**Spring MVC Nedir?**

Spring MVC, web aplikasyonları yapmaya yarayan bir Java Frameworküdür. MVC’nin açılımı Model-View-Controllerdir. Bu şekildeki katmanlı tasarım mimarisinin amacı, mimariyi parçalayarak işlerin kolay yönetilmesini sağlamaktır.

**Model bileşeni** mimarinin veri katmanıdır. Veri ile alakalı tüm işlemler bu katmanda gerçekleştirilir. Model katmanı View ve Controller arasında veya işle ilgili diğer herhangi veri aktarımı işleminde gerekli olan bir katmandır.

**Controller bileşeni** Model ve View katmanı arasında bir arayüz görevi görür. Tüm iş mantığı gelen taleplerle alakalı işlemlerin gerçekleştirildiği, Model ve View katmanları arasındaki etkileşimin sağlandığı yerdir.

**View bileşeni** uygulamanın UI kısmı için kullanılır. Kullanıcı ile etkileşimde olan kısım bu kısımdır.

**Front Controller** Spring MVC’de dispatcherServlet sınıfı arkada bir controller gibi çalışır. Bu katman Spring MVC’nin adımlarını organize etmekten sorumludur.

**Spring Web MVC’nin çalışma mantığı**



Figürde gösterildiği gibi bütün katmanlar DispatcherServlet’den gelen istemleri içermekte. DispatcherServlet XML dosyasından aldığı girdiyi Controller’a iletir. Controller ise Model ve View’a bir obje döndürür. DispatcherServlet XML dosyasının view resolverinin girişini kontrol eder ve belirtilen görünüm bileşenini çağırır.

**Spring MVC FrameWork’ünün Faydaları**

* Ayrılmış roller
* Hafiflik
* Güçlü konfigürasyon
* Hızlı gelişim
* Test etme kolaylığı
* Esnek Haritalama

**DispatcherServlet**

Spring MVC’nin kullandığı bir Front-Controller sınıfıdır.

**Controller class**

Controller classı üretilirken iki anotasyon kullanılır. @ Conroller ve @ RequestMapping @ Controller anotasyonu bulunduğu classın Controller class olduğunu belirtmekte kullanılır. @ RequestMapping anotasyonu ise classı belirtilen URL ile eşlemekte kullanılır.

**Spring Anotasyonları**

**@RequestMapping**

Controller Classında belirtilen URL ile eşlemekte kullanılır.

Kullanım şekli örneği:

*@Controller*

*class VehicleController {*

*@RequestMapping(value = "/vehicles/home", method = RequestMethod.GET)*

*String home() {*

*return "home";*

*}*

*}*

**@RequestBody**

Http'nin requestini bir nesne ile eşlemekte kullanılır.

Kullanım şekli örneği:

*@PostMapping("/save")*

*void saveVehicle(@RequestBody Vehicle vehicle) {*

*// ...*

*}*

**@PathVariable**

Bu anotasyon Metod argümentinin URI değişkenine bağlı olduğunu gösterir. Birden fazla kullanım şekli vardır.

Kullanım şekli örnekleri:

*@RequestMapping("/{id}")*

*Vehicle getVehicle(@PathVariable("id") long id) {*

*// ...*

*}*

*@RequestMapping("/{id}")*

*Vehicle getVehicle(@PathVariable long id) {*

*// ...*

*}*

*@RequestMapping("/{id}")*

*Vehicle getVehicle(@PathVariable(required = false) long id) {*

*// ...*

*}*

**@RequestParam**

Bu anotasyon HTTP istek parametresine ulaşmak için kullanılır. Birden fazla kullanım şekli vardır.

Kullanım şekli örnekleri:

*@RequestMapping*

*Vehicle getVehicleByParam(@RequestParam("id") long id) { // ...*

*}*

*@RequestMapping("/buy")*

*Car buyCar(@RequestParam(defaultValue = "5") int seatCount) { // ...*

*}*

**@ResponseBody**

Eğer bir metodu bu anotasyon ile işaretlersek, Spring sonuçları bizzat metodun kendisine gönderir.

Kullanım Şekli örneği:

*@ResponseBody*

*@RequestMapping("/hello")*

*String hello() {*

*return "Hello World!";*

*}*

**@ExceptionHandler**

Bu anotasyon ile biz özel olarak bir hata belirtebiliriz ve bu sayede Spring belirttiğimiz hata ile karşılaştığında bu metodu dönderebilir. Javadaki try-catch gibi düşünülebilir.

Kullanım şekli örneği:

*@ExceptionHandler(IllegalArgumentException.class)*

*void onIllegalArgumentException(IllegalArgumentException exception) {*

*// ...*

*}*

**@ResponseStatus**

Bu anotasyon ile yanıtın istenen HTTP durumunu belirtebiliriz.

Kullanım şekli örneği:

*@ExceptionHandler(IllegalArgumentException.class)*

*@ResponseStatus(HttpStatus.BAD\_REQUEST)*

*void onIllegalArgumentException(IllegalArgumentException exception) {*

*// ...*

*}*

**@RestController**

@Controller ve @ResponseBody anotasyonları ile birlikte kullanılan bir anotasyondur.

Kullanım şekli örneği:

*@Controller*

*@ResponseBody*

*class VehicleRestController {*

*// ...*

*}*

**@ModelAttribute**

Bu anotasyonla, MVC @Controller modelin zaten içerisinde olan elementlere erişebiliriz.

Kullanım şekli örnekleri:

*@PostMapping("/assemble")*

*void assembleVehicle(@ModelAttribute("vehicle") Vehicle*

*vehicleInModel) {*

*// ...*

*}*

*@PostMapping("/assemble")*

*void assembleVehicle(@ModelAttribute Vehicle vehicle) {*

*// ...*

*}*

*@ModelAttribute("vehicle")*

*Vehicle getVehicle() {*

*// ...*

*}*

*@ModelAttribute*

*Vehicle vehicle() {*

*// ...*

*}*

**@CrossOrigin**

Bu anotasyon işaretlendiği metod için çapraz alan iletişimine olanak sağlar.

Kullanım alanı örneği:

*@CrossOrigin*

*@RequestMapping("/hello")*

*String hello() {*

*return "Hello World!";*

*}*