InstantFeed - ein Newsticker mit Websockets und Notifikationen

Einführung

In meiner Projektarbeit an der Hochschule Karlsruhe habe ich einen Newsticker programmiert, der als Referenz für einen Newsticker mit den Einsatz von *neuen* Technologien dient. Eine eingesetzte Technologie sind Websockets, mit welcher neue Nachrichten vom Server an den Brower des Benutzers geschickt werden. Damit auch keine dieser Nachrichten verpasst wird, zeigt das Betriebssystem eine Benachrichtigung über die Notifikationen API an.
Da die Nachrichten ohne Verzögerung an den Nutzer weitergeleitet werden, wird die Webapplikation *InstantFeed* genannt.

Ich habe mich dabei auf Probleme konzentriert, die mir selbst bei der Nutzung von Tickern gestört haben. Hauptsächlich verwende ich den Ticker von www.kicker.de um Fußballspiele zu verwenden. Ich habe mir zu beginn der Projektarbeit noch weitere Ticker angesehen, die alle die gleichen Probleme haben.
Neben dem Verfolgen einzeler Spiele, kann es beispielsweise bei Ligaspielen auch interessant sein, mehrere Spiele gleichzeitig zu verfolgen.

Technologien

InstantFeed ist eine Client-Server Anwendung, deren Client im Browser läuft. Der Server ist für die Persistierung und Weiterleitung der Daten von der Datenbank zum Browser zuständig. Eingesetzt wird dazu ein Express Webserver, der in Javascript geschrieben ist und in Node.js läuft. Die Daten speichert Express in der NoSQL Datenbank MongoDB. Im Browser wird AngularJs und Bootstrap eingesetzt.

Probleme von herkömmlichen Newstickern

Viele Webseiten werden heute noch als statische Webseite ausgeliefert, die nach dem initialen laden der Webseite keine weiteren Interaktionen mit dem Server vernehmen. Dabei wird man oft nicht über neue Veränderungen auf der Webseite informiert. Somit ist ein Neuladen nötig um zu prüfen, ob es Neuerungen gab. Durch das ständige neu laden einer Webseite werden neben den neuen Daten auch viele alte Daten erneut vom Server geladen. Dadurch wird die Netzauslastung unnötig erhöht.

Bei mehreren parallelen Spielen müssen die verschiedenen Feeds in unterschiedlichen Tabs geöffnet werden. Die Mögichkeit mehrere, ausgewählte Feeds in einen Feed zu Mischen ist nicht gegeben. Somit werden für jeden Tab die

gleichen Daten geladen.
Durch mehrere Feeds in verschiedenen Tabs sind gegebenfalls Tabs im Hintergrund und Änderung in diesen werden verpasst. Dies kann durch ein Benachrichtigung auf Betriebssystemebene gelöst werden.

Lösungen

Die Lösungen für die Probleme aus dem letzten Abschnitt werden im Folgenden Abschnitt erläutert. Dabei wird immmer auf Lösungen aus dem Blickwinkel der Webapplikation InstantFeed. Zuerst werden Websockets erklärt. Dann werden die Desktopnotifikationen erklärt. Zum Abschluss gehe ich auf die personalisierte Feeds eingehen.

Websockets

Wird eine Webseite im Browser aufgerufen, fragt der Browser alle Resourcen bei dem entsprechenden Server ab. Wurden alle Anfragen beantwortet, kann der Server keine weiteren Daten an den Browser schicken. Aller Datenverkehr zwischen Browser und Server muss vom Browser initialisiert werden. Um neue Daten vom Server zu bekommen, nachdem die Webseite geladen wurde, gibt es zwei Möglichkeiten.
Eine Möglichkeit besteht darin, in periodischen Zeitabständen Anfragen an den Server zu schicken. Wenn neue Daten vorhanden sind werden diese geladen, jedoch werden auch die alten Daten erneut geladenn. Dadurch und durch die die vielen Abfragen entsteht eine unnötige Last auf dem Server und dem Netzwerk.
Die andere Möglichkeit ist das sogenannte *long polling*. Bei long polling wird vom Browser eine Anfrage an den Server gestellt, welche nicht sofort beantwortet wird. Die Verbindung wird so lange offen gehalten, bis entweder neue Daten beim Server zur Verfügung stehen oder die maximale Wartezeit überschriten wird. Auch hierbei wird die Last auf dem Server, durch viele offene Anfragen, erhöht.
Zusätzlich nutzen beide Methoden HTTP(S) und damit desen Overhead. Websckets setzen direkt auf TCP, welches auch ein Unterbau von HTTP(S) ist.

