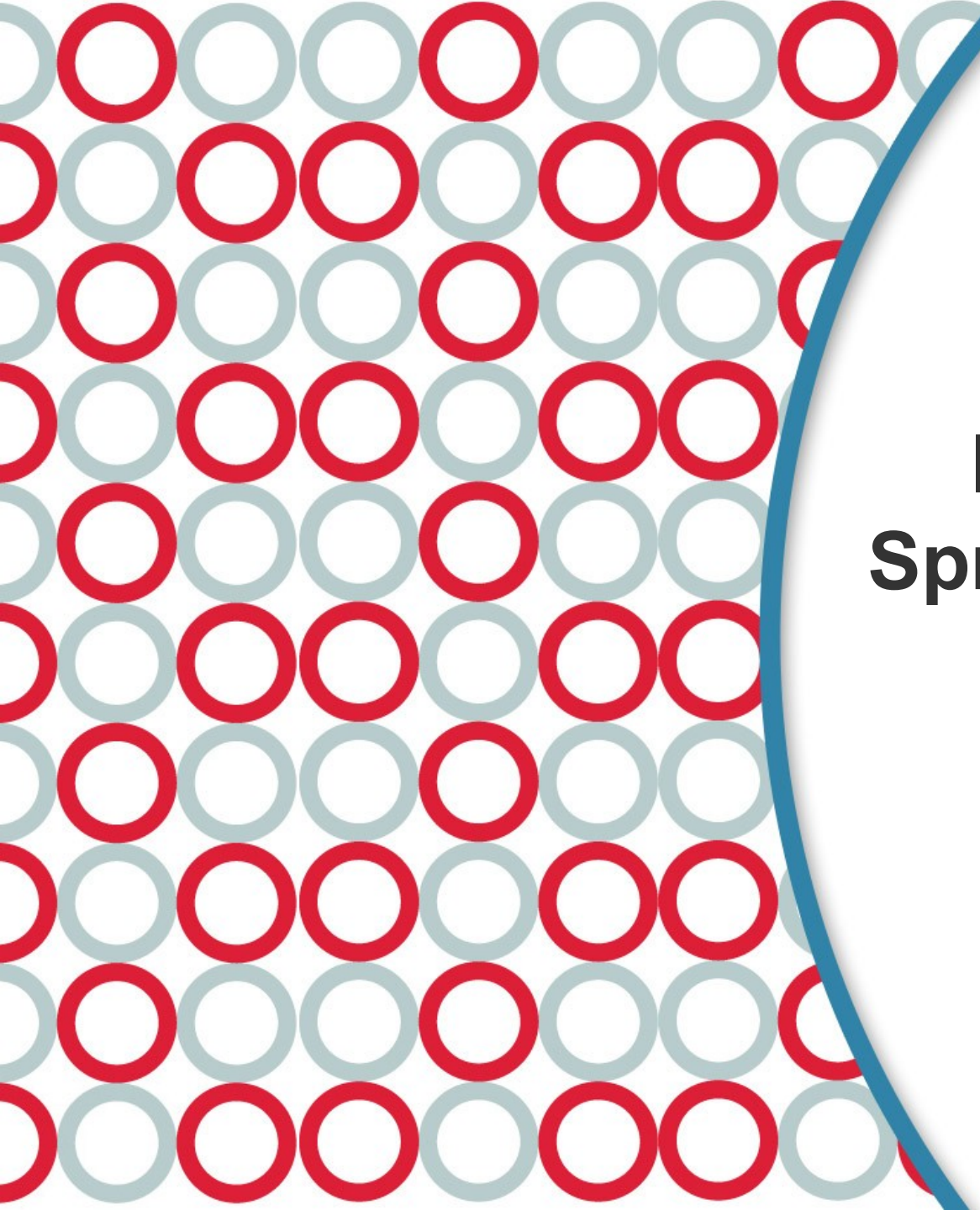


April 2018, IPT Course
Introduction to Spring 5

Introduction to Spring Web MVC

Trayan Iliev

tiliev@iproduct.org
http://iproduct.org

Agenda for This Session

- Spring MVC architecture: DispatcherServlet, HandlerMappingm Controller, ViewResolver, View
- Using JSP+JSTL and Thymeleaf templates
- Building simple controller - @Controller, @RequestMapping
- Passing data to the View using Model, ModelMap, and ModelAndView
- Getting data with a request parameter – @RequestParam
- Building a sample Spring MVC application.
- Configuring dispatcher, resolver, static resources, locale, multipart, error handling and encoding with Spring Boot and Spring embedded servlet container (Jetty, Tomcat).

Where to Find the Demo Code?

