creamlab/revcor

Go online rewrite of Python offline project

Guillaume Denis Feb. 24, 2022
Café Neuro @ FEMTO-ST

Goals

- Easy to install and to extend
  - No database, rely on configuration files and produce CSV output
    Create a new experiment: https://github.com/creamlab/revcor/tree/main/data
  - Go language:
    - HTTP server with WebSocket upgrade
    - Esbuild for JS assets
    - Compile application and copy to server
    - Language advertised as “simple”
- Use existing UX front-end library
  - jspsych, see https://www.jspsych.org/
  - JavaScript framework for creating behavioral experiments
  - Timeline, (composable) plugins, stimuli

Demo

Made With

- Server
  - Go language
  - Go standard library: net/http, encoding/json, encoding/csv, html/template, math/rand, os (files)...
  - https://github.com/gorilla/mux for HTTP routing
  - https://github.com/gorilla/websocket to manage WebSockets
  - https://github.com/evanw/esbuild to process JS assets
    → webpack/rollup.js, yarn/npm, babel.js… not needed!
- Front
  - JavaScript language (latest version thanks to esbuild)
  - jspsych plugins used:
    - preload
    - survey-html-form
    - html-keyboard-response
    - audio-keyboard-response
- Flexible setup for other experiments

Pages

- Run experiment with declared participant
  - https://neuro-xp.femto-st.fr/revcor/xp/example/run/102
  - If 102 is declared in the participants.txt file
  - Page refresh is allowed
  - In Go source: publicRouter, websocketHandler, soundHandler
- Run experiment with new participant
  - https://neuro-xp.femto-st.fr/revcor/xp/example/new
  - if `allowCreate: true` in settings.json
  - In Go source: publicRouter, newHandler, createHandler (then same as above)
- Results
  - http://neuro-xp.femto-st.fr/revcor/xp/example/results
  - Protected by HTTP auth basic, see password in settings.json
  - In Go source: resultsRouter, resultsAuthMiddleware

Server Setup

- Preparation
  - Direct exposure of revcor or possible nginx/Apache front that proxies to revcor
    → set environment variables accordingly https://github.com/creamlab/revcor#environment-variables
  - Run in background or manage with tools like supervisor, pm2, docker…
- Deployment
  - Compile, copy revcor binary to server:

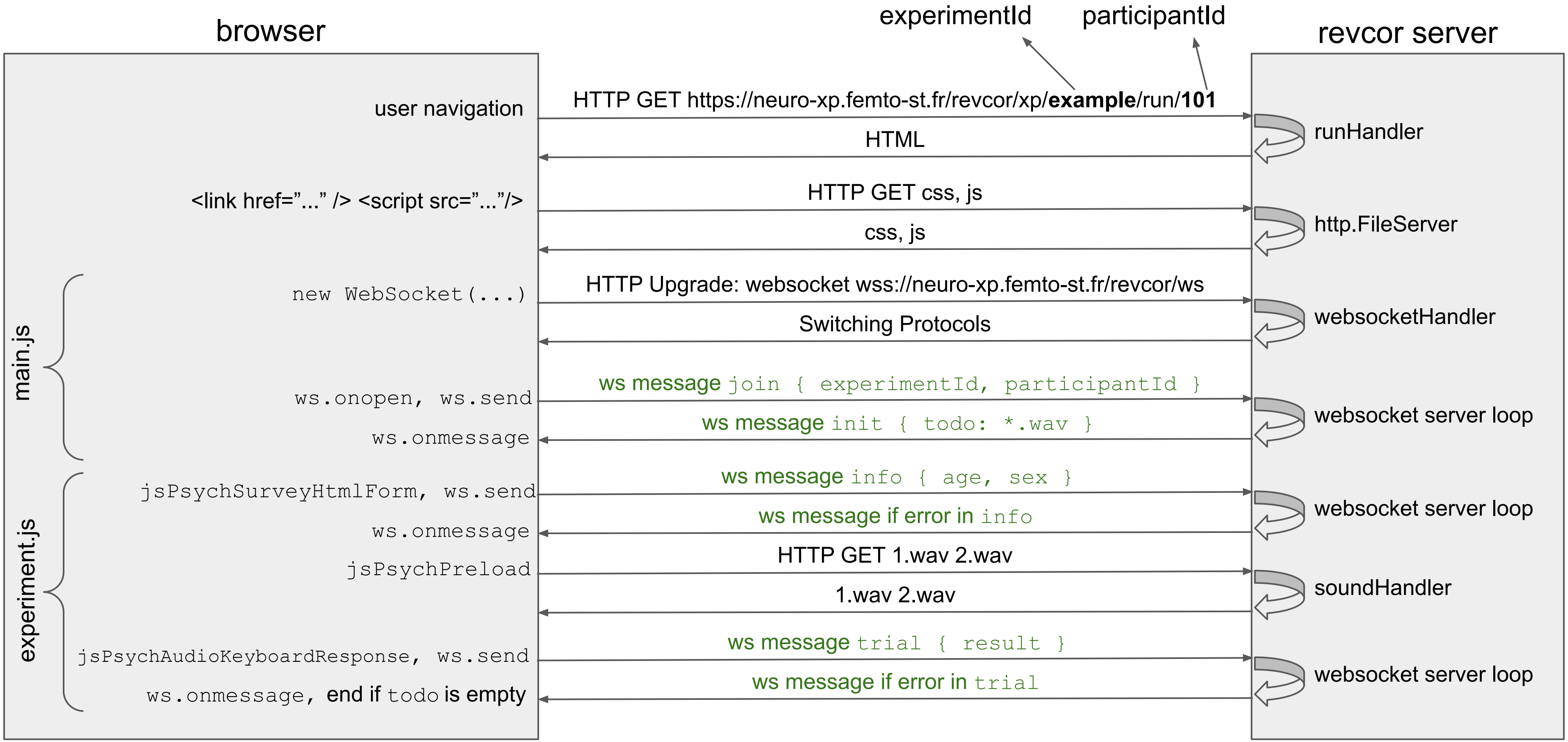
    ```
    dev@laptop$ go build
    dev@laptop$ rsync -a revcor user@server:~/deploy-revcor
    ```
  - Restart:

    ```
    deploy@server$ supervisorctl restart revcor
    ```

Code Walkthrough

Source https://github.com/creamlab/revcor

Page Load and Interaction



browser

experimentId

participantId

revcor server

user navigation

HTTP GET https://neuro-xp.femto-st.fr/revcor/xp/**example**/run/**101**

HTML

runHandler

<link href="..." /> <script src="..."/>

HTTP GET css, js

css, js

http.FileServer

main.js

`new WebSocket(...)`

HTTP Upgrade: websocket wss://neuro-xp.femto-st.fr/revcor/ws

Switching Protocols

websocketHandler

`ws.onopen, ws.send`

ws message `join { experimentId, participantId }`

`ws.onmessage`

ws message `init { todo: *.wav }`

websocket server loop

experiment.js

`jsPsychSurveyHtmlForm, ws.send`

ws message `info { age, sex }`

`ws.onmessage`

ws message if error in `info`

websocket server loop

`jsPsychPreload`

HTTP GET 1.wav 2.wav

1.wav 2.wav

soundHandler

`jsPsychAudioKeyboardResponse, ws.send`

ws message `trial { result }`

`ws.onmessage`, end if `todo` is empty

ws message if error in `trial`

websocket server loop

In green: bidirectional messages over WebSocket connexion