**Login mit Session, Ajax für erweiterte Usability**

**Lernziele:**

* Sie können mittels eines Login Benutzerdaten aus der DB lesen und die Daten in einer Session halten.
* Sie haben sich mit Ajax beschäftigt und verstehen das Prinzip der Anwendung

# 1 Login Session

Wir haben uns bereits mit Session-Daten beschäfigt und dabei gemerkt, dass diese dann in der ganzen Web-Applikation abrufbar sind. Wir wollen uns noch einmal diesem Thema widmen und zwar im Zusammenhang mit Logins.

Erstellen Sie ein neues Projekt „Login“ und verwenden Sie die Daten & Skripts aus der login.zip Datei auf dem BSCW.

1. Lesen Sie die Beschreibung durch und führen Sie die Skripte aus.
2. Wie hängen die Dateien zusammen? Erstellen Sie ein Aktivitätendiagramm mit dem Ablauf.
3. Wir wollen auch hier möglichst MVC-mässig arbeiten. Wie könnte man das bestehende Login abändern? Machen Sie Vorschläge und zeigen Sie es der Lehrperson.

# 2 Erweiterung: Login für Artikel-Verwaltung

Verwenden Sie die Applikation von der letzten Woche. Fügen Sie der Applikation noch ein Login hinzu, so dass die DB nur mit gültigem Login verwendet werden kann.

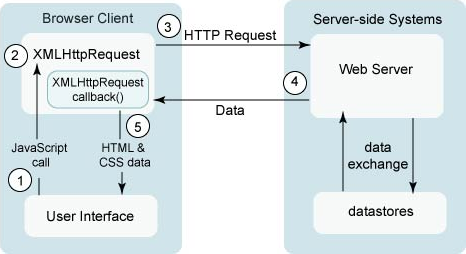
Zeigen Sie Ihre Vorschläge der Lehrperson.

# 3 Verbesserte Usability mit Ajax

Wir haben uns letzte Woche ganz kurz mit *XMLHttpRequest* beschäftigt. Nun wollen wir das etwas Näher betrachten.

Alle modernen Browser unterstützen *XMLHttpRequest* als Objekt, um innerhalb einer Webseite mit dem Server zu kommunizieren und Daten zusätzlich (in einer bereits geladenen HTML-Seite) zu laden[[1]](#footnote-1).

Schematischer Aufruf mit *XMLHttpRequest*:



Kurze Erklärungen:

1. Mittels JavaScript wird das *XMLHttpRequest* Objekt instanziiert
2. Objekt wird instanziiert, dabei wird auch der Listener *onreadystatechange* aktiviert.
3. Absetzen des HttpRequest mit *open* und *send*.
4. Server-seitig wird der Request behandelt (z.Bsp. innerhalb eines php-Skripts)
5. Daten werden an den Client zurückgeschickt und innerhalb von HTML angezeigt

Das *XMLHttpRequest* Objekt gibt Daten in unterschiedlichen Formaten zurück (also nicht nur in XML, wie der Name andeutet). Es kann auch JSON, HTML oder simplen Text zurückliefern.

Beispiel (aus w3 Quelle):

<html>

<head>

<script>

**function** showHint(str) {

**if** (str.length == 0) {

document.getElementById("txtHint").innerHTML = "";

**return**;

} **else** {

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

xmlhttp.onreadystatechange = **function**() {

**if** (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

document.getElementById("txtHint").innerHTML = xmlhttp.responseText;

}

};

xmlhttp.open("GET", "gethint.php?q=" + str, **true**);

xmlhttp.send();

}

}

</script>

</head>

<body>

<p><b>Start typing a name in the input field below:</b></p>

<form>

First name: <input type=*"text"* onkeyup="showHint(this.value)">

</form>

<p>Suggestions: <span id=*"txtHint"*></span></p>

</body>

</html>

Ablauf:

Mit dem Aufruf des Listeners *onreadystatechange* wird der Status der Verbindung zum Server geprüft. Dabei wird hier auch das Format der Rückgabe definiert (hier wäre es HTML). In der Methode *open* wird die Verbindung definiert mit der HTTP Request-Methode (POST, GET, etc.), der URL und ob es sich asynchron oder synchron verhalten soll. Der dritte Parameter is optional, jedoch empfiehlt es sich, diesen mit “true” mitzugeben. Asynchron heisst, die restliche Ausführung der Seite erfolgt ohne dass auf den *XMLHttpRequest* gewartet wird. Und da *Ajax* auch mit “Asynchronous” anfängt, macht es Sinn, den Aufruf asynchron zu benützen….

**Kleine Übung**

Verwenden Sie ein *XMLHttpRequest* Objekt, um ein Login zu prüfen oder um einen *guess* abzusetzen (wie bei einer Suche)*.* Experimentieren Sie mit Möglichkeiten.

Schauen Sie auch folgende Seite an für Beispiele, Tutorials, etc: <http://www.w3schools.com/Ajax/default.asp>

1. Falls nicht, kann mittels eines Frameworks die Kompatibilität mit älteren Browsern gewährleistet werden. [↑](#footnote-ref-1)