**Ինչ է Spring MVC – ն (v6․0․4 02․02․2023 դրությամբ)**

Spring Framework – ի կոմպոնենտ է որը թույլ է տալիս ստեղծել web app – ներ Java – ով։

App – ները ստեղծվում են Modev-View-Controller կոնցեպտի հիման վրա։

MVC – ն պատտերն է որը ունի 3 կոմպոնենտ՝

Model – տվյալների հետ աշխատելու լոգիկա

View – user -ի համար տեսանելի հատվածի լոգիկա (օր HTML), ինտերֆեյս

Controller – նավիգացիայի լոգիկա, request – ների մշակում

Spring MVC- ում կիրառելի են Spring Core – ի բոլոր հնարավորությունները։

Spring MVC – ն կազմված է Java կլասսներից և դրանցում ակտիվ օգտագործվող անոտացիաներից, HTML – ներից JS կոդի կիրառմամբ, կոնֆիգուրացիոն ֆայլերից Java/Anotation/Xml

Http request – ի և Controller – ների միջև կան նաև DispatcherServlet որը հանդիսանում է Spring MVC app – ի մուտքի կետը և որը Spring տրամադրում է ավտոմատ։ Request – ը սերվերին փոխանցվելուց հետո փոխանցվում է DispatcherServleti – ի որը այն ուղարկում է ճիշտ controller – ին։ Նշված մոդելը կոչվում է առաջնային controller – ի շաբլոն որտեղ կենտրոնական սերվլետը DispatcherServlet – ը տրամադրում է ընդհանուր ալգորիթմ request – ների մշակման համար իսկ ավելի նեղ մշակումը կատարում են առանձին կոմպոնոնտները (controller - ները)։ Սերվլետների նման DispatcherServlet – ի համար նույնպես պետք է mapping` Java – ի կամ web.xml – ի միջոցով:

AnnotationConfigWebApplicationContext context = new AnnotationConfigWebApplicationContext();

context.register(AppConfig.class);

// Create and register the DispatcherServlet

DispatcherServlet servlet = new DispatcherServlet(context);

ServletRegistration.Dynamic registration = servletContext.addServlet("app", servlet);

registration.setLoadOnStartup(1);

registration.addMapping("/app/\*");

web.xml – ի դեպքում՝

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/app-context.xml</param-value>

</context-param>

<servlet>

<servlet-name>app</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value></param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>app</servlet-name>

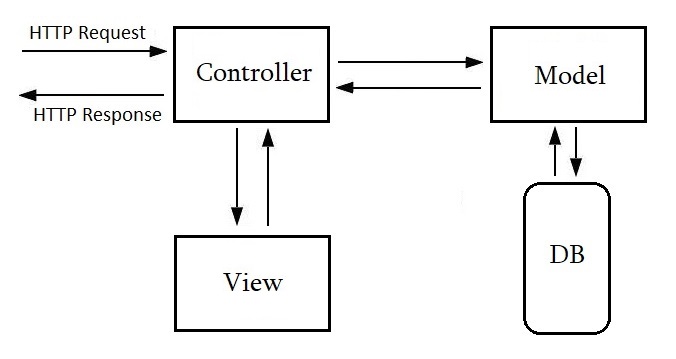
<url-pattern>/app/\*</url-pattern>

</servlet-mapping>

Controller- ը մշակում է request – ը, տվյալներով փոխանակվում model – ի հետ, user – ին ցույց է տալիս ճիշտ view, ըստ անհրաժեշտության վերահացեագրում է դեպի այլ ռեսուրսներ։

Model – ը պահում է իր մեջ տվյալներ, աշխատում է DB – ի հետ, controller – ին է տրամադրում տվյալներ։

View – ն տվյալները ստանում է controller – ից և արտապատկերում է դրանք բրաուզերում, HTML – ի դինամիկ գեներացման համար կարող են օգտագործվել շաբլոնիզաորներ օր՝ Thymeleaf, Freemarker, Velocity:

****

**@Controller** – ժառանգում է @Component անոտացիայից, այսինքն @ComponentScan – ը Controller – ի հետ աշխատում է նույն մեթոդով, եթե գտնում է անոտացիան ստեղծում է դրա bean – ը։ @Controller կլասսը կարող է իր մեջ պարունակել մեթոդներ ցանկացած անունով, որոնք համապատասխանում են 1 url – ի և սովորաբար վերադարձնում են String (պարտադիր չէ)։ Թե որ մեթոդը պետք է մշակի կոնկրետ request – ը որոշվում է mapping – ի միջոցով, որը սահմանվում մեթոդի վրա է HTTP request մեթոդներին համապատասխան անոտացիաներով՝ **@GetMapping, @PostMapping, @PutMapping, @DeleteMapping, @PatchMapping:** Երբեմն կարելի է հանդիպել հին սիտաքսին՝ **@RequestMapping(value = “/url” method = RequestMethod.GET):**

@RequestMapping – ը կարող է լինել նաև կլասսի վրա՝

@Controller

@RequestMapping(“/someUrl”)

public class MyController {}

Այս դեպքում մնացած բոլոր mapping մեթոդներըի հասցեները պետք է իրենց մեջ պարունակեն RequestMapping – ում նշված արժեքը։

Mapping – ը կապում է controller – ը հասցեի հետ որին կարելի է դիմել։