Java MVC Application

Part III.A – Custom MVC Framework - Integrating with FRONTEND

1. Pentru a adauga o interfata grafica aplicatiei de backend dezvoltate cu mini-framework-ul MCV realizat la workshopurile anterioare, adaugati in fork-ul vostru de pe Github aplicatia FrontendApp de pe <https://github.com/ZeroToHero-2016/FrontendCode> sau utilizati aplicatia dezvoltata in cadrul workshop-ului de Frontend.
2. In linie de comanda, navigati catre directorul radacina al aplicatiei FrontendApp, acolo unde sunt prezente fisierele *package.json* si *bower.json*. Rulati comenzile *npm install* si *bower install*. Care este scopul acestor comenzi?
3. Copiati instructiunile de mai jos in fisierul *web.xml* din aplicatia de backend. Ce rol credeti ca au acestea?

<**filter**>  
 <**filter-name**>CorsFilter</**filter-name**>  
 <**filter-class**>org.apache.catalina.filters.CorsFilter</**filter-class**>  
 <**init-param**>  
 <**param-name**>cors.allowed.origins</**param-name**>  
 <**param-value**>\*</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>cors.allowed.methods</**param-name**>  
 <**param-value**>GET,POST,PUT,DELETE,HEAD,OPTIONS</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>cors.allowed.headers</**param-name**>  
 <**param-value**>Content-Type,X-Requested-With,accept,Origin,Access-Control-Request-Method,Access-Control-Request-Headers</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>cors.exposed.headers</**param-name**>  
 <**param-value**>Access-Control-Allow-Origin,Access-Control-Allow-Credentials</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>cors.support.credentials</**param-name**>  
 <**param-value**>false</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>cors.preflight.maxage</**param-name**>  
 <**param-value**>10</**param-value**>  
 </**init-param**>  
</**filter**>  
<**filter-mapping**>  
 <**filter-name**>CorsFilter</**filter-name**>  
 <**url-pattern**>/\*</**url-pattern**>  
</**filter-mapping**>

1. Pentru inceput, porniti in modul debug aplicatia de backend. Acest lucru ne va permite sa identificam mai usor eventualele buguri, urmand ca dupa rezolvarea lor sa realizam deploy in masina virtuala Docker. Porniti simultan si aplicatia de frontend, asigurandu-va ca utilizati porturi diferite.
2. Accesati link-ul <http://localhost:9089/#/jobList>. Realizati modificarile necesare in aplicatia de frontend astfel incat sa se afiseze lista tuturor joburilor din baza de date.
3. Accesati link-ul View din grid-ul de job-uri. Realizati modificarile necesare astfel incat sa afiseze detaliile job-ului selectat
4. Accesati link-ul Delete din grid-ul de job-uri. Realizati modificarile necesare astfel incat inregistrarea sa fie stearsa din baza de date.
5. Accesati pagina de adaugare a unui job: <http://localhost:9089/#/jobAdd> si incercati sa adaugati un nou job. Realizati modificarile necesare astfel incat acesta sa fie salvat in baza de date. In mod similar, asigurati-va ca optiunea de Edit a unei inregistrari din lista de job-uri este functionala.
6. Realizati implementarile necesare pentru a avea functionalitatile de listare, vizualizare, stergere, creare si editare si pentru paginile de angajati, departamente si locatii.
7. Deployati aplicatia de backend pe masina Docker si realizati modificarile necesare in aplicatia de frontend, pentru a va conecta la aceasta.

Part III.B – Understanding Spring MVC

1. Creati o copie a aplicatiei de backend intr-un nou folder si adaugati sursele in proiectul din contul personal de Github. In fisierul *pom.xml* adaugati dependinte pentru framework-ul Spring MCV:

<spring.version>4.3.1.RELEASE</spring.version>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

1. Pentru integrarea aplicatie web cu framework-ul Spring, configurati DispatcherServlet-ul pus la dispozitie de acesta. Ce se va intampla cu MyDispatcherServlet?
2. Configurati framework-ul Spring indicandu-i pachetul de clase in care se afla Application Controllerele (utilizati un fisier mvc-dispatcher-servlet.xml, pe care il salvati in WEB\_INF).
3. Realizati modificarile necesare in clasa EmployeeController pentru a continua sa raspunda la request-urile HTTP cu metoda GET treansmise catre */employees/all.* Cum vom proceda pentru pune la dispozitia controllerului o instanta de EmployeeService fara a instantia in mod explicit clasa EmployeeServiceImpl?
4. Introduceti dependinte pentru urmatoarele librarii Jackson si configurati metodele din EmployeeController pentru a transmite inapoi raspunsuri in format JSON.

<!-- Jackson JSON Mapper -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.8.0</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.8.0</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

<version>2.8.0</version>

</dependency>

1. Care este diferenta dintre un RequestParam si un PathVariable?
2. Configurati metoda de afisare a unui angajat dupa id, din clasa EmployeeController, astfel incat sa primeasca id-ul angajatului sub forma de RequestParam. Metoda va raspunde la url-ul /employees/one.
3. Configurati metoda de afisare a unui angajat dupa id, din clasa EmployeeController, astfel incat sa primeasca id-ul angajatului sub forma de PathVariable. Metoda va raspunde tot la url-ul /employees/one. Cum se va modifica apelul acestui url?
4. Modificati clasa EmployeeController astfel incat sa proceseze urmatoarele url-uri:

/employees (GET - find All)

/employees/{id} (GET - find One)

/employees (POST - create, primeste pe request body un obiect JSON cu informatiile despre angajatul care se creeaza)

/employees (PUT - update, primeste pe request body un obiect JSON cu informatiile despre angajatul care se editeaza)

/employees/{id} (DELETE - delete)

1. (Pentru acasa) Configurati in mod similar lui EmployeeController si celelalte application controllere. Integrati aplicatia obtinuta cu aplicatia de frontend.