REliable, INtelligent & Scalable Systems

互联网应用开发技术

Web Application Development

第9课 WEB前后台通信-AJAX & JSON

Episode Nine

AJAX & JSON

陈昊鹏

chen-hp@sjtu.edu.cn

Web Application Development

Contents

Ajax

- AJAX = Asynchronous JavaScript and XML

  异步很关键的

- Traditionally webpages required reloading to update their content.
  - For web-based email this meant that users had to manually reload their inbox to check and see if they had new mail.
  - This had huge drawbacks: it was slow and it required user input.
  - When the user reloaded their inbox, the server had to reconstruct the entire web page and resend all of the HTML, CSS, JavaScript, as well as the user's email.
  - This was hugely inefficient.

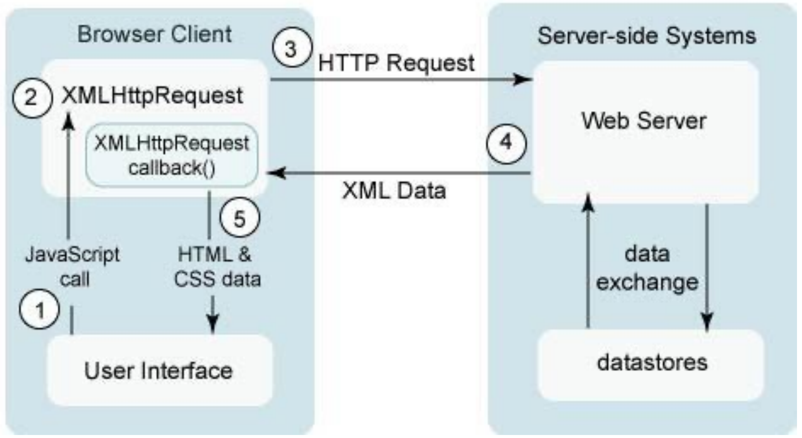
Ajax

- Ideally, the server should only have to send the user's new messages, not the entire page.
  - By 2003, all the major browsers solved this issue by adopting the XMLHttpRequest (XHR) object, allowing browsers to communicate with the server without requiring a page reload.
  - The XMLHttpRequest object is part of Ajax.

- Using Ajax, data could then be passed between the browser and the server, using the XMLHttpRequest API, without having to reload the web page.

Ajax

- Ajax requests are triggered by JavaScript code;
  - your code sends a request to a URL, and when it receives a response, a callback function can be triggered to handle the response.
  - Because the request is asynchronous, the rest of your code continues to execute while the request is being processed, so it‘s imperative that a callback be used to handle the response.



Ajax -XMLHttpRequest

- XMLHttpRequest
  - It is the basis of Ajax
  - All modern browsers, such as IE7+, Firefox, Chrome, Safari and Opera, have built-in `XMLHttpRequest`
  - In IE 5 and IE 6, it is `ActiveXObject`

- Obtain `XMLHttpRequest`

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

- Or in IE 5 and IE 6

```
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

Ajax -XMLHttpRequest

- Obtain XMLHttpRequest

```
var xmlhttp;
if(window.XMLHttpRequest) {
   xmlHttpRequest = new XMLHttpRequest();
} else if(window.ActiveXObject) {
   xmlHttpRequest = new ActionXObject("Microsoft.XMLHTTP");
}
```

Ajax - Send

- Send request to Server

```
xmlhttp.open("GET","test1.txt",true);
xmlhttp.send();

open(method,url,async)
    method: type of request, GET or POST
    url: location of required file
    async: true or false

send(string)
    string: only used in POST
```

用于POST传递url中的参数

Ajax - Get or Post ?

- The two most common "methods" for sending a request to a server are GET and POST.

- The GET method should be used for non-destructive operations
  - that is, operations where you are only "getting" data from the server, not changing data on the server.
  - For example, a query to a search service might be a GET request.
  - GET requests may be cached by the browser, which can lead to unpredictable behavior if you are not expecting it.
  - GET requests generally send all of their data in a query string.

- The POST method should be used for destructive operations
  - that is, operations where you are changing data on the server.
  - For example, a user saving a blog post should be a POST request.
  - POST requests are generally not cached by the browser;
  - a query string can be part of the URL, but the data tends to be sent separately as post data.

Ajax - Get

- Get

```
xmlHttpRequest.open("GET","GetUser",true);
xmlHttpRequest.onreadystatechange = ajaxCall;
xmlHttpRequest.send();
```

自己写的回调函数

- Add information into request

```
xmlHttpRequest.open("GET","GetUser?id=1",true);
xmlHttpRequest.onreadystatechange = ajaxCall;
xmlHttpRequest.send();
```

The information can be retrieved as request parameters

Ajax - Post

- Post

```
xmlHttpRequest.open("POST","GetUser",true);
xmlHttpRequest.setRequestHeader("Content-type",
          "application/x-www-form-urlencoded");
xmlHttpRequest.onreadystatechange = ajaxCall;
xmlHttpRequest.send("id=" + id);
```

- The information in send method can be retrieved as request parameters.

Ajax - onreadystatechange

- onreadystatechange
  - Onreadystatechange: the function invoked when `readyState` is changed.
  - readyState: the state of XMLHttpRequest, ranges from 0 to 4
  - status: 200-OK, 404-no page

```
xmlhttp.onreadystatechange=function() {
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
 {
    document.getElementById("myDiv").innerHTML=
        xmlhttp.responseText;
 }
}
```

Ajax - response

- XMLHttpRequest
  - responseText: response data in string.
  - responseXML: response data in XML

- responseText

```
document.getElementById("myDiv").innerHTML=
          xmlHttpRequest.responseText;
```

- responseXML

```
xmlDoc=xmlHttpRequest.responseXML;
txt="";
x=xmlDoc.getElementsByTagName("username");
for (i=0;i<x.length;i++) {
 txt=txt + x[i].childNodes[0].nodeValue + "<br />";
}
document.getElementById("myDiv").innerHTML=txt;
```

Ajax – an example

- Index.html

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Demo</title>
</head>
<body>
    <script src="js/jquery-2.1.0.js"></script>

    <script type="text/javascript">
      var xmlHttpRequest = null;
      function ajaxRequest() {
        if(window.ActiveXObject) {
            xmlHttpRequest = new ActionXObject("Microsoft.XMLHTTP");
        }
        else if(window.XMLHttpRequest) {
            xmlHttpRequest = new XMLHttpRequest();
        }
```

Ajax – an example

- Index.html

```
        if(xmlHttpRequest != null) {
            var id = document.getElementById("id").value;

            xmlHttpRequest.open("GET", "GetUser?id="+id, true);
            xmlHttpRequest.onreadystatechange = ajaxCall;
            xmlHttpRequest.send();
```

  – or

```
            xmlHttpRequest.open("POST","GetUser",true);
            xmlHttpRequest.setRequestHeader("Content-type",
                "application/x-www-form-urlencoded");
            xmlHttpRequest.onreadystatechange = ajaxCall;
            xmlHttpRequest.send("id=" + id);
         }
      }
```