Auch bei Websockets wird die Verbindung vom Client initialisiert, jedoch nicht nach dem Datentransfer abgebaut. Über diese bidirektionale Verbindung kann in beide Richtungen, Client zu Server und Server zu Client, Daten geschickt werden. Dadurch das auf HTTP(S) verzichtet wird und direkt auf TCP aufgesetzt wird, ist der Overhead deutlich kleiner, wie bei den zuvor vorgestellten Methoden.

Bei InstantFeed wird eine Websocketverbindung zum Server aufgebaut, wenn der Feed angezeigt wird. Wird eine neue Nachricht von einem Client an den Server geschickt, wird nach dem Speichern die Nachricht über den Websocket an alle Clients verteilt. Die Variable `doc` enthält dabei die gespeicherte Nachricht als Javascriptobjekt.

```
exports.register = function(socket) {
  Message.schema.post('save', function (doc) {
    onSave(socket, doc);
  });
}
function onSave(socket, doc, cb) {
  socket.emit('message:save', doc);
}
```

Auf dem Client wurde eine Funktion vorbereitet, die Nachrichten über Websockets verarbeitet. Diese Funktion hat einen Identifier für die Art der Daten, das Array, welchem die Nachricht hinzugefügt wird, sowie eine Callbackfunktion übergeben bekommen.
Der Identifier ist nötig, da nicht nur Nachrichten, sondern alle Daten werden über den gleichen Websocket syncronisiert werden. In dieser Methode sollen aber nur Nachrichten verarbeiten werden, deshalb müssen diese anhand des Idenifiers hergeausfiltert werden.
Das Array ist im Beispielcode `array` genannt. Zuerst wird überprüft, ob sich darin bereits ein Objekt mit der gleichen `id` befindet. Das Objekt mit dieser `id` wird dann ersetzt oder wenn das Objekt noch nicht vorhanden ist, das neue Objekt hinzugefügt.
Am Ende wird die übergebene Callbackfunktion ausgeführt. In InstantFeed wird in der Callbackfunktion wird die Topic für die Nachricht gesetzt, geprüft, ob die Topic weggelassen werden kann und gegebenfalls eine Benachrichtigungen geschickt.

```
socket.on(modelName + ':save', function (item) {
  var oldItem = _.find(array, {_id: item._id});
  var index = array.indexOf(oldItem);
  var event = 'created';

  // replace oldItem if it exists
  // otherwise just add item to the collection
  if (oldItem) {
    array.splice(index, 1, item);
    event = 'updated';
  } else {
    array.push(item);
  }
  cb(event, item, array);
});
```

In InstantFeed enthält das Objekt, welches über den Websocket empfangen wird, nur die Nachricht selbst. Die Nachricht selbst hat eine Referenz auf die zugehörige Topic und gegebenfalls auf ein Bild. Die Topic und das Bild werden wiederum per HTTP(S) Anfrage vom Server geladen.

Notifications

Desktopnotifikationen sind Benachrichtigungen, welche vom Browser initialisiert und vom Betriebssystem angezeigt werden. Dadurch muss der Browser nicht im Vordergrund laufen, um den Benutzer zubenachrichtigen. Die Webapplikation muss trozdem im Browser laufen. Wenn zum ersten Mal eine Benachrichtigung des Browsers geschickt werden soll, wird der Nutzer gefragt, ob er diese zulassen will.
Das erste nachfolgende Bild zeigt eine Benachrichtigung unter Kubuntu, das zweite unter OSX.

