Progress MarkLogic

Unit 5

Learn More

Grove UI

Exercise 1: Grove UI

This exercise focuses on getting you up and running with a single page web app (SPA) using MarkLogic as the back-end database, Express as the middle-tier and ReactJS as the front-end.

1. Open a Terminal window (Applications→System Tools→Mate Terminal).
2. Install the command line interface for ml-grove.

```
$ sudo npm install -g @marklogic-community/grove-cli
```

   - This makes the `grove` command available.

3. Navigate to your desktop:

```
$ cd /home/cent/Desktop/
```

4. Bootstrap your first grove project.

```
$ grove new myApp
```

   - Answer the questions as prompted by the script.

5. Go to the project folder.

```
$ cd myApp
```

   - You should see something similar to the following folder structure.

```
myApp
├──docs
├──marklogic
│   ├──gradle
│   ├──mlcp
│   └──src
│       └──main
│           ├──ml-config
│           ├──ui-data
│           ├──ui-modules
│           └──ui-schemas
├──middle-tier
│   ├──grove-default-routes
│   ├──grove-legacy-routes
│   ├──grove-node-server
│   ├──grove-node-server-utils
│   └──routes
│       └──api
└──ui
```

   - Apart from the naming prefix (`ml-` vs `ui-`), notice that `marklogic` folder contains your typical ml-gradle project.
   - `middle-tier` contains your application server files.
   - `ui` contains your web application implemented in either VueJS or ReactJS depending on your earlier selection.

6. Synchronize the ports to be used by running the following command:

```
$ grove config
```

- Use the following information to answer the prompts:

Prompt	Value
What host…	localhost
What port…REST Server	8400
What port…Node…	9003

- Above script changes the corresponding files for `marklogic`, `middle-tier` and `ui`:

```
$ grep -r 8400
```

- Output:

```
[cent@centos7-vm myApp]$ grep -r 8400
marklogic/gradle.properties:mlRestPort=8400
middle-tier/.env:GROVE_ML_REST_PORT=8400
```

```
$ grep -r 9003
```

- Output:

```
README.markdown:Note that this will run on `http://localhost:9003` by default (rather than port 3000, where the development Webpa
ck server runs by default).
docker-compose.yml:      - "9003:9003/tcp"
middle-tier/Dockerfile:EXPOSE 9003
middle-tier/grove-node-server-utils/options.js:    default: 9003,
middle-tier/.env:GROVE_APP_PORT=9003
ui/README.markdown: * The middle tier is listening on port 9003
ui/nginx.conf:        server grove-node:9003;
ui/package.json:  "proxy": "http://localhost:9003",
```

7. Go to the `marklogic` folder

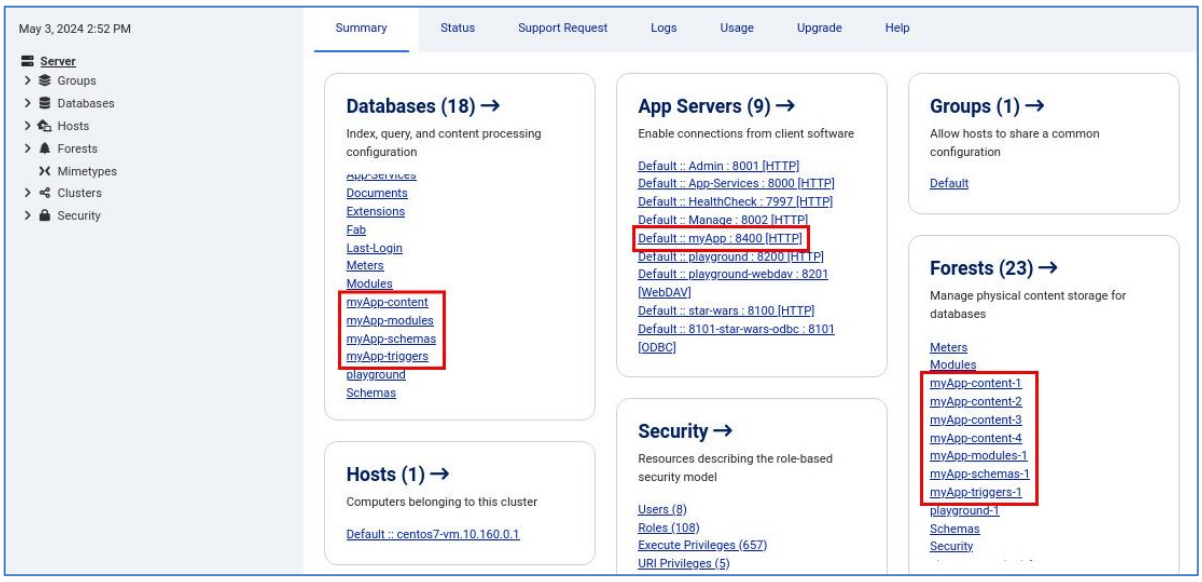
```
$ cd marklogic
```

8. Deploy this application to marklogic and deploy some sample data.

```
$ ./gradlew mlDeploy loadSampleData
```

- Open the `build.gradle` and see that the second task is a custom MLCP task.
- When `gradlew` is present, use it make sure you are running the same gradle as specified by the application author.

9. Open the Admin UI: http://localhost:8001.
    - Notice that you now have additional Databases, App Servers, Forests, User and Role created for myApp.



10. Open Query Console: http://localhost:8000/qconsole
11. Select the myApp-content database and click on “Explore”
    - Notice that you now have data available.
12. Back at your terminal window, go back to the “myApp” folder of your project

```
$ cd ..
```

13. Install the dependencies of your middle-tier and ui

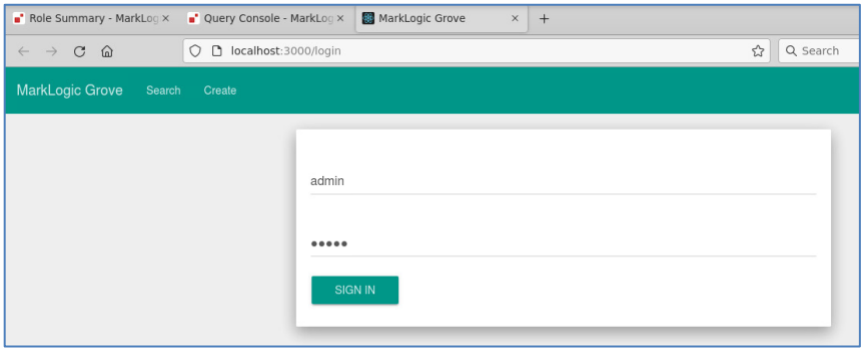
```
$ npm install
```

14. Run your node application server for local use.

```
$ npm start
```

    - Your browser will automatically load http://localhost:3000 as soon as your components are ready.

15. Log in as admin / admin.



16. Click the “Search” button and explore the app as desired.