Experiences on a Design Approach for Interactive Web Applications

Experiences on a Design Approach for Interactive Web Applications

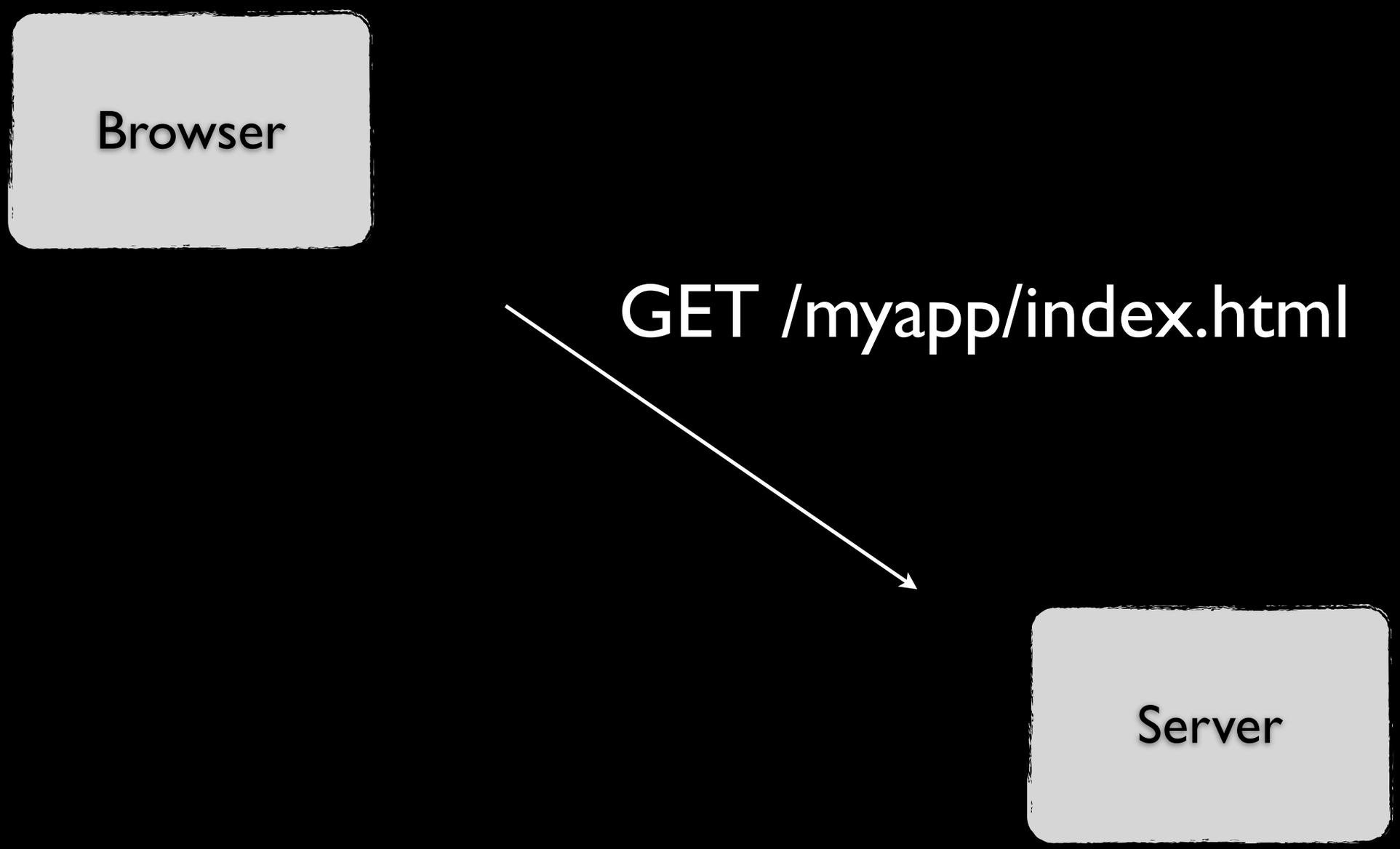
Janne Kuuskeri
Tampere University of Technology
Stratagen Systems

Current

Browser

Server

Current



Current

Browser

GET /myapp/userlist.html

Server

Current

Browser

POST /myapp/userdata

Server

Current

Browser

- resume app state
- execute app logic
- update model
- find view
- update session
- generate page

Server

Current

Browser

Current

Browser

Java

Current

Browser

HTML

Java

Current

Browser

JavaScript

HTML

Java

Current

Browser

CSS

JavaScript

HTML

Java

Current

Browser

JavaScript
CSS
HTML
Java
JSP

Current

Browser

JavaScript
CSS
HTML
Java

JSP

HTML page

Current

Java

```
String hql = "from Person p where p.age > :age";
Query q = session.createQuery();
q.setInteger("age", 33);
List people = q.list(hql);
```

Browser

JSP

```
<table>
<% for (Person p : people) { %>
  <tr>
     <td>p.getName()</td>
     <td>p.getAge()</td>
  </tr>
<% } %>
</table>
```

JavaScript

CSS

JSP

HTML

Java

Current

JSP

```
<%
if (request.getParameter("age") != null) {
  age=Integer.parseInt(request.getParameter(“age”));
}

String hql = "from Person p where p.age > :age";
Query q = session.createQuery();
q.setInteger("age", age);
List people = q.list(hql);
%>

<table>
<% for (Person p : people) { %>
  <tr>
    <td>p.getName()</td>
    <td>p.getAge()</td>
  </tr>
<% } %>
</table>
```

Browser

JavaScript

CSS

HTML

Java

JSP

JavaScript

Current

```
var onClick = function () {
  $.ajax({
    url: '/people/',
    dataType: 'json',
    data: { age: 33 },
    success: function (data) {
      var rows = [];
      $.each(data, function (person) {
        rows.push('<tr><td>' + person.name +
          '</td><td>' + person.age +'</td></tr>');
      }
      $('#peopletable > tbody:last').append(
        ''.join(rows));
    }
}
```

Browser

JavaScript

CSS

HTML

JSP

Java

MVC

MVC

Browser

View

Server

Controller, Model

MVC

Browser

View
Controller, Model

Server

?

