kaartje

This project can be used as a starting point to create your own Vaadin application with Spring Boot. It contains all the necessary configuration and some placeholder files to get you started.

Running the application

The project is a standard Maven project. To run it from the command line, type `mvnw` (Windows), or `./mvnw` (Mac & Linux), then open http://localhost:8080 in your browser.

You can also import the project to your IDE of choice as you would with any Maven project. Read more on how to import Vaadin projects to different IDEs (Eclipse, IntelliJ IDEA, NetBeans, and VS Code).

Deploying to Production

To create a production build, call `mvnw clean package -Pproduction` (Windows), or `./mvnw clean package -Pproduction` (Mac & Linux). This will build a JAR file with all the dependencies and front-end resources, ready to be deployed. The file can be found in the `target` folder after the build completes.

Once the JAR file is built, you can run it using `java -jar target/kaartje-1.0-SNAPSHOT.jar`

Project structure

- `MainLayout.java` in `src/main/java` contains the navigation setup (i.e., the side/top bar and the main menu). This setup uses App Layout.
- `views` package in `src/main/java` contains the server-side Java views of your application.
- `views` folder in `frontend/` contains the client-side JavaScript views of your application.
- `themes` folder in `frontend/` contains the custom CSS styles.

Useful links

- Read the documentation at vaadin.com/docs.
- Follow the tutorials at vaadin.com/tutorials.
- Watch training videos and get certified at vaadin.com/learn/training.
- Create new projects at start.vaadin.com.
- Search UI components and their usage examples at vaadin.com/components.
- View use case applications that demonstrate Vaadin capabilities at vaadin.com/examples-and-demos.
- Discover Vaadin's set of CSS utility classes that enable building any UI without custom CSS in the docs.
- Find a collection of solutions to common use cases in Vaadin Cookbook.
- Find Add-ons at vaadin.com/directory.
- Ask questions on Stack Overflow or join our Discord channel.
- Report issues, create pull requests in GitHub.

Deploying using Docker

To build the Dockerized version of the project, run

```
docker build -t your_dockerhub_username/kaartje:latest .
```

Once the Docker image is correctly built, you can test it locally using

```
docker run -p 8080:8080 -d your_dockerhub_username/kaartje:latest
```

Deploying using Kubernetes

We assume here that you have the Kubernetes cluster from Docker Desktop running (can be enabled in the settings).

First build the Docker image for your application. You then need to make the Docker image available to you cluster. With Docker Desktop Kubernetes, this happens automatically. With Minikube, you can run `eval $(minikube docker-env)` and then build the image to make it available. For other clusters, you need to publish to a Docker repository or check the documentation for the cluster.

The included `kubernetes.yaml` sets up a deployment with 2 pods (server instances) and a load balancer service. You can deploy the application on a Kubernetes cluster using

```
kubectl apply -f kubernetes.yaml
```

If everything works, you can access your application by opening http://localhost:8000/. If you have something else running on port 8000, you need to change the load balancer port in `kubernetes.yaml`.

Tip: If you want to understand which pod your requests go to, you can add the value of `VaadinServletRequest.getCurrent().getLocalAddr()` somewhere in your UI.

Troubleshooting

If something is not working, you can try one of the following commands to see what is deployed and their status.

```
kubectl get pods
kubectl get services
kubectl get deployments
```

If the pods say `Container image "kaartje:latest" is not present with pull policy of Never` then you have not built your application using Docker or there is a mismatch in the name. Use `docker images ls` to see which images are available.

If you need even more information, you can run

```
kubectl cluster-info dump
```

that will probably give you too much information but might reveal the cause of a problem.

If you want to remove your whole deployment and start over, run

```
kubectl delete -f kubernetes.yaml
```