Ajax – an example

- Index.html

```
  function ajaxCall() {
        if(xmlHttpRequest.readyState == 4 ) {
            if(xmlHttpRequest.status == 200) {
                var text = xmlHttpRequest.responseText;
                document.getElementById("myDiv").innerHTML =
                      "<h2>"+ text + "</h2>";
            }
        }
    }
    </script>

    <div id="myDiv"><h2>Let AJAX change this text</h2></div>
    User id: <input type="text" name="id" id="id" /> <br/>
    <button type="button" onclick="ajaxRequest()">
        Query
    </button>
  </body>
</html>
```

Ajax – an example

- UserServlet

```
package user;
@WebServlet("/GetUser")
public class UserServlet extends HttpServlet {
    private static final long serialVersionUID = 1892537774889413L;

    @Resource(name="jdbc/sample")
    DataSource ds;

    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        System.out.println("doGet invoked!");

        try {
            String id = (String) request.getParameter("id");
            Connection con = ds.getConnection();
            PreparedStatement ps = con.prepareStatement(
                    "SELECT * FROM tbl_user WHERE id = ?");
            ps.setString(1, id);
```

Ajax – an example

- UserServlet

```
        ResultSet rs = ps.executeQuery();
        rs.last();
        int count = rs.getRow();

        if ( count == 0) {
               out.println("no such user");
        } else{
               String s = "username: " + rs.getString(2) +
                          " email: " + rs.getString(2);
               out.println(s);
        }
       out.flush();
    } catch(Exception e){
       e.printStackTrace();
    }
    finally {
        out.close();
    }
}
```

Ajax – an example

- UserServlet

```
    @Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

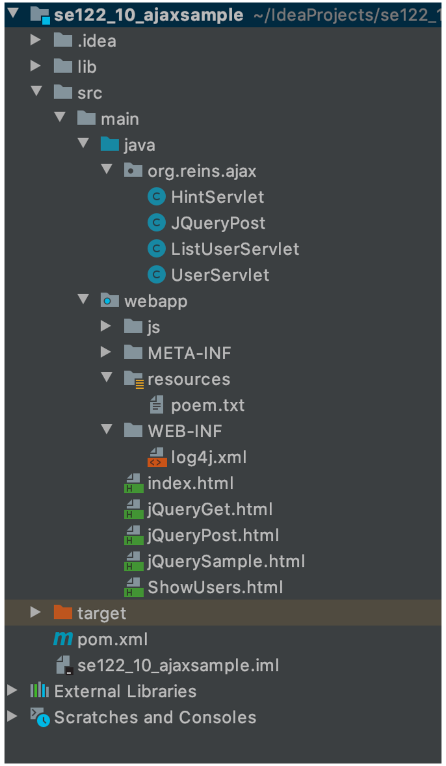
Ajax – an example

REliable, INtelligent & Scalable Systems

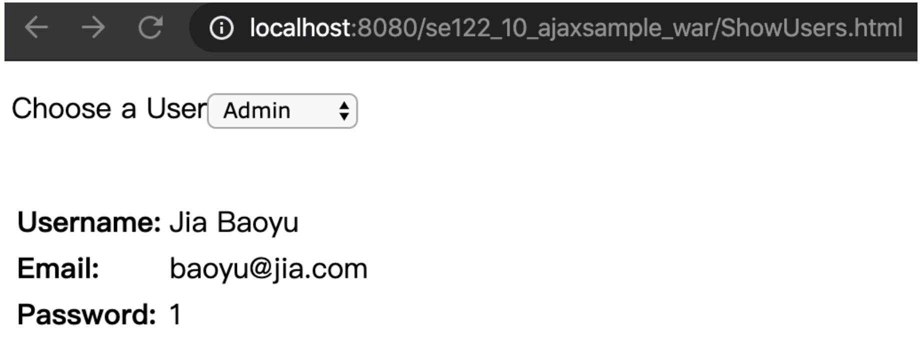
- Context.xml

```
<Context>
  <Resource name="jdbc/sample“
            auth="Container"
            type="javax.sql.DataSource"
            maxActive="100"
            maxIdle="30"
            maxWait="10000"
            username="root"
            password="reins2011!"
            driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/ajax"/>
</Context>
```

Ajax – an example



```
▼ se122_10_ajaxsample ~/IdeaProjects/se122_1
  ▶ .idea
  ▶ lib
  ▼ src
    ▼ main
      ▼ java
        ▼ org.reins.ajax
            HintServlet
            JQueryPost
            ListUserServlet
            UserServlet
      ▼ webapp
        ▶ js
        ▶ META-INF
        ▼ resources
            poem.txt
        ▼ WEB-INF
            log4j.xml
          index.html
          jQueryGet.html
          jQueryPost.html
          jQuerySample.html
          ShowUsers.html
  ▶ target
    pom.xml
    se122_10_ajaxsample.iml
▶ External Libraries
▶ Scratches and Consoles
```



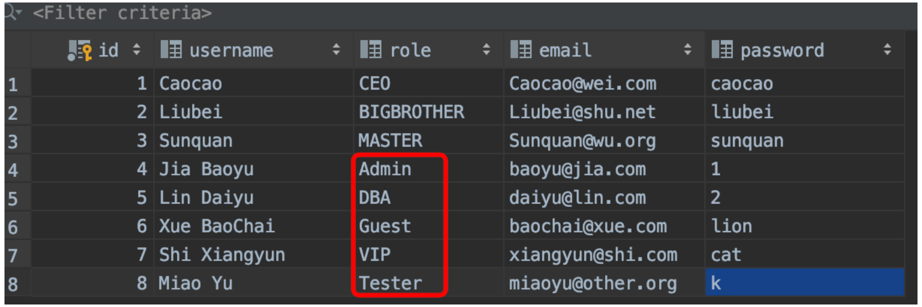
localhost:8080/se122_10_ajaxsample_war/ShowUsers.html

Choose a User [Admin]

Username: Jia Baoyu

Email: baoyu@jia.com

Password: 1



<Filter criteria>

	id	username	role	email	password
1	1	Caocao	CEO	Caocao@wei.com	caocao
2	2	Liubei	BIGBROTHER	Liubei@shu.net	liubei
3	3	Sunquan	MASTER	Sunquan@wu.org	sunquan
4	4	Jia Baoyu	Admin	baoyu@jia.com	1
5	5	Lin Daiyu	DBA	daiyu@lin.com	2
6	6	Xue BaoChai	Guest	baochai@xue.com	lion
7	7	Shi Xiangyun	VIP	xiangyun@shi.com	cat
8	8	Miao Yu	Tester	miaoyu@other.org	k

Ajax - Asyn

REliable, INtelligent & Scalable Systems

- Asyn = true

```
xmlHttpRequest.open("GET", "GetUser?id="+id, true);
xmlHttpRequest.onreadystatechange = ajaxCall;
xmlHttpRequest.send();
function ajaxCall() {
  if(xmlHttpRequest.readyState == 4 ) {
     if(xmlHttpRequest.status == 200) {
        var text = xmlHttpRequest.responseText;
          document.getElementById("myDiv").innerHTML =
                   "<h2>"+ text + "</h2>";
     }
  }
}
```

- Asyn = false

```
xmlHttpRequest.open("GET", "GetUser?id="+id, false);
xmlHttpRequest.send();
document.getElementById("myDiv").innerHTML =
           "<h2>"+ xmlHttpRequest.responseText + "</h2>";
```

Ajax - callback function

- Callback function