MVC

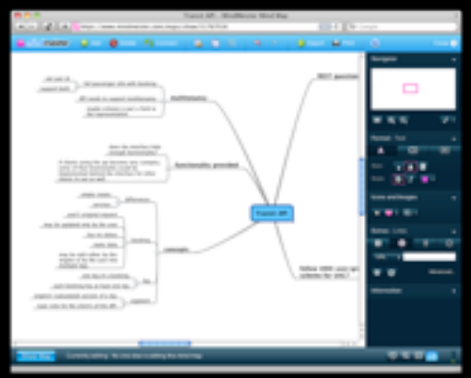
Browser

View
Controller, Model

Server

REST

MVC



Single page web application

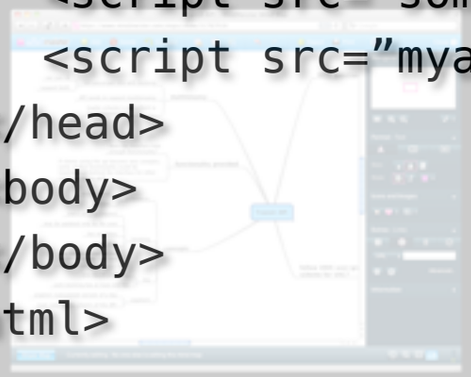
Server

REST

HTML

MVC

```
<html>
  <head>
    <link type="text/css" href="someframework.css">
     <script src=”someframework.js”></script>
     <script src=”myapp.js”></script>
  </head>
  <body>
  </body>
</html>
```

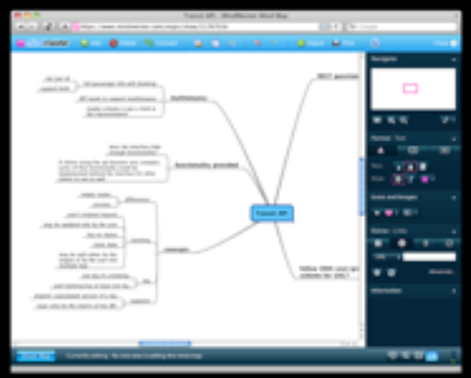


Single page web application

Server

REST

MVC

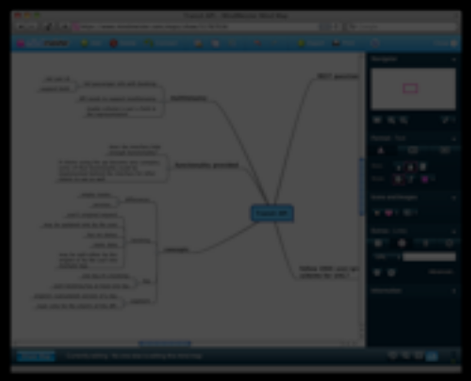


Single page web application

Server

REST

MVC



Single page web application

Server

REST

Case: Valvomo

Browser

Valvomo

Server

Transit API

Paratransit System

Browser

Valvomo

Single page UI

Server

Transit API

RESTful API

Paratransit System

Browser

Valvomo Web Application

HTML

Server

Transit API

Valvomo Web Application

Paratransit System

Browser

Valvomo

JSON

Server

Transit API

Paratransit System

DEMO

Pros

- Accessible API

- browsers
- mobile devices
- programmatic clients
- HTTP

Pros

- Accessible API
- Reusable API

- can have several different server implementations
- same UI can be placed “on top” of another server implementation as long as it is the same API
- no predefined application flow

Pros

- Accessible API
- Reusable API
- Decoupling

- responsibilities are easier to assign
- data validation
- error handling
- localization

Pros

- Accessible API
- Reusable API
- Decoupling
- Application flow / state

- unambiguously assign responsibilities for state handling
- server is responsible for resources and their states
- application (client) is responsible for the application flow and state

Pros

- Accessible API
- Reusable API
- Decoupling
- Application flow / state
- Development model

- fewer technologies to worry about
- one language for the client
- one language (maybe even the same) for the server
- client and server are easier to develop in parallel

Pros

- Accessible API
- Reusable API
- Decoupling
- Application flow / state
- Development model
- Testing

• client and server may be tested separately

• server API may be tested using isolated HTTP request and checking the result codes and content

• UI may be tested without the server or the network traffic

Pros

- Accessible API
- Reusable API
- Decoupling
- Application flow / state
- Development model
- Testing
- Network traffic

• REST offers minimal HTTP overhead

• network is utilized only when necessary (no page reloads)

• no data (e.g. HTML or CSS) overhead when transferring only payload data (e.g. JSON or XML)

Cons

- Framework support

- some server frameworks have REST support
- MVC support on client depends solely on the chosen toolkit

Cons

- Framework support
- Search engine support

- single page web applications are difficult to crawl and index
- RESTful API could be crawled and indexed but difficult to rank

Cons

- Framework support
- Search engine support
- Accessibility

• highly dynamic JavaScript UIs require extra work to be accessible for the visually impaired

• some JavaScript toolkits offer support for accessibility

Future work

- Authentication
- Flexible authorization configuration
  - per resource
  - per request method
  - representations per auth level?

Thank You