www.tutorialrepublic.com/jquery-tutorial/jquery-ajax.php

- O internacionalizaciji sadržaja statičkih html stranica videti više na
- **Internationalization Using jquery.i18n.properties.js**

https://dzone.com/articles/internationalization-using

URL - Podsetnik

Get ili Post

- GET je brži i jednostavniji od POST.
- Semantika:
  - GET – dobavljanje podataka. Više uzastopnih poziva GET metode bi uvek trebalo da ima isti rezultat. Browseri keširaju rezultate GET metode u cilju veće efikasnosti.
  - POST – osvežavanje informacija na serveru. Rezultati uzastopnih POST operacija se međusobno razlikuju. Poziv POST metode će uvek uzimati odgovor od servera (nema keširanja rezultata prethodnog upita).

Get ili Post

- Koristiti POST u sledećim situacijama:
  - Keširanje fajla nije opcija (osvežavanja fajla ili baze podataka na serveru);
  - Slanje velike količine podataka (POST nema ograničenja veličine);
  - Slanje korisničkog inputa – POST je robusniji i sigurniji.

jQuery i Ajax

Primer sa Servletima

- Korišćenje Jquery i Ajax sa Servlet tehnologojom možete videti u Biblioteka.zip projektu.