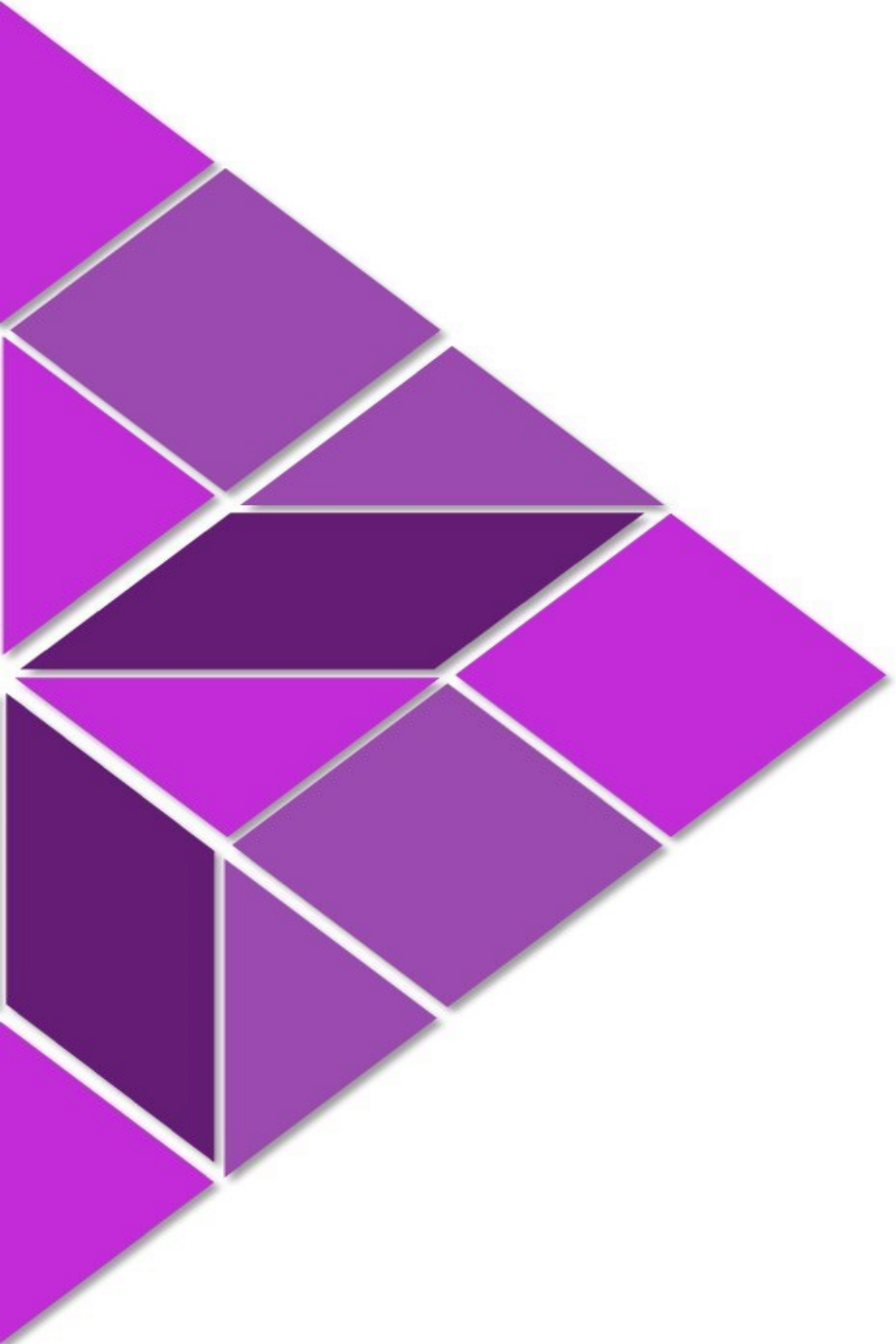


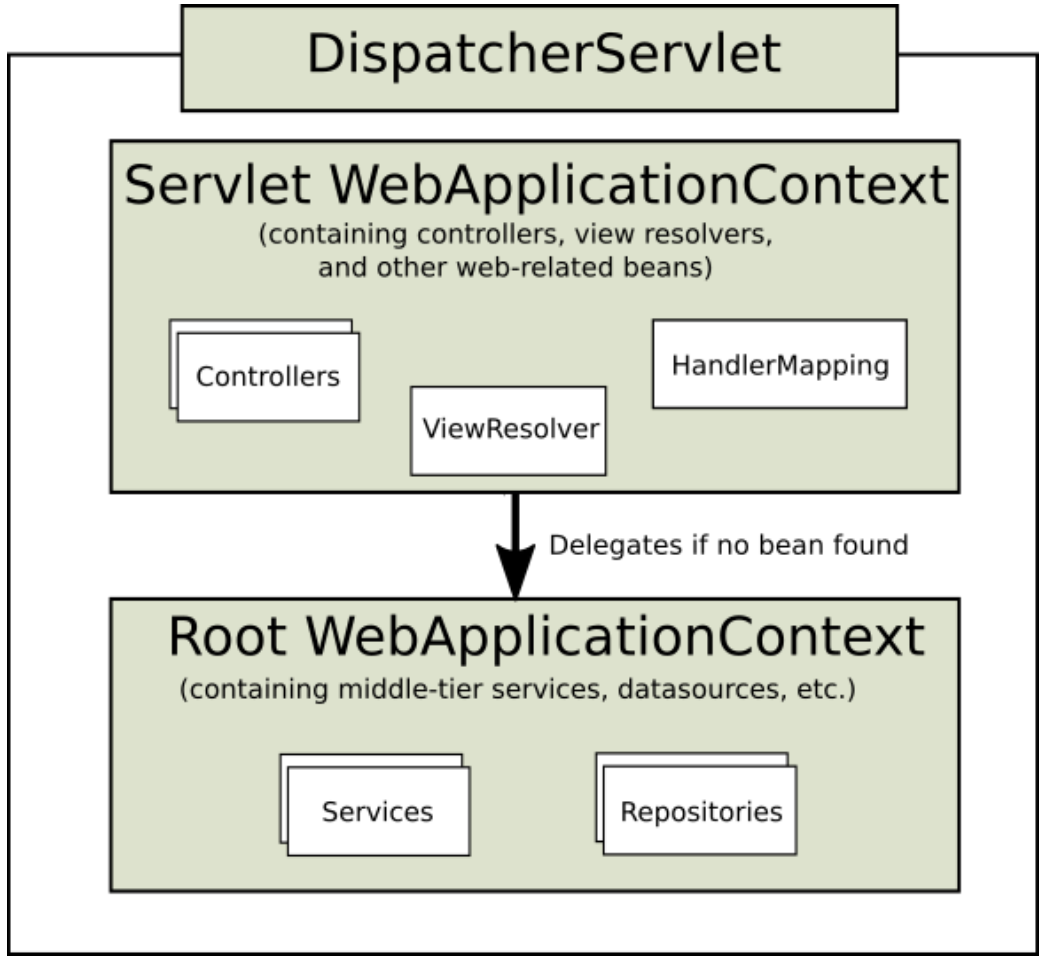
Spring Web MVC

Presented by Hosein Zare
Twitter: @zare88r

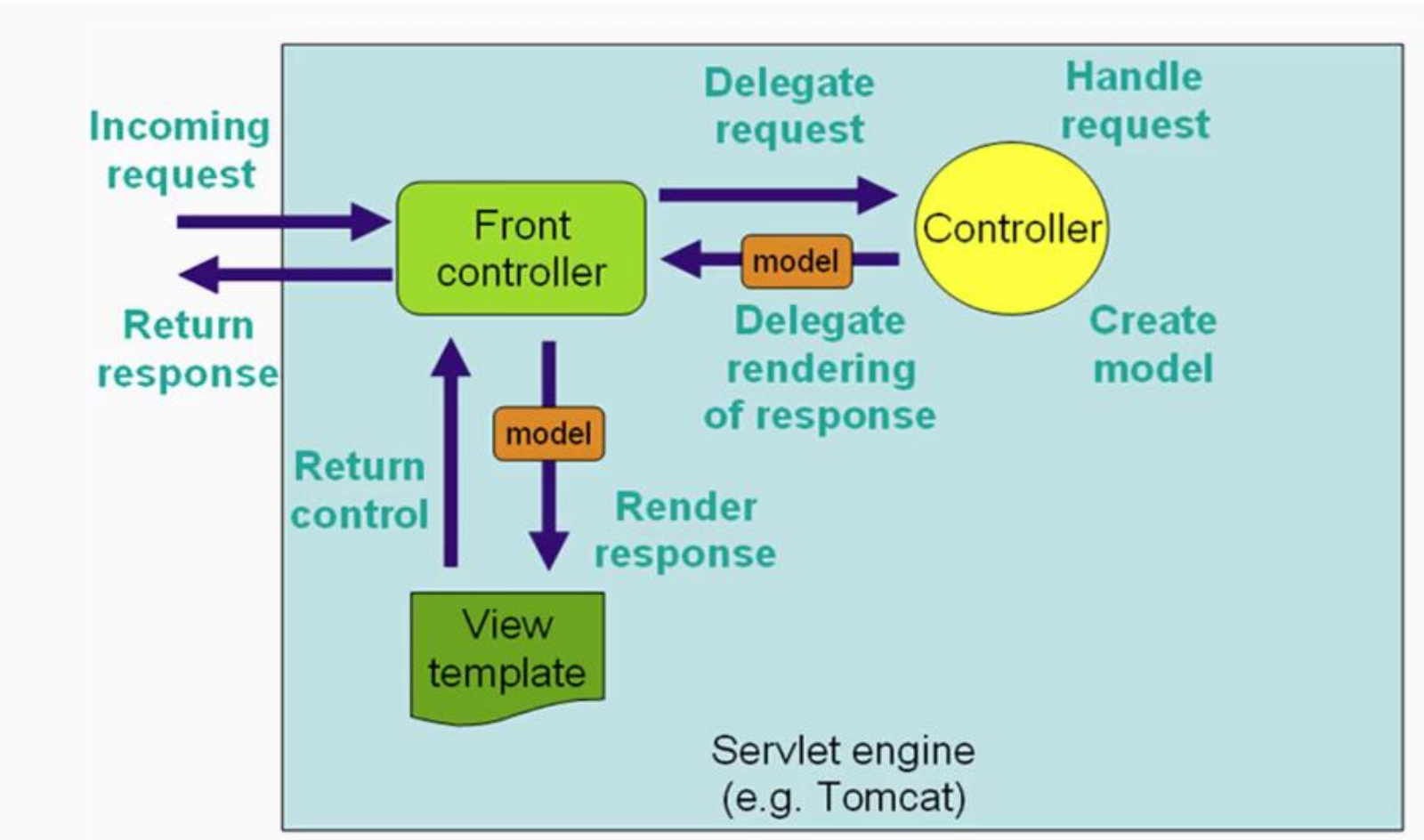
مقدمه

- Spring Web MVC یک فریم ورک تحت وب است که بر مبنای Servlet API تولید شده است
- این فریم ورک براساس الگوی Front Controller طراحی شده است
- نام Servlet اصلی Dispatcher Servlet میباشد
- این servlet درخواست ها را تحلیل میکند و به کامپوننت مورد نظر در spring container ارجاع میدهد

مقدمه


DispatcherServlet
Servlet WebApplicationContext
(containing controllers, view resolvers,
and other web-related beans)
Controllers
ViewResolver
HandlerMapping
Delegates if no bean found
Root WebApplicationContext
(containing middle-tier services, datasources, etc.)
Services
Repositories

مقدمه


Incoming request
Delegate request
Handle request
Front controller
Controller
model
Return response
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

Bean های مخصوص Spring MVC

- HandlerMapping
- HandlerAdapter
- HandlerExceptionResolver
- ViewResolver
- LocaleResolver, LocaleContextResolver
- ThemeResolver
- MultipartResolver
- FlashMapManager

Controller

```
server.port=4041
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
```

```
@Controller
@RequestMapping("/index")
public class HomeController {
    @RequestMapping(method = RequestMethod.GET)
    public String index() {
        return "home";
    }
    @RequestMapping(method = RequestMethod.GET , value = "/p2")
    public String page2() {
        return "page2";
    }
}
```