```
<html>
 <head>
  <script type="text/javascript">
   var xmlhttp;
   function loadXMLDoc(url,cfunc)
   {
      if (window.XMLHttpRequest)
      {// code for IE7+, Firefox, Chrome, Opera, Safari
         xmlhttp=new XMLHttpRequest();
      }
      else
      {// code for IE6, IE5
         xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
      }
      xmlhttp.onreadystatechange=cfunc;
      xmlhttp.open("GET",url,true);
      xmlhttp.send();
}
```

将前端发送请求的函数处理为接口方便复用

Ajax - callback function

- Callback function

```
   function myFunction()
   {
    loadXMLDoc("/ajax/test1.txt",function()
     {
      if (xmlhttp.readyState==4 && xmlhttp.status==200)
      {
       document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
      }
     });
    }
   </script>
  </head>
  <body>
    <div id="myDiv"><h2>Let AJAX change this text</h2></div>
    <button type="button" onclick="myFunction()">Change Content</button>
  </body>
 </html>
```

Ajax - another example

- Index.html(snippet)

```
User id: <input type="text" name="id" id="id"
   onkeyup="showHint(this.value)"/> <br/>
<p>Suggestion: <span id="txtHint"></span></p>

function showHint(str)
{
    var xmlhttp;

    if (str.length==0)
    {
      document.getElementById("txtHint").innerHTML="";
      return;
    }

    if (window.XMLHttpRequest)
      xmlhttp=new XMLHttpRequest();
    else
      xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

    xmlhttp.onreadystatechange=function()
    {
      if (xmlhttp.readyState==4 && xmlhttp.status==200)
        document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }

    xmlhttp.open("GET","Hint?q="+str,true);
    xmlhttp.send();
}
```

Ajax - another example

REliable, INtelligent & Scalable Systems

- HintServlet.java(snippet)

```
@WebServlet("/Hint")
public class HintServlet extends HttpServlet {
  protected void processRequest(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        String hint = "";
        String a[] = new String[30];
        a[0] = "Anna";  a[1] = "Brittany"; a[2] = "Cinderella";
        ……………
        String q = request.getParameter("q");
            if (q.length() > 0) {
             for (int i = 0; i < 30; i++)
              if (a[i].indexOf(q) >= 0)
                if (hint == "")
                  hint = a[i];
                   else
                  hint = hint + ", " + a[i];
             if (hint == "")
              out.println("no suggestion");
             else
              out.println(hint);

     }
```

Ajax - list user information

- ShowUser.html

```
<html>
<head>
  <script type="text/javascript">
   function showUsers(str) {
     var xmlhttp;
     if (str == "") {
       document.getElementById("txtHint").innerHTML = "";
       return;
     }
     if (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp = new XMLHttpRequest();
     } else {// code for IE6, IE5
       xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
     }
     xmlhttp.onreadystatechange = function() {
       if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
         document.getElementById("txtHint").innerHTML = xmlhttp.responseText;
       }
     }
     xmlhttp.open("GET", "ListUsers?q=" + str, true);
     xmlhttp.send();
   }
 </script>
</head>
```

Ajax - list user information

REliable, INtelligent & Scalable Systems

- ShowUser.html

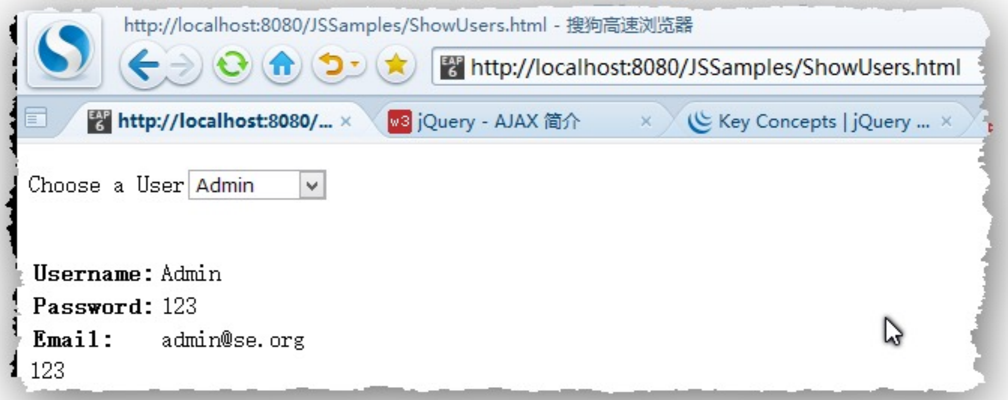
```
   <body>
      <form action="" style="margin-top: 15px;">
        <label>Choose a User
         <select name="users"
           onchange="showUsers(this.value)"
           style="font-family: Verdana, Arial, Helvetica, sans-serif;">
          <option value="Admin">Admin</option>
          <option value="DBA ">DBA</option>
          <option value="Guest">Guest</option>
          <option value="VIP">VIP</option>
          <option value="Tester">Tester</option>
          <option value="Developer">Developer</option>
         </select>
        </label>
      </form>
      <br />
     <div id="txtHint">Here's the information of chosen user:</div>
   </body>
 </html>
```

Ajax - list user information

- ListUserServlet.html

```
@WebServlet("/ListUsers")
public class ListUserServlet extends HttpServlet {
@Resource(name="jdbc/sample")
    DataSource ds;
    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)throws ServletException, IOException {
            PrintWriter out = response.getWriter();

            String username = (String) request.getParameter("q");
            Connection con = ds.getConnection();
            PreparedStatement ps = con.prepareStatement("SELECT * FROM tbl_user WHERE username = ?");
            ps.setString(1, username);
            ResultSet rs = ps.executeQuery();
            rs.last();
            int count = rs.getRow();