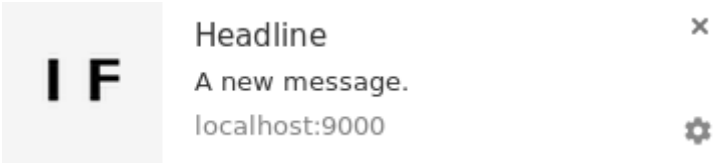

Figure 1: Notification Linux

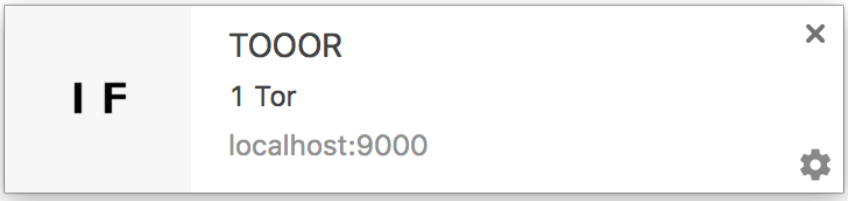

Figure 2: Notification OSX

Eine Notification zeigt das Icon der Webapplikation und deren Url an. Außerdem die Überschrift und der Anfang des Textes der neuen Nachricht. Nach 10 Sekunden verschwindet sie automatisch wieder.

Für die Notifications wird die Bibliothek *angular-web-notification* verwendet. Damit muss die Funktion `showNotification` auf dem Objekt `webNotification` aufgerufen werden. Der Funktion wird als erster Übergabeparameter die Überschrift der Benachrichtigung übergeben.
Der zweite Parameter ist eine Objekt mit Einstellungen für die Benachrichtigung. Das Attribut `body` enthält den Text, der in der Benachrichtigung angezeigt wird. 'icon' ist das Icon, welches am linken Rand angezeigt wird. Die Funktion `onClick` wird ausgeführt, wenn die Benachrichtigung angeklickt wird. In InstantFeed wird dabei der Tab, welcher die Benachrichtigung geschickt hat, in den Vordergrund geholt. In `autoClose` wird angegeben, nach wie vielen Millisekunden die Benachrichtigung wieder geschlossen wird.

Der dritte Parameter ist eine Funktion die ausgeführt wird, wenn die Benachrichtigung angezeigt wird. Darin findet die Fehlerbehandlung statt.

```
function notify(message) {
  webNotification.showNotification(message.headline, {
    body: message.text,
    icon: 'favicon.ico',
    onClick: function onNotificationClicked() {
      $window.location.hash = message._id;
      $window.focus();
    },
    autoClose: 10000
  }, function onShow(error) {
    if (error) {
      alert('Unable to show notification: ' + error.message);
    }
  });
}
```

Unterstützt werden Benachrichtigung von Chrome und Firefox. Jedoch von keinem mobilen Browser.

Benutzerdefinierte Feeds

Ein weiteres Feature sind benutzerdefinierte Feeds. Dabei kann sich der Benutzer aus den aktuellen Topics, die aussuchen von denen er Nachrichten in seinem Feed angezeigt bekommen will.
Diese Einstellungen werden im Local Storage des Browsers gespeichert. Damit ist die Auswahl für einen Computer gespeichert, auch wenn der Browser geschlossen und der Computer ausgeschaltet wird. Die Einstellungen werden aber nicht über den Computer hinaus syncronisiert.

Beim Aufrufen der Webseite und hinzufügen einer Topic wird eine Anfrage an den Server gestellt, der nur die Nachrichten zu den ausgewählten Topics läd. Neue Nachrichten werden nicht auf dem Server gefiltert, da sonst der Server sich zu jedem Websocket die ausgewählten Topics speichern müsste. Stattdessen werden alle Nachrichten über Websockets an die Clients verteilt und dort gefiltert.

Fazit

Mit Websockets konnten sowohl unnötige Anfragen an den Server verhindert werden, als auch die Netzauslastung verringert werden. Dank der Notifications API werden keine Ereignisse mehr verpasst. Die Technologien können dank entsprechender Bibliotheken, welche eine Abstraktion der Funktionalität zur Verfügung stellen, von jedem Entwickler eingesetzt werden.

Anhang

https://github.com/KordonDev/InstantFeed
https://nodejs.org/
http://expressjs.com
http://mongoosejs.com
https://angularjs.org/ https://github.com/sagiegurari/angular-web-notification