Creating a New Experiment With Pushkin

The Back End

The back end of a quiz is made up of 5 components: an API controller, models, migrations, seeds, and a python worker. Migrations are created with **Knex.js** and models are created with **Bookshelf.js**. The migrations should match the database schema for the quiz. The models handle the transfer of data into and out of the database. By using **Bookshelf.js** we ensure that when stimuli are retrieved from the database they arrive at the python worker in the format of a regular object. This way they can be modified, scrambled, or sorted before they are fed to the front-end. The seeds handle inserting all the stimuli for a quiz into the database. The stimuli are saved in a **csv** file that follows the format of the database table they need to be inserted in. The seeder script ensures that the stimuli from the **csv** file populate the stimuli database table at the time when the quiz is deployed.

The Front End

The front end of a quiz consists of an experiment file (index.js), the jsPsych plugins the quiz uses, a css file with the styles for the quiz, and an Axios file used by the front-end to make calls to the API from the browser.

Creating a New Quiz

The Back End

IMPORTANT NOTE: Do **NOT** use uppercase letters in the name of the new quiz. This will cause problems when the Docker containers are rebuilt after the quiz has been added.

You can use the **cli tool** to scaffold the back end items for a new quiz. Alternatively, you can modify the files for an existing quiz and create brand new files. To generate the template files for a new quiz run:

```
$ pushkin generate dbItems QUIZ_NAME
```

OR

```
$ pushkin scaffold QUIZ_NAME
```

All the scaffolded files for a quiz can be found in a folder with the quiz name located in the **experiments** folder. After those files have been modified to reflect the requirements of the new quiz, run the following commands:

```
$ pushkin sync
```

After syncing, open the **docker-compose.debug.yml** file and make sure that the new quiz has been added to the end of the file. Pay attention to the **image**. Make sure it follows the same format as previous quizzes. When all the required changes have been made, run:

```
$ docker-compose -f docker-compose.debug.yml up --build
```

to rebuild the Docker containers, followed by:

```
$ cd db-worker
$ docker ps
```

Select the Container ID of db-worker and run:

```
$ docker exec -it CONTAINER_ID bash
$ npm run migrations
$ node seeder.js NAME_OF_QUIZ
```

A series of questions will appear; the answer to all of them should be **Yes**.

The Front End

To create the front end for the quiz,

In the `/front-end/experiments` folder, add a folder named after the quiz, and put index.js, the jsPsych plugins the quiz uses, the css file with the styles for the quiz, and an Axios file inside this folder. Additionally, you will need to add the link to the quiz in `front-end/pages/quizzes/index.js`. The last step is adding your new quiz to `/front-end/core/routes.js'

Updating an existing Pushkin project with a new quiz

If you want to publish a new quiz on your existing website, follow these steps:

1. Add the back end files in the **main_project_folder->experiments** folder.
2. Add the front end files in the **main_project_folder -> front-end->experiments** folder.
3. Add the route in **routes.js** and add the quiz on the **Quizzes** page (front-end/pages/quizzes/index.js)
4. Add your quiz in **docker-compose.debug.yml**, **docker-compose.production.yml** and **rancher-compose.yml**.
5. Run the publish, copy, tag and push script:

```
env NODE_ENV=production npm run publish &&
cd .. &&
cp -rf ./front-end/public/**.ico ./server/html &&
cp -rf ./front-end/public/**.txt ./server/html &&
cp -rf ./front-end/public/**.html ./server/html &&
cp -rf ./front-end/public/**.xml ./server/html &&
docker build -t DOCKERHUB_ID/api:latest api &&
docker build -t DOCKERHUB_ID/cron:latest cron &&
docker build -t DOCKERHUB_ID/db-worker:latest db-worker &&
docker build -t DOCKERHUB_ID/server:latest server &&
docker build -t DOCKERHUB_ID/NEW_QUIZ:latest workers/NEW_QUIZ &&
docker push DOCKERHUB_ID/api &&
docker push DOCKERHUB_ID/cron &&
docker push DOCKERHUB_ID/db-worker &&
docker push DOCKERHUB_ID/server &&
docker push DOCKERHUB_ID/NEW_QUIZ
```

1. In Rancher, add a new service in your existing stack.
2. Upgrade all existing Rancher services (make sure you pull the new images before upgrading).