            if ( count == 0) {
            out.println("no such user");
            } else{
               out.println("<table>");
               out.println("<tr><td><b> Username: </b></td>" + "<td>"+ rs.getString(2) + "</td></tr>");
               out.println("<tr><td><b> Password: </b></td>" + "<td>"+ rs.getString(3) + "</td></tr>");
               out.println("<tr><td><b> Email: </b></td>" + "<td>"+ rs.getString(4) + "</td></tr>");
               out.write("</table>");
            }
     }
```

Ajax - list user information


http://localhost:8080/JSSamples/ShowUsers.html - 搜狗高速浏览器
http://localhost:8080/JSSamples/ShowUsers.html
http://localhost:8080/...
jQuery - AJAX 简介
Key Concepts | jQuery ...

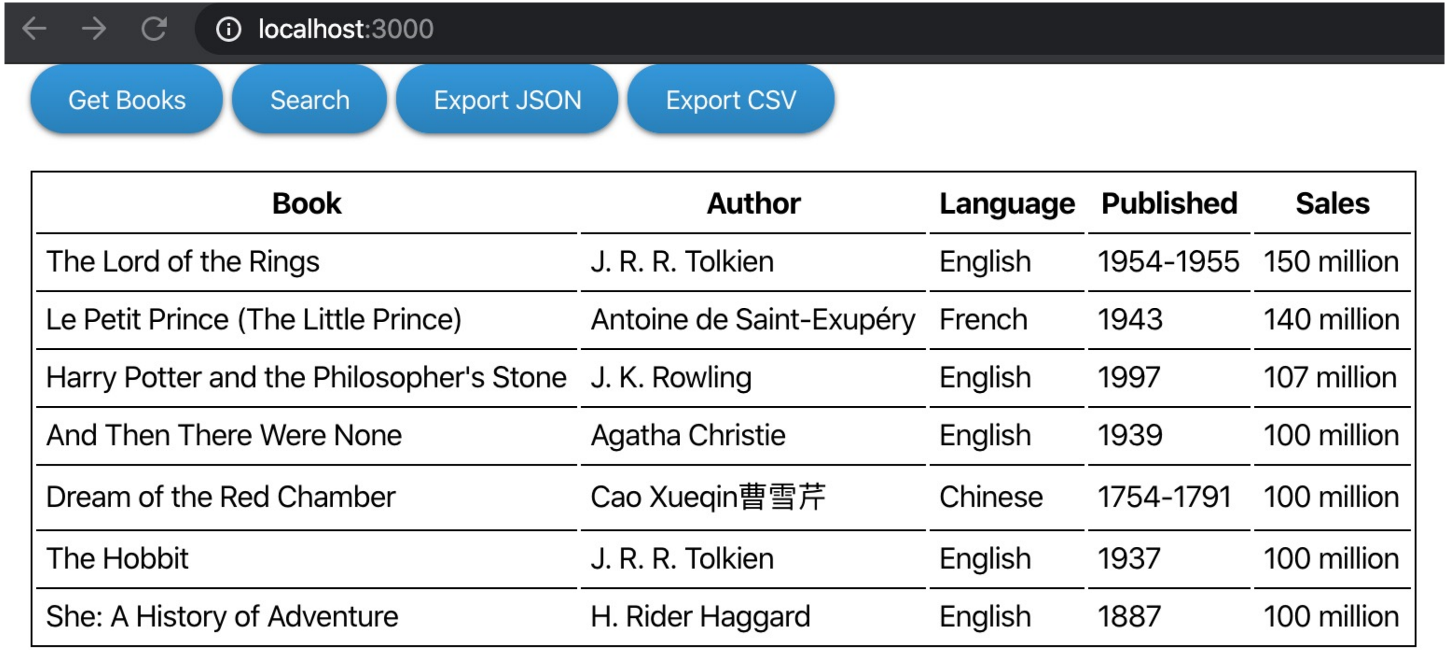
Choose a User Admin

Username: Admin
Password: 123
Email: admin@se.org
123

React: Ajax & JSON

REliable, INtelligent & Scalable Systems

- Ajax & JSON in React



Book	Author	Language	Published	Sales
The Lord of the Rings	J. R. R. Tolkien	English	1954-1955	150 million
Le Petit Prince (The Little Prince)	Antoine de Saint-Exupéry	French	1943	140 million
Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	107 million
And Then There Were None	Agatha Christie	English	1939	100 million
Dream of the Red Chamber	Cao Xueqin曹雪芹	Chinese	1754-1791	100 million
The Hobbit	J. R. R. Tolkien	English	1937	100 million
She: A History of Adventure	H. Rider Haggard	English	1887	100 million

React: Ajax & JSON

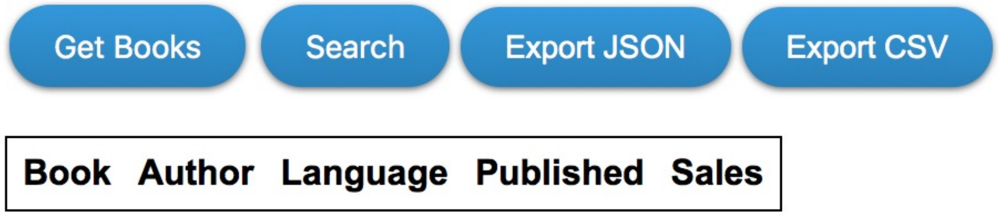
- App.js

```
const data = [];
class Excel extends React.Component {
  getBooks = () => {
    fetch("http://localhost:8080/se122_10_reactdb_war/BookManager")
      .then(response => response.json())
      .then(data => {
        // alert("data:" + data);
        this.setState({
          data: data,
        });
      }).catch(function (ex) {
      console.log('parsing failed', ex)
    })
  }
  renderToolbar = () => {
    return (
      <div className="toolbar">
        <button onClick={this.getBooks}>Get Books</button>
        <button onClick={this.toggleSearch}>Search</button>
        <a onClick={this.download.bind(this, 'json')}
          href="data.json">Export JSON</a>
        <a onClick={this.download.bind(this, 'csv')}
          href="data.csv">Export CSV</a>
      </div>
    );
  };
```

React: Ajax & JSON

REliable, INtelligent & Scalable Systems

- Initial State

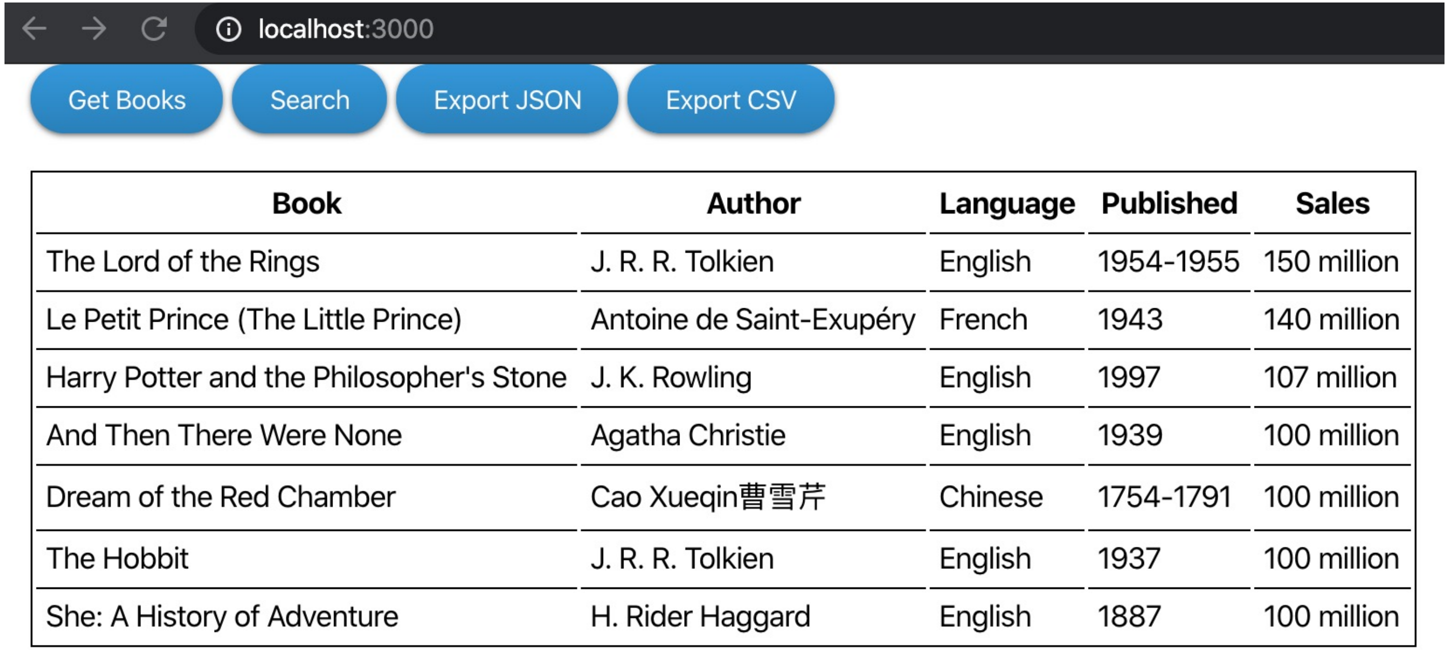


Book	Author	Language	Published	Sales

React: Ajax & JSON

REliable, INtelligent & Scalable Systems

- After click the button “Get Books”



Book	Author	Language	Published	Sales
The Lord of the Rings	J. R. R. Tolkien	English	1954-1955	150 million
Le Petit Prince (The Little Prince)	Antoine de Saint-Exupéry	French	1943	140 million
Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	107 million
And Then There Were None	Agatha Christie	English	1939	100 million
Dream of the Red Chamber	Cao Xueqin曹雪芹	Chinese	1754-1791	100 million
The Hobbit	J. R. R. Tolkien	English	1937	100 million
She: A History of Adventure	H. Rider Haggard	English	1887	100 million

React: Ajax & JSON

REliable, INtelligent & Scalable Systems

- Book.java

