Java MVC Application

Part I Custom MVC Framework

1. Sincronizati repository-ul personal cu <https://github.com/ZeroToHero-2016/JavaDocs> si accesati folderul MVC/Workshop. Incarcati proiectul JavaMVCApp si editati fisierul pom.xml pentru a configura url-ul, username-ul si parola pentru tomcat. Exemplu:

<**plugin**>  
 <**groupId**>org.apache.tomcat.maven</**groupId**>  
 <**artifactId**>tomcat7-maven-plugin</**artifactId**>  
 <**version**>2.2</**version**>  
 <**configuration**>  
 <**url**>http://docker.local:49100/manager/text</**url**>  
 <**username**>admin</**username**>  
 <**password**>admin</**password**>  
 <**path**>/app</**path**>  
 </**configuration**>  
</**plugin**>

Editati datele de conectare la baza de date din fisierul DBProperties.java.

1. Creati un servlet care va avea rol de DispatcherServlet.

Suprascrieti metodele *doGet* si *doPost*.

1. Scrieti o metoda care sa preia sarcinile metodelor *doGet* si *doPost*. Denumiti metoda *dispatchReply.* Care vor fi parametrii metodei?
2. Conform arhitecturii MVC, impartiti sarcinile metodei *dispatchReply* in:

* Transmitere spre procesare – *dispatch()*
* Transmitere raspunsului obtinut in urma procesarii catre client – *reply()*
* Tratarea posibilelor erori de procesare – *sendExceptionError()*

1. Inregistrare custom DispatcherServlet in descriptorul aplicatiei web. Servletul va raspunde tuturor requesturilor de forma “*/mvc/\**”.

TEST:

<http://docker.local:49100/app/mvc/test>

1. Creare clasa *EmployeeController*

* Va apartine pachetului *ro.teamnet.zth.appl.controller*;
* Va contine metoda *getAllEmployees()* fara parametri;
* Metoda *getAllEmployees()* va returna variabila de tip *String* “allEmployees”.

1. Configurare DispatcherServlet astfel incat sa delege catre clasa *EmployeeController* procesarea requesturilor care incep cu “/employees”

TEST:

<http://docker.local:49100/app/mvc/employees>

<http://docker.local:49100/app/mvc/test>

<http://docker.local:49100/app/mvc/employees/all>

1. Creare clasa *DepartmentController*

* Va apartine pachetului *ro.teamnet.zth.appl.controller*;
* Va continemetoda *getAllDepartments()* fara parametric;
* Metoda *getAllDepartments* va returna variabila de tip *String* “allDepartments”.

1. Configurare DispatcherServlet astfel incat sa delege catre clasa *DepartmentController* procesarea requesturilor care incep cu “/departments”

TEST:

<http://docker.local:49100/app/mvc/departments>

1. Creare adnotare *MyController* in pachetul *ro.zth.api.annotations*

@Target({ElementType.TYPE})

@Retention(RUNTIME)

@Documented

public @interface MyController {

String urlPath();

}

Atributul *urlPath* va anunta DispatcherServletul-ul ca application controllerul poate prelucra un anumit requestPath primit de la client.

1. Pentru fiecare metoda a unui controller care poate face prelucrari de request-uri (pentru a imparti sarcina pe care si-o asuma controllerul), vom folosi adnotarea *MyRequestMethod*. Creati aceasta adnotare, cu atributele *methodType* si *urlPath*.

Atributul *urlPath* va anunta DispatcherServletul-ul ca poate prelucra un anumit *urlPath* primit de la client, in contextul controllerului de care apartine.

1. Se vor adauga metadate clasei *EmployeeController*, astfel incat controllerul sa anunte ca poate prelucra request-urile de forma “/employees”

Metoda *getAllEmployees* din clasa *EmployeeController* va ajuta Controllerul sa proceseze request-uri de forma *“/all”*, in contextul controllerului de care apartine. Astfel, request-urile de forma *“/employees/all”* vor fi prelucrate de catre metoda *getAllEmployees()* din clasa *EmployeeController*. Adaugati metadatele necesare metodei.

1. Se va adauga metoda *getOneEmployee()* in controller-ul *EmployeeController*.

* Metoda nu are parametri
* Metoda va returna o variabila de tip String cu valoarea “oneRandomEmployee”
* Metoda va prelucra request-uri de forma *“/one”* in contextul controllerului de care apartine.

1. Se vor adauga metadate clasei *DepartmentController*, astfel incat controllerul sa anunte ca poate prelucra request-urile de forma *“/departments”* .

Metoda getAllDepartments() va putea prelucra request-uri de tip *“/all”.* Adaugati metadatele necesare.

1. Ce modificari ar trebui puse in metoda *dispatch()* din DispatcherServlet astfel incat sa acomodam modificarile de la punctele **12** si **13** ?
2. Scanare si inregistrare controllere si metode care vor prelua sarcini de la DispatcherServlet.

In ce moment din lifecycle-ul aplicatiei ar fi cel mai convenabil sa cautam clasele cu rol de application controller?

1. Prin conventie, toate clasele cu rol de controller se vor afla in pachetul ro.teamnet.zth.appl.controller. Folosindu-va de clasa *AnnotationScanUtils.java*, incarcati toate clasele din pachetul *ro.teamnet.zth.appl.controller.*
2. In interiorul DispatcherServletului-ului, scrieti o metoda care pentru fiecare clasa din cele identificate la punctul 17 verifica daca poate primi request-uri de procesat de la DispatcherServletS; afisati in consola url-ul pe care controllerul respectiv il poate procesa.
3. Pentru fiecare clasa adnotata cu *MyController*, identificati toate metodele care pot procesa request-uri in contextul controllerului de care apartin. Afisati in consola url-ul pe care il poate procesa o metoda si tipul metodei (atributul methodType de pe adnotare).
4. Care dintre atributele identificate la punctele 18 si 19 identifica unic o metoda in contextual aplicatiei?
5. Salvati informatiile obtinute la puntele 18 si 19 intr-o colectie de tip Map. Care va fi cheia unica? Care va fi valoarea asociata cheii?
6. Modificati metoda *dispatch()* din DispatcherServlet, astfel incat controllerul si metoda care vor realiza procesarea sa fie preluate din Map-ul creat la punctul 21.

TEST:

<http://docker.local:49100/app/mvc/departments/all>

1. Creati clasa *JobController* cu metodele *getAllJobs* si *getOneJob*.
2. Creati clasa *LocationController* cu metodele *getAllLocations* si *getOneLocation*.