REST Hooks
(Why) do we need them?

F. Viola[1]

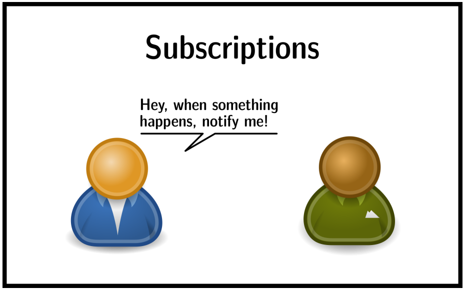
[1]ARCES
University of Bologna

Mar. 2017

Outline

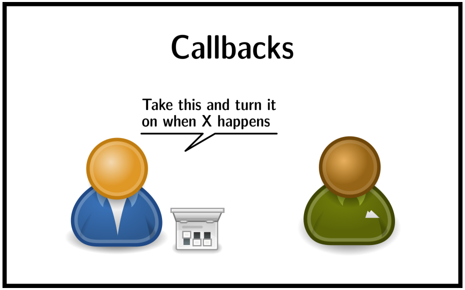
Outline

Subscriptions



Example: when a person registers to the mailing list send me an email.

Callbacks



Example: building graphical applications we bind functions to buttons. When the button is pressed, the function is executed.

Outline

Outline

Web Hooks

A WebHook is simply an HTTP callback!

A web application implementing WebHooks will POST a message to a URL when certain things happen. [CUT] For the user, WebHooks are **a way to receive valuable information when it happens**, rather than continually polling for that data and receiving nothing valuable most of the time.

Ref: https://webhooks.pbworks.com/w/page/13385124/FrontPage

Web Hooks

Web Hooks
Hey! When something happens, do this HTTP POST to XYZ

Example: when a new post is published on the blog, perform an HTTP POST to my server.

Is someone using Web Hooks?

Among others, these are some of the users of Web Hooks:

- BitBucket
- Facebook App Platform
- **GitHub**
- Google Code
- PayPal
- Shopify
- ZenDesk
- . . .

Ref: https://webhooks.pbworks.com/w/page/13385124/FrontPage

Outline

WebHooks and GitHub

When configuring a webhook, you can choose which events you would like to receive payloads for.

Examples of events on GitHub:

- `*` – every time something changes;
- `commit_comment` – every time a commit receives a comment;
- `fork` – when a repository is forked;
- `repository` – every time a repository is created, deleted, modified…
- `team` – every time a team is created, deleted, modified…
- ...

Each event type has a specific payload format with the relevant event information.

Ref: https://developer.github.com/webhooks/

WebHooks on GitHub

In order to experiment with the WebHooks functionalities provided by GitHub, we need a Web server where notifications will be posted.

I wrote my own server using **Ruby** and its framework **Sinatra**. The project is made up by a ruby file (`github-notif.rb`) and an erb file (simply an HTML template including ruby code).

`github-notif.rb` only implements two HTTP methods:

- `POST /subscriptions`: used by GitHub to send notifications;
- `GET /subscriptions`: used by us to read the notifications.

The server listens on: `http://wot.arces.unibo.it:4567`

WebHooks on GitHub

Let's try WebHooks!

1. First of all: let's start my Web Server! :-)
2. Go to the **settings page** of our GitHub project (in my case I will use this one)
3. Click on **WebHooks**, then **Add WebHook**
4. Now, wait for notifications!

Outline

Outline

REST Hooks

REST Hooks itself is not a specification, it is a collection of patterns that treat webhooks like subscriptions. These subscriptions are manipulated via a REST API just like any other resource.

Ref: `http://resthooks.org/docs/`

REST Hooks

The resource dedicated to subscriptions is then accessed through:

- `GET /subscriptions` – to access the list of all the subscriptions
- `GET /subscriptions/<id>` – to access a single subscription
- `POST /subscriptions` – to create a subscription
- `PUT /subscriptions/<id>` – to modify a subscription
- `DELETE /subscriptions` – to delete all the subscriptions
- `DELETE /subscriptions/<id>` – to delete a subscription

REST Hooks

The resource dedicated to subscriptions is then accessed through:

- `GET /subscriptions` – to access the list of all the subscriptions
- `GET /subscriptions/<id>` – to access a single subscription
- `POST /subscriptions` – to create a subscription
- `PUT /subscriptions/<id>` – to modify a subscription
- `DELETE /subscriptions` – to delete all the subscriptions
- `DELETE /subscriptions/<id>` – to delete a subscription

. . . REST. No surprises.

Outline

Hands on code!

Here again with another **Hands on code** session. In this one we will build a web server using **Javascript** and its framework **Node JS**.

Just to get familiar with Node JS: is an asynchronous event-driven JS runtime environment built on top of the ChromeV8 engine and designed for scalable network applications.

Structure of a NodeJS application

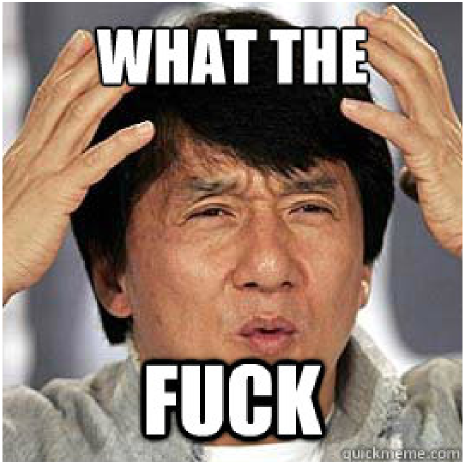
A NodeJS application is composed by a folder containing a `package.json`, the main JS file and the JS libraries.

```
\
|_index.js
|_package.json
|_node_modules
   |_/modules/
```

In our application `index.js` relies on

Outline

Wot: the FAQ



Web Hooks and the WoT

How do Web Hooks and WoT relate?

WoT needs a subscription mechanism and since WoT is based on **servients**, implementing WebHooks is possible. Guinard and Trifa [1] state that the simplest way to implement a publish-subscribe system over HTTP without breaking the REST model is to treat every entity as both a client and a server. To implement a subscription mechanism based on Web Hooks, we just need to implement a REST API on both the Web Thing and its client (that then becomes a server as well).

[1] in their book "Building the Web of Things"

Web Hooks and the WoT

How do Web Hooks and WoT relate?

WoT needs a subscription mechanism and since WoT is based on **servients**, implementing WebHooks is possible. Guinard and Trifa [1] state that the simplest way to implement a publish-subscribe system over HTTP without breaking the REST model is to treat every entity as both a client and a server. To implement a subscription mechanism based on Web Hooks, we just need to implement a REST API on both the Web Thing and its client (that then becomes a server as well).

[1] in their book "Building the Web of Things"

Web Hooks and the WoT

Web Hooks look a bit demanding. Aren't they?

Yes. Every client must also be a server. It is not so trivial. Clients should have a public IP address or should have an available third party HTTP server. . . This problem is also mentioned in "Architecting the Internet of Things" by Uckelmann et al.

Web Hooks and the WoT

Are Web Hooks the definitive choice for subscription in the WoT?

From the WebThing model: An Extended Web Thing **SHOULD** support subscriptions via the WebSocket [RFC6455] protocol and **MAY** support subscriptions via WebHooks (HTTP callbacks). Subscriptions are enabled via the HTTP protocol upgrade mechanism [RFC2616] as shown below.

Currently the Thing Description refers ti an *Event* resource for the HTTP interface that is aimed at handling subscriptions. The same is valid for the CoAP binding. . . So are they referring to web hooks?

Conclusions

The presentation is over. Now it's time for discussion.

Thank you for the attention.