View Technologies

- Thymeleaf 
- FreeMarker 

- Groovy Markup
- JSP & JSTL
- Tiles 

View Resolver

- XmlViewResolver
- UrlBasedViewResolver
- InternalResourceViewResolver
- FreeMarkerViewResolver
- ContentNegotiatingViewResolver

Request Mapping

- @RequestMapping به شما اجازه میدهد یک controller را به یک url path نگاشت دهید.
- همچنین میتوانید از annotation های زیر استفاده کنید:
  - @GetMapping
  - @PostMapping
  - @PutMapping
  - @DeleteMapping
  - @PatchMapping

متود ها

- در ورودی متود های داخل Controller اگر از اشیاء از قبل تعریف شده ای استفاده کنید spring به طور خودکار آنها را پر کرده و در اختیارتان قرار میدهد
- خروجی متود شما برای spring mvc معنا دار میباشد

ورودی متود ها

WebRequest	java.io.InputStream, java.io.Reader	@ModelAttribute
NativeWebRequest	java.io.OutputStream, java.io.Writer	Errors, BindingResult
javax.servlet.ServletRequest	@PathVariable // GET /pets/42;q=11;r=22	@SessionAttribute
javax.servlet.ServletResponse	@MatrixVariable // GET /pets/42;q=11;r=22	SessionStatus
javax.servlet.http.HttpSession	@RequestParam // GET /pets?id=42	java.util.Map
javax.servlet.http.PushBuilder	@RequestHeader	org.springframework.ui.Model
java.security.Principal	@CookieValue	org.springframework.ui.ModelMap
HttpMethod	@RequestBody	
java.util.Locale	HttpEntity<B>	
java.util.TimeZone	@RequestPart	
java.time.ZoneId	RedirectAttributes	

خروجی متود ها

`@ResponseBody`
`HttpEntity<B>`
`ResponseEntity<B>`
`HttpHeaders`
`String`
`View`
`ModelAndView`
`java.util.Map`
`org.springframework.ui.Model`
`@ModelAttribute`
`Any other return value (RequestToViewNameTranslator )`

Exception Handling

- برای هندل کردن exception ها میتوانید متودهایی با @ExceptionHandler تعریف کنید . این متود ها به صورت عمومی تر میتوانند در کلاس هایی با @ControllerAdvice تعریف شوند

```
@ExceptionHandler
@ResponseStatus(HttpStatus.BAD_REQUEST)
public ModelAndView handleError(HttpServletRequest req, Exception ex)
{
    ModelAndView modelMap = new ModelAndView("handler");
    modelMap.addObject("exception", ex);
    modelMap.addObject("url", req.getRequestURL());
    return modelMap;
}
```

JSR 303: Bean Validation

- سنجش اعتبار (validating) به عملیات مشترک در فرم های لایه نمایش در اپلیکیشن ها اطلاق میشود
- در روش های مختلف ممکن است کد نویسی اعتبار سنجی در لایه های مختلف نوشته شود که این ممکن است وقت گیر باشد.
- برای جلوگیری از تکرار آن JSR 303 یک سری annotation برای تعریف validation ها معرفی میکند که مستقیما در لایه لاجیک سیستم روی مدل ها تعریف میشود.

JSR 303: Bean Validation

```
public class Order {
    private Long orderId;
    private String orderName;
    private String orderAddress;
    private Boolean shipped;
    @NotNull
    public Long getOrderId() {
        return orderId;
    }
    @NotNull
    @Size(min = 2, max = 10)
    public String getOrderName() {
        return orderName;
    }
    @NotNull
    @AssertTrue
    public Boolean getShipped() {
        return shipped;
    }
```

JSR 303: Bean Validation

```
@GetMapping("/save-order")
public String save2(@Valid Order order
        , BindingResult bindingResult, ModelMap modelMap) {
    if (bindingResult.hasErrors()) {
        modelMap.addAttribute("result",bindingResult.getAllErrors());
        modelMap.addAttribute("order",order);
        return "page2";
    }
    return "page3";
}
```