Micamu - Serverseitiges und clientseitiges Javascript

Lukas Körfer

Aachen, am 31. Juli 2016

Die im Juli 2016 von IT-Talents veranstaltete *Code Competition* sieht den Entwurf und die Implementierung einer Weboberfläche zur Überwachung von angebundenen System vor. In mehreren Programmen, die Systeme des Typs *MICA* von Harting imitieren, sollen ferngesteuert Software-Container, die virtuellen Maschinen entsprechen, hinzugefügt, entfernt, gestartet und gestoppt werden. IT-Talents und Harting bieten gemeinsam mit der Aufgabenstellung ein Programm an, das die erwähnten *MICA*-Systeme simuliert. Dieses kann über eine *JSON-RPC* Schnittstelle angesteuert werden und läuft unter *Node.js*, also serverseitigem Javascript. Mir bietet diese *Code Competition* nicht nur eine weitere interessante Aufgabe, sondern soll auch als Einstieg in das Thema *Node.js* dienen und zeitgleich meine Kenntnisse in Javascript und Typescript (typisiertes Javascript) auffrischen und ausbauen. Dem Ziel von *Node.js*, die Webentwicklung durch die Nutzung einer einzigen Sprache nicht nur im Browser, sondern auch auf dem Server, zu vereinfachen, folgend, wird neben den anzusteuernden virtuellen *MICAs* auch der Server meines Projektes Micamu (MICA Management Unit) auf *Node.js* aufbauen.

Inhaltsverzeichnis

1 Vorbereitungen

Das mit der Aufgabenstellung ausgelieferte Programm *virtuellemaschine.js* wurde geringfügig verändert, in erster Linie um das Starten mehrerer virtueller Maschinen, die auf unterschiedlichen Ports agieren, zu ermöglichen. Der gewünschte Port kann nun als Kommandozeilenargument beim Start der *MICA*-Imitation angegeben werden.

Um *JSON-RPC*-Anfragen zu verwenden, werden im Micamu Projekt verschiedene Drittanbieter-Bibliotheken verwendet. Zwei dieser Bibliotheken wurden zur besseren Integration geringfügig modifiziert und können daher nicht mehr unmittelbar aus ihrer ursprünglichen Quelle bezogen werden:

Die erste Bibliothek, die unter dem Namen **node-json-rpc** im Node Package Manager (NPM) zu finden ist, wurde um eine Möglichkeit zur Fehlerbehandlung bei Verbindungsfehlern erweitert.

Die zweite Bibliothek, **jquery-jsonrpc**, die sich auf GitHub befindet[1], wurde für eine vereinfachte Verwendung um eine Überprüfung der *JSON-RPC*-Version gekürzt.

[1] https://github.com/datagraph/jquery-jsonrpc

2 Programmaufbau

Micamu ist wie beinahe jede Webanwendung in zwei Teile aufgeteilt, den Server und den Client. Als Besonderheit kommuniziert dieses System mit weiteren Einheiten, die Geräte im Netzwerk des Systems nachbilden. Die Kommunikation sowohl mit diesen Einheiten als auch untereinander erfolgt in erster Linie über *JSON-RPC*-Anfragen, nur beim Start einer Sitzung werden die Client-Dateien (HTML, CSS und Javascript) über gewöhnliche HTTP-Anfragen vom Server an den anfragenden Client übermittelt. Sämtliche weitergehende Kommunikation geht vom Client-Programm aus.

Trotz der Server/Client-Architektur bestehen die Daten des Micamu Projektes ausschließlich im Client. Da sämtliche Informationen über die angebundenen Maschinen von diesen autonom verwaltet werden, können sie immer wieder abgefragt werden, ein verwaltender Server wird somit überflüssig. Um Aufrufe, die gegen die Same-Origin-Policy (SOP) verstoßen, zu verhindern, werden sämtliche Anfragen des Clients an die einzelnen Maschinen über den Server geroutet, indem die ursprüngliche *JSON-RPC*-Anfrage vom Server angenommen und in eine Anfrage an die gewünschte Maschine umgebaut wird. Die jeweilige Antwort wird dann zurück an den Client gesendet, dieser nutzt die enthaltenen Daten zur Darstellung.

Der durch diese Aufteilung ausgesprochen schlanke Server agiert somit ausschließlich zur Verteilung des Client-Programmes an den Browser sowie zur Weiterleitung der Anfragen an die virtuellen Maschinen. Über einen zusätzlichen *JSON-RPC*-Aufruf kann zudem eine Liste von Maschinen-Adressen sitzungsübergreifend gespeichert werden. Beim Start der nächsten Sitzung werden die gespeicherten Adressen über eine mitgelieferte Javascript-Datei wieder an den Client übergeben.

3 Grafische Oberfläche

Die grafische Oberfläche, bestehend aus einer HTML-Seite, die vom Micamu-Client angesteuert wird, setzt als Motor das *AngularJS*-Framework ein. AngularJS bietet in der Webentwicklung die Möglichkeit, eine Trennung von Datenlogik und Datendarstellung im Sinne von MVC oder MVVM vorzunehmen. Die einzelnen HTML-Elemente werden dazu über spezielle Attribute (**ng-...**) an ein in Javascript definiertes Modell gebunden. Ändert sich dieses Modell, ändert sich automatisch die Darstellung in HTML und werden Steuerelement der Ansicht verwendet, werden wiederum Methoden des Modells aufgerufen, die dann wieder automatische Änderungen in der Oberfläche hervorrufen können.

Im Fall von Micamu besteht dieses Modell aus einem *Session*-Objekt, das die komplette Micamu-Sitzung verwaltet. Dieses enthält mehrere *Device*-Objekte, welche unterschiedlich viele *Container*-Objekte beinhalten.

Die Nutzeroberfläche setzt dies folgendermaßen um: Auf der linken Seite der Ansicht werden die einzelnen Geräte (*Device*) angezeigt und mittig rechts bei Auswahl eines Gerätes die enthaltenen Container (*Container*). Es können über sogenannte Aktionsschaltflächen unter anderem jeweils Geräte oder Container hinzugefügt werden. Beim Hovern über einzelne Elemente bieten weitere Schaltflächen zusätzliche Bedienmöglichkeiten.

Für eine angenehmere Nutzererfahrung wird für die Gestaltung der Schaltflächen das bekannte *Bootstrap*-Framework verwendet. Die verwendeten Icons stammen aus dem freien *Font Awesome*-Projekt.

4 Ausblick

Obwohl bereits sämtliche vorgegebenen sowie einige freiwillige Features implementiert wurden, besitzt Micamu nach nur einem Monat Entwicklungszeit noch ein großes Verbesserungspotential. Die sitzungsübergreifende Persistenz der Gerätedaten könnte so beispielsweise über eine Datenbank bereitgestellt werden, die dann auch die Speicherung und Nutzung unterschiedlichen Konfigurationen und Zusammenstellungen auf einer Server-Instanz erlaubt.

Auch im Bereich der bereits recht ausgefeilten grafischen Oberfläche werden Schwächen sichtbar, sobald man sich auf die Betrachtung von unterschiedlichen Endgeräten und Bildschirmgrößen (Stichwort *responsive*) fokussiert.