```
@Entity
@Table(name = "book")
public class Book {

  private Long id;

   private String title;
   private String author;
   private String language;
   private String published;
   private String sales;

   public Book() {}

   @Id
   @GeneratedValue(generator = "increment")
   @GenericGenerator(name = "increment",
                                 strategy = "increment")
   public Long getId() {
      return id;
   }
   public void setId(Long id) {
      this.id = id;
   }

public String getTitle() {      return title;    }
public void setTitle(String title) {      this.title = title;    }

public String getAuthor() {      return author;    }
public void setAuthor(String author) {
   this.author = author;
}

public String getLanguage() {      return language;    }
public void setLanguage(String language) {
   this.language = language;
}

public String getPublished() {      return published;    }
public void setPublished(String published) {
   this.published = published;
}

public String getSales() {      return sales;    }
public void setSales(String sales) {
   this.sales = sales;
}

}
```

React: Ajax & JSON

REliable, INtelligent & Scalable Systems

- BookManagerServlet.java

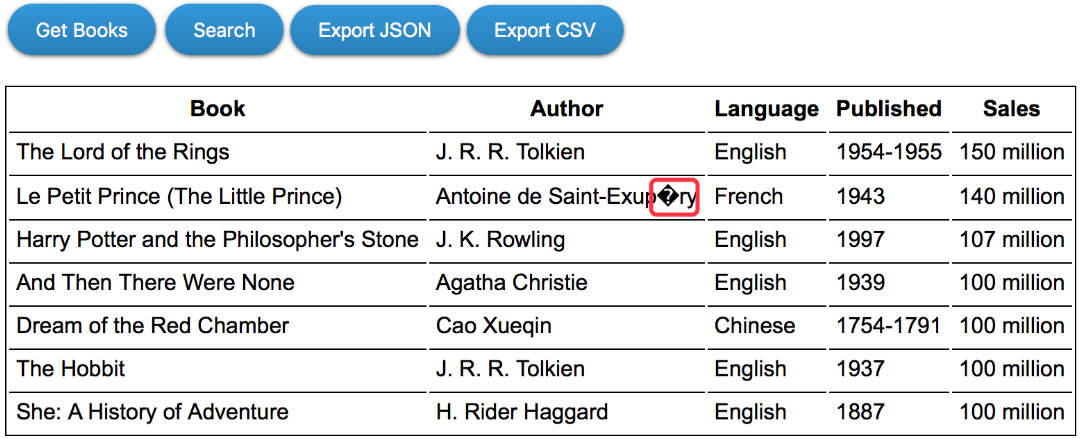
```
@WebServlet("/BookManager")
public class BookManagerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public BookManagerServlet() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException {
        try {
            HibernateUtil.getSessionFactory().getCurrentSession().beginTransaction();
            PrintWriter out = response.getWriter();
            response.setContentType("application/json; charset=UTF-8");
            response.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");


            System.out.println("This is a book manager");

            Session session = HibernateUtil.getSessionFactory().getCurrentSession();
            List<Book> result = session.createQuery("from Book").list();
            Iterator<Book> it = result.iterator();
```

React: Ajax & JSON

- BookManagerServlet.java

```
        ArrayList<JSONArray> booksJson = new ArrayList<JSONArray>();
        while (it.hasNext()) {
            Book book = (Book) it.next();
            ArrayList<String> arrayList = new ArrayList<String>();
            arrayList.add(book.getTitle());
            arrayList.add(book.getAuthor());
            arrayList.add(book.getLanguage());
            arrayList.add(book.getPublished());
            arrayList.add(book.getSales());
            booksJson.add((JSONArray) JSONArray.toJSON(arrayList));
        }
        String  booksString = JSON.toJSONString(booksJson, SerializerFeature.BrowserCompatible)
        System.out.println(booksString);
        session.getTransaction().commit();
        out.println(booksString);
        out.flush();
        out.close();
    } catch (Exception ex) {
        HibernateUtil.getSessionFactory().getCurrentSession().getTransaction().rollback();
        if (ServletException.class.isInstance(ex)) {
            throw (ServletException) ex;
        } else {
            throw new ServletException(ex);
        }
    }
}
```

React: Ajax & JSON

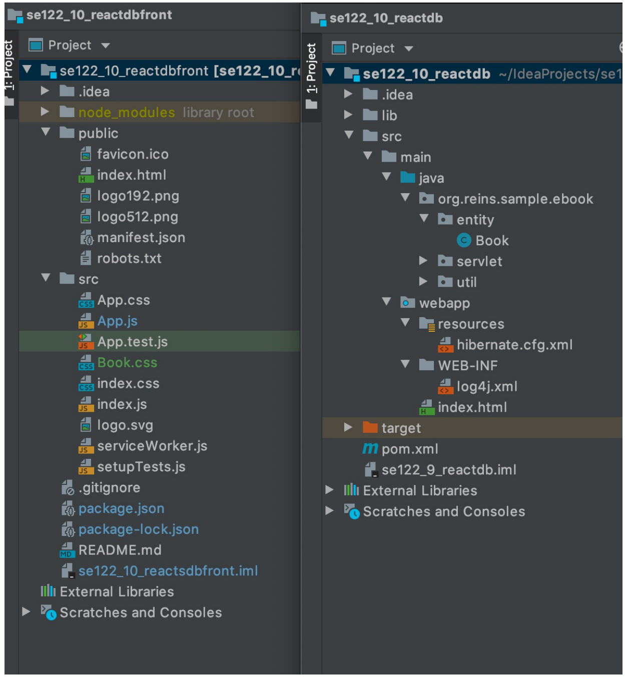


Book	Author	Language	Published	Sales
The Lord of the Rings	J. R. R. Tolkien	English	1954-1955	150 million
Le Petit Prince (The Little Prince)	Antoine de Saint-Exup�ry	French	1943	140 million
Harry Potter and the Philosopher's Stone	J. K. Rowling	English	1997	107 million
And Then There Were None	Agatha Christie	English	1939	100 million
Dream of the Red Chamber	Cao Xueqin	Chinese	1754-1791	100 million
The Hobbit	J. R. R. Tolkien	English	1937	100 million
She: A History of Adventure	H. Rider Haggard	English	1887	100 million

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.serializer.SerializerFeature;

response.setContentType("application/json; charset=UTF-8");

String  booksString = JSON.toJSONString(booksJson, SerializerFeature.BrowserCompatible);
```