Introduction to Spring 5 demos and examples are available @ GitHub:

https://github.com/iproduct/course-spring5

MVC Comes in Different Flavors

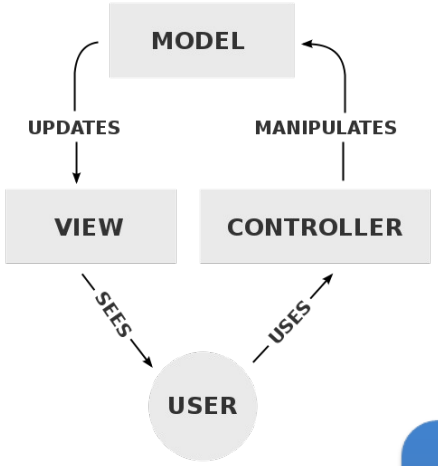
What is the difference between following patterns:

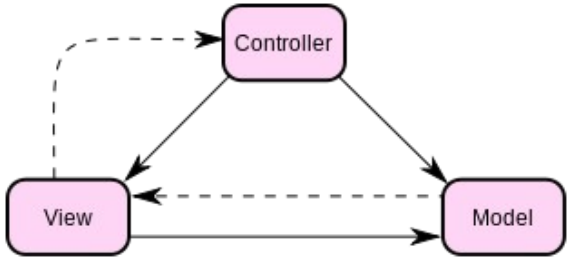
- Model-View-Controller (MVC)
- Model-View-ViewModel (MVVM)
- Model-View-Presenter (MVP)