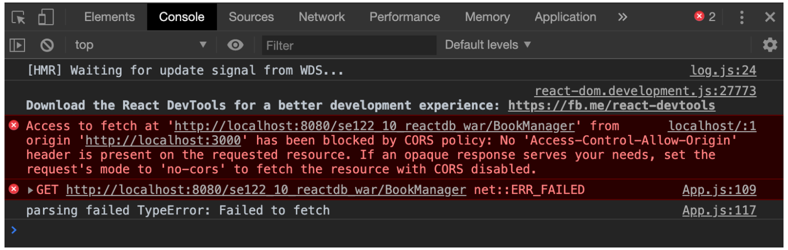
React: Ajax & JSON

REliable, INtelligent & Scalable Systems



```
se122_10_reactdbfront
Project
▼ se122_10_reactdbfront [se122_10_r
  ▶ .idea
  ▶ node_modules  library root
  ▼ public
      favicon.ico
      index.html
      logo192.png
      logo512.png
      manifest.json
      robots.txt
  ▼ src
      App.css
      App.js
      App.test.js
      Book.css
      index.css
      index.js
      logo.svg
      serviceWorker.js
      setupTests.js
    .gitignore
    package.json
    package-lock.json
    README.md
    se122_10_reactsdbfront.iml
  External Libraries
▶ Scratches and Consoles
```

```
se122_10_reactdb
Project
▼ se122_10_reactdb  ~/IdeaProjects/se1
  ▶ .idea
  ▶ lib
  ▼ src
    ▼ main
      ▼ java
        ▼ org.reins.sample.ebook
          ▼ entity
              Book
          ▶ servlet
          ▶ util
      ▼ webapp
        ▼ resources
            hibernate.cfg.xml
        ▼ WEB-INF
            log4j.xml
          index.html
  ▶ target
    pom.xml
    se122_9_reactdb.iml
▶ External Libraries
▶ Scratches and Consoles
```



```
[HMR] Waiting for update signal from WDS...                                    log.js:24
                                                          react-dom.development.js:27773
Download the React DevTools for a better development experience: https://fb.me/react-devtools
Access to fetch at 'http://localhost:8080/se122_10_reactdb_war/BookManager' from      localhost/:1
origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin'
header is present on the requested resource. If an opaque response serves your needs, set the
request's mode to 'no-cors' to fetch the resource with CORS disabled.
▸GET http://localhost:8080/se122_10_reactdb_war/BookManager net::ERR_FAILED          App.js:109
parsing failed TypeError: Failed to fetch                                            App.js:117
```

No 'Access-Control-Allow-Origin' header is present on the requested resource'

Same Origin Policy

BookManagerServlet.java

```
response.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
```

JSON

- JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.

- JSON defines only two data structures: objects and arrays.
  - An object is a set of name-value pairs, and an array is a list of values.
  - JSON defines six data types: *string, number, object, array, true, false and null*

JSON

- The following example shows JSON data for a sample object that contains name-value pairs.

```
{
 "firstName": "Duke",
 "lastName": "Java",
 "age": 18,
 "streetAddress": "100 Internet Dr",
 "city": "JavaTown",
 "state": "JA",
 "postalCode": "12345",
 "phoneNumbers": [
    { "Mobile": "111-111-1111" },
    { "Home": "222-222-2222" }
  ]
 }
```

JSON Syntax

- JSON has the following syntax:
  - Objects are enclosed in braces ({ }),
    - their name-value pairs are separated by a comma (,),
    - and the name and value in a pair are separated by a colon (:).
    - Names in an object are strings, whereas values may be of any of the six data types, including another object or an array.
  - Arrays are enclosed in brackets ([ ]),
    - and their values are separated by a comma (,).
    - Each value in an array may be of a different type, including another array or an object.
  - When objects and arrays contain other objects or arrays, the data has a tree-like structure.

Uses of JSON

- JSON is often used as a common format to
  - serialize and deserialize data in applications that communicate with each other over the Internet.

- The HTTP header used to
  - indicate that the content of a request or a response is JSON data is the following:

```
Content-Type: application/json
```

- For generating and parsing JSON data,
  - there are two programming models, which are similar to those used for XML documents:
    - The object model creates a tree that represents the JSON data in memory.
    - The streaming model uses an event-based parser that reads JSON data one element at a time.

JSON Processing in the Java EE

- The Java API for JSON Processing contains the following packages:
  - The `javax.json` package contains a reader interface, a writer interface, and a model builder interface for the object model. This package also contains other utility classes and Java types for JSON elements.
  - The `javax.json.stream` package contains a parser interface and a generator interface for the streaming model.

  - In `javax.json`
    - `Json, JsonReader, JsonObjectBuilder, JsonArrayBuilder, JsonWriter, JsonValue, JsonStructure, JsonObject, JsonArray, JsonString, JsonNumber, JsonException`
  - In `javax.json.stream`
    - `JsonParser, JsonGenerator`

Using the Object Model API

- Creating an Object Model from JSON Data

```
import java.io.FileReader;
import javax.json.Json;
import javax.json.JsonReader;
import javax.json.JsonStructure;
...
JsonReader reader = Json.createReader(new FileReader("jsondata.txt"));
JsonStructure jsonst = reader.read();
```

- The object reference jsonst can be
  - either of type JsonObject or of type JsonArray, depending on the contents of the file.
  - JsonObject and JsonArray are subtypes of JsonStructure.

Using the Object Model API

- Creating an Object Model from Application Code

```
import javax.json.Json;
import javax.json.JsonObject;
...
JsonObject model = Json.createObjectBuilder()
  .add("firstName", "Duke")
  .add("lastName", "Java")
  .add("age", 18)
  .add("streetAddress", "100 Internet Dr")
  .add("city", "JavaTown")
  .add("state", "JA")
  .add("postalCode", "12345")
  .add("phoneNumbers", Json.createArrayBuilder()
     .add(Json.createObjectBuilder()
         .add("type", "mobile")
         .add("number", "111-111-1111"))
     .add(Json.createObjectBuilder()
         .add("type", "home")
         .add("number", "222-222-2222")))
  .build();
```

Using the Object Model API

- Navigating an Object Model

```
public static void navigateTree(JsonValue tree, String key) {
 if (key != null)
   System.out.print("Key " + key + ": ");
 switch(tree.getValueType()) {
   case OBJECT:
     System.out.println("OBJECT");
     JsonObject object = (JsonObject) tree;
     for (String name : object.keySet())
       navigateTree(object.get(name), name);
     break;
   case ARRAY:
     System.out.println("ARRAY");
     JsonArray array = (JsonArray) tree;
     for (JsonValue val : array)
       navigateTree(val, null);
     break;
   case STRING:
     JsonString st = (JsonString) tree;
     System.out.println("STRING " + st.getString());
     break;
   case NUMBER:
     JsonNumber num = (JsonNumber) tree;
     System.out.println("NUMBER " + num.toString());
     break;
   case TRUE:
   case FALSE:
   case NULL:
     System.out.println(tree.getValueType().toString());
     break;
 }
}
```

Using the Object Model API

- Writing an Object Model to a Stream

```
StringWriter stWriter = new StringWriter();
try (JsonWriter jsonWriter = Json.createWriter(stWriter))
{
  jsonWriter.writeObject(model);
}

String jsonData = stWriter.toString();
System.out.println(jsonData);
```

Using the Streaming API

- Reading JSON Data Using a Parser

```
JsonParser parser = Json.createParser(new StringReader(jsonData));
while (parser.hasNext()) {
 JsonParser.Event event = parser.next();
 switch(event) {
   case START_ARRAY:
   case END_ARRAY:
   case START_OBJECT:
   case END_OBJECT:
   case VALUE_FALSE:
   case VALUE_NULL:
   case VALUE_TRUE:
     System.out.println(event.toString());
     break;
   case KEY_NAME:
     System.out.print(event.toString() + " " + parser.getString() + " - ");
     break;
   case VALUE_STRING:
   case VALUE_NUMBER:
     System.out.println(event.toString() + " " + parser.getString());
     break;
 }
```

Using the Streaming API

REliable, INtelligent & Scalable Systems

- Writing JSON Data Using a Generator

```
FileWriter writer = new FileWriter("test.txt");
JsonGenerator gen = Json.createGenerator(writer);
gen.writeStartObject()
 .write("firstName", "Duke")
 .write("lastName", "Java")
 .write("age", 18)
 .write("streetAddress", "100 Internet Dr")
 .write("city", "JavaTown")
 .write("state", "JA")
 .write("postalCode", "12345")
 .writeStartArray("phoneNumbers")
   .writeStartObject()
     .write("type", "mobile")
     .write("number", "111-111-1111")
   .writeEnd()
   .writeStartObject()
     .write("type", "home")
     .write("number", "222-222-2222")
   .writeEnd()
  .writeEnd()
.writeEnd();
gen.close();
```

An example

- index.html

```
……
<form action="" style="margin-top: 15px;">
<table>
  <tr>
        <td align="right">First Name:</td>
        <td><input type="text" name="firstname" id="firstname" size=20 /></td>
  </tr>
  ……
  <tr>
        <td align="right">Phone Number 1:</td>
        <td>
          <input type="text" name="phoneNumber1" id="phoneNumber1" size=20 />
          <select name="phoneType1" id="phoneType1">
                  <option value="Home">Home</option>
                  <option value="Mobile">Mobile</option>
          </select>
        </td>
  </tr>
  ……
</table>
<p>
  <button type="button" onclick="ajaxRequest()">Create a JSON Object</button>
</p>
<textarea id="textarea" cols="70" rows="20"></textarea>
</form>
```

An example

REliable, INtelligent & Scalable Systems

- index.html

```
var infoMsg = new Object();

infoMsg.firstname = document.getElementById("firstname").value;
……
var phone = new Object();
var phoneType1 = document.getElementById("phoneType1").
        options[document.getElementById("phoneType1").selectedIndex].text;
var phoneNumber1 = document.getElementById("phoneNumber1").value;
phone[phoneType1] = phoneNumber1;
var phoneType2 = document.getElementById("phoneType2").
        options[document.getElementById("phoneType2").selectedIndex].text;
var phoneNumber2 = document.getElementById("phoneNumber2").value;
phone[phoneType2] = phoneNumber2;

infoMsg.phoneNumbers = phone;

var jsonstr = JSON.stringify(infoMsg);

xmlHttpRequest.open("POST","JsonServlet",true);
xmlHttpRequest.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlHttpRequest.onreadystatechange = ajaxCall;
xmlHttpRequest.send("content=" + jsonstr);

console += "Sent: " + jsonstr + "\n";
document.getElementById("textarea").innerHTML = console;
```

An example

- JsonServlet.java

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

    String content = request.getParameter("content");
    parseJson(content);

    PrintWriter out = response.getWriter();
    out.println(buildJson());
}

public void parseJson(String content) {
    try (JsonReader reader =
            Json.createReader(new StringReader(content))) {
        parsed = reader.readObject();
    }
    this.printTree(parsed, 0, "");
}
```

An example

REliable, INtelligent & Scalable Systems

- JsonServlet.java

```
public void printTree(JsonValue tree, int level, String key) {
    switch (tree.getValueType()) {
      case OBJECT:
        JsonObject object = (JsonObject) tree;
        System.out.println( level + " " +
                tree.getValueType().toString()  + " " +  key + "--");
        for (String name : object.keySet()) {
          this.printTree(object.get(name), level+1, name);
        }
        break;
      case ARRAY:
        JsonArray array = (JsonArray) tree;
                        System.out.println( level + " " +
                tree.getValueType().toString() + " " + key + "--");
        for (JsonValue val : array) {
          this.printTree(val, level+1, "");
        }
        break;
          …………
    }
}
```

An example

- JsonServlet.java

```
public String buildJson() {
  JsonObject model = Json.createObjectBuilder()
      .add("firstName", "Tom")
      .add("lastName", "Jerry")
      .add("age", 10)
      .add("streetAddress", "Disney Avenue")
      .add("city", "los angles")
      .add("state", "CA")
      .add("postalCode", "12345")
      .add("phoneNumbers", Json.createArrayBuilder()
        .add(Json.createObjectBuilder()
          .add("number", "911")
          .add("type", "HOME"))
        .add(Json.createObjectBuilder()
          .add("number", "110")
          .add("type", "OFFICE")))
    .build();
```

An example

REliable, INtelligent & Scalable Systems

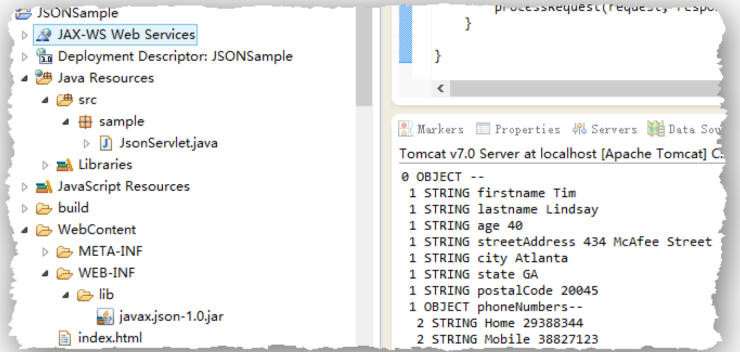
- JsonServlet.java

```
StringWriter stWriter = new StringWriter();
try (JsonWriter jsonWriter = Json.createWriter(stWriter)) {
   jsonWriter.writeObject(model);
}
return stWriter.toString();
```

- or

```
    /* Write formatted JSON Output */
    Map<String,String> config = new HashMap<>();
    config.put(JsonGenerator.PRETTY_PRINTING, "");
    JsonWriterFactory factory = Json.createWriterFactory(config);

    StringWriter stWriterF = new StringWriter();
    try (JsonWriter jsonWriterF = factory.createWriter(stWriterF)) {
       jsonWriterF.writeObject(model);
    }

    return stWriterF.toString();
}
```

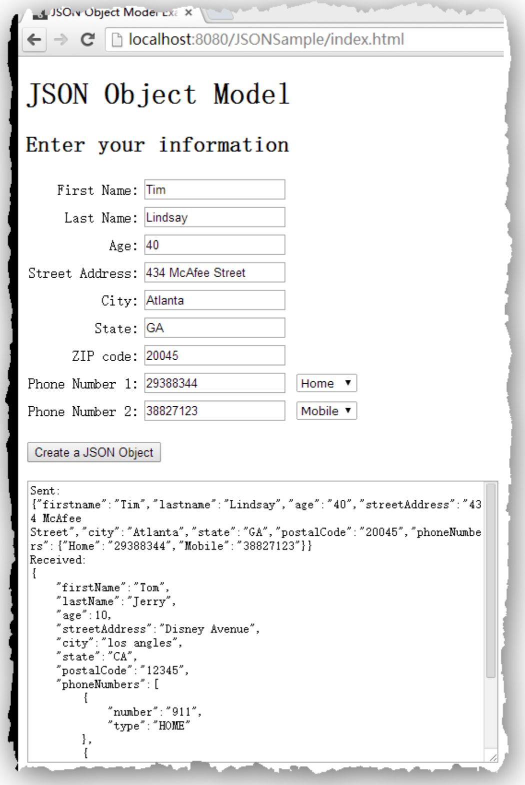
An example



```
processRequest(request, respo
        }
    }
```

Tomcat v7.0 Server at localhost [Apache Tomcat] C:

```
0 OBJECT --
 1 STRING firstname Tim
 1 STRING lastname Lindsay
 1 STRING age 40
 1 STRING streetAddress 434 McAfee Street
 1 STRING city Atlanta
 1 STRING state GA
 1 STRING postalCode 20045
 1 OBJECT phoneNumbers--
  2 STRING Home 29388344
  2 STRING Mobile 38827123
```



JSON Object Model

Enter your information

First Name:	Tim
Last Name:	Lindsay
Age:	40
Street Address:	434 McAfee Street
City:	Atlanta
State:	GA
ZIP code:	20045
Phone Number 1:	29388344 Home
Phone Number 2:	38827123 Mobile

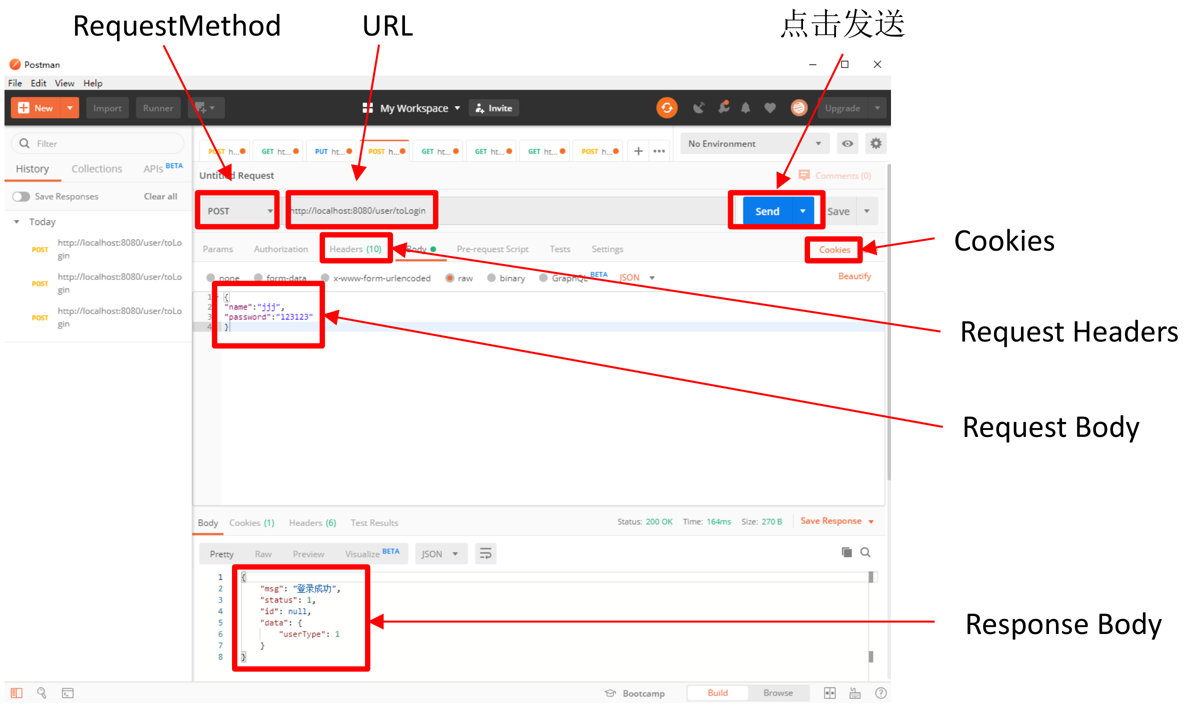
```
Sent:
{"firstname":"Tim","lastname":"Lindsay","age":"40","streetAddress":"43
4 McAfee
Street","city":"Atlanta","state":"GA","postalCode":"20045","phoneNumbe
rs":{"Home":"29388344","Mobile":"38827123"}}
Received:
{
    "firstName":"Tom",
    "lastName":"Jerry",
    "age":10,
    "streetAddress":"Disney Avenue",
    "city":"los angles",
    "state":"CA",
    "postalCode":"12345",
    "phoneNumbers":[
        {
            "number":"911",
            "type":"HOME"
        },
        {
```

Postman

- 下载
  - https://www.getpostman.com
- 文档
  - https://learning.getpostman.com/docs/postman/launching-postman/introduction/

用于接口测试

Postman



POST http://localhost:8080/user/toLogin

```
{
"name":"jjj",
"password":"123123"
}
```

Body | Cookies (1) | Headers (6) | Test Results — Status: 200 OK Time: 164ms Size: 270 B

```
{
    "msg": "登录成功",
    "status": 1,
    "id": null,
    "data": {
        "userType": 1
    }
}
```

References

- Ajax
  - http://www.w3school.com.cn/ajax/index.asp
- jQuery Ajax
  - http://www.w3school.com.cn/jquery/jquery_ajax_intro.asp
- jQuery Ajax
  - http://learn.jquery.com/ajax/
- Fastjson 常见问题
  - https://www.w3cschool.cn/fastjson/fastjson-howto.html
- No 'Access-Control-Allow-Origin' header is present on the requested resource'， 跨域访问的解决方法
  - https://blog.csdn.net/dear_little_bear/article/details/83999391
- fetch API
  - https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
  - https://www.w3cschool.cn/fetch_api/

- *Web开发技术*
- *Web Application Development*

Thank You!