http://csl.ensm-douai.fr/noury/uploads/20/ModelViewController.mp3

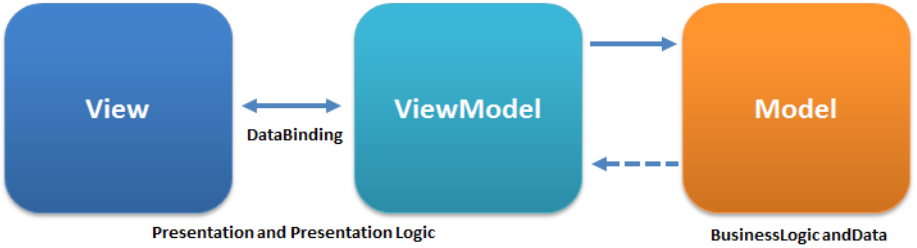
MVC Comes in Different Flavors - II

- MVC

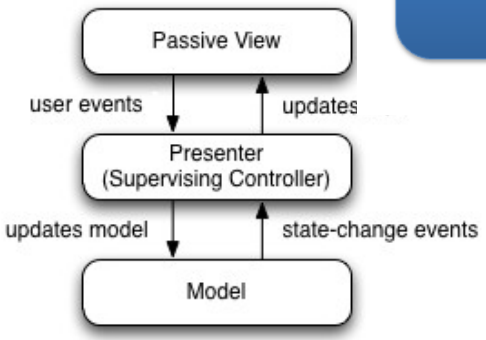




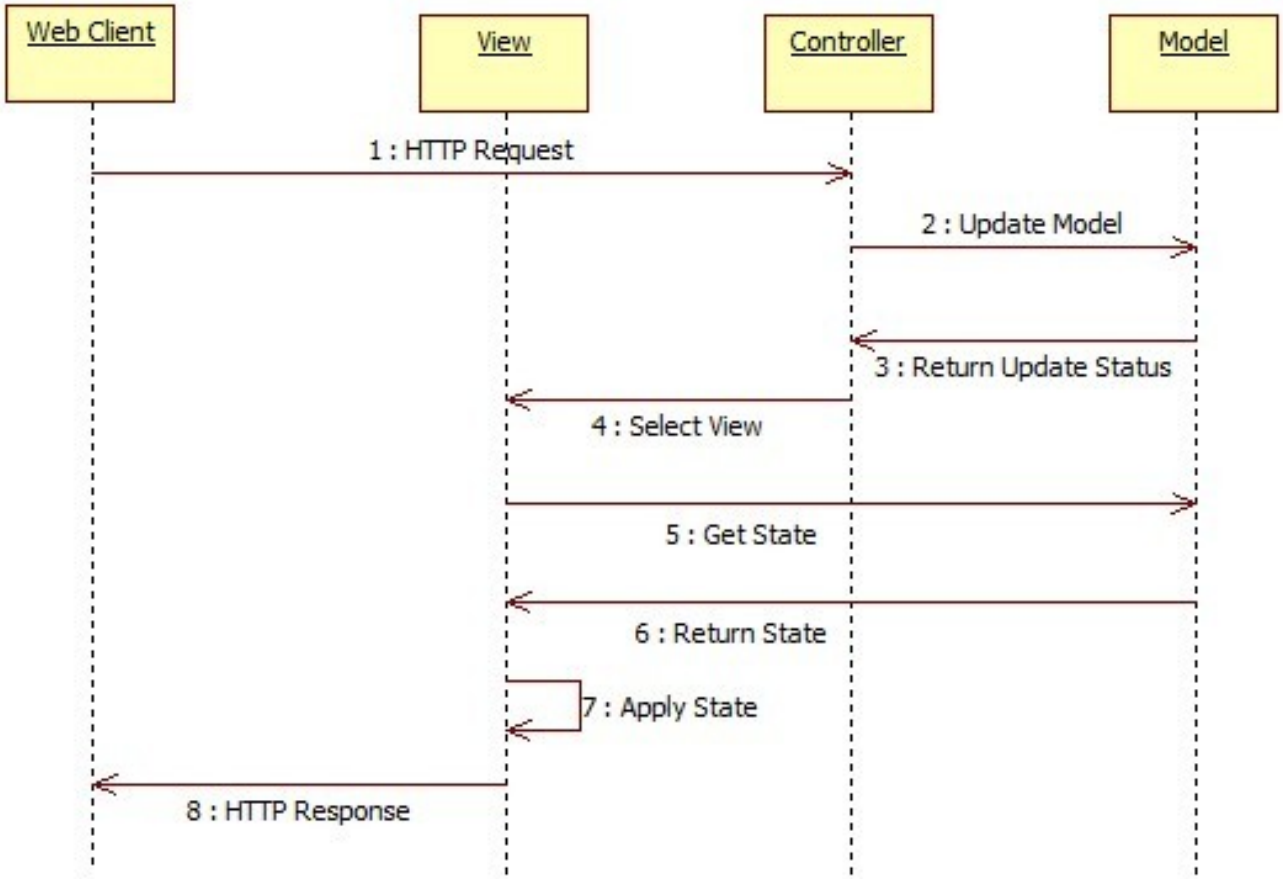
- MVVM



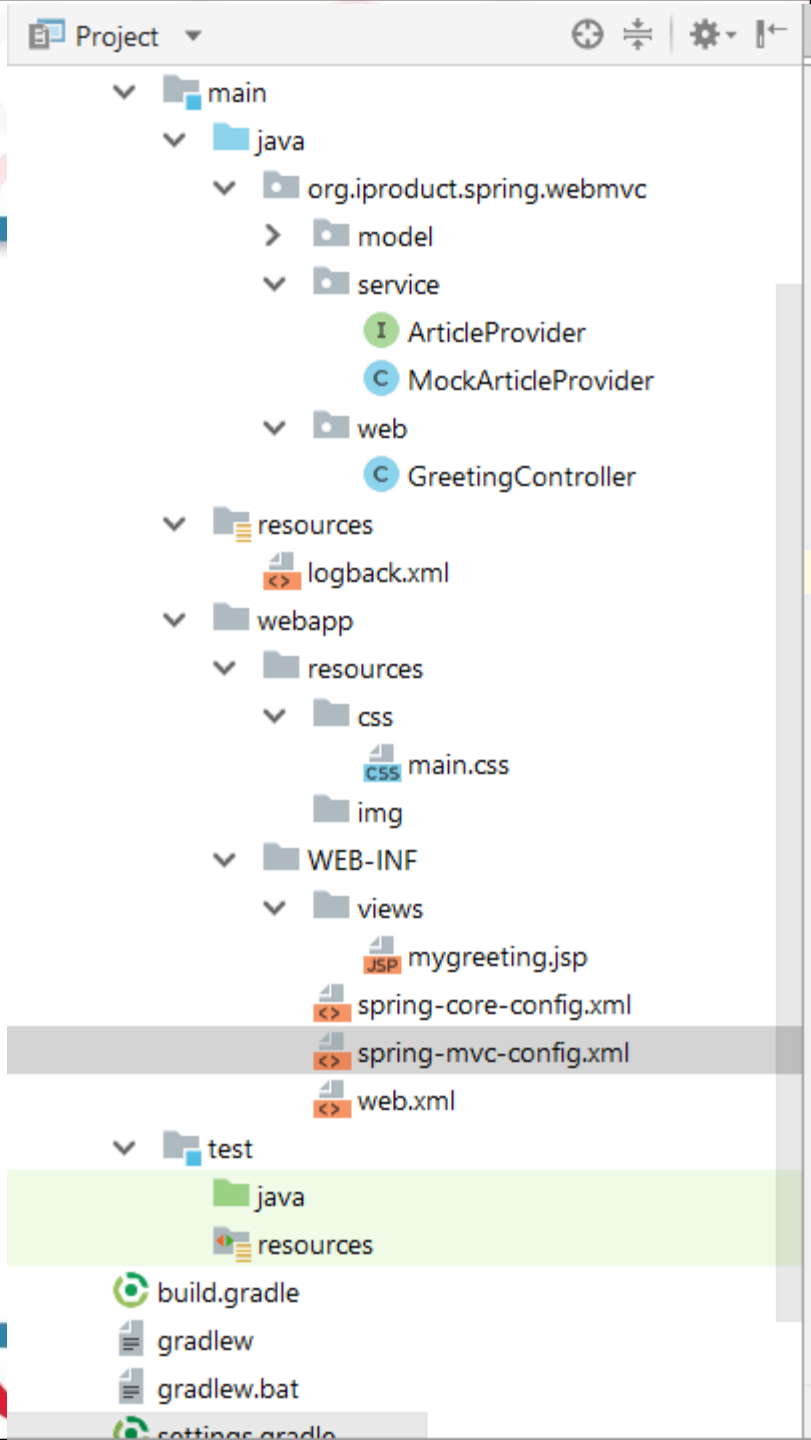
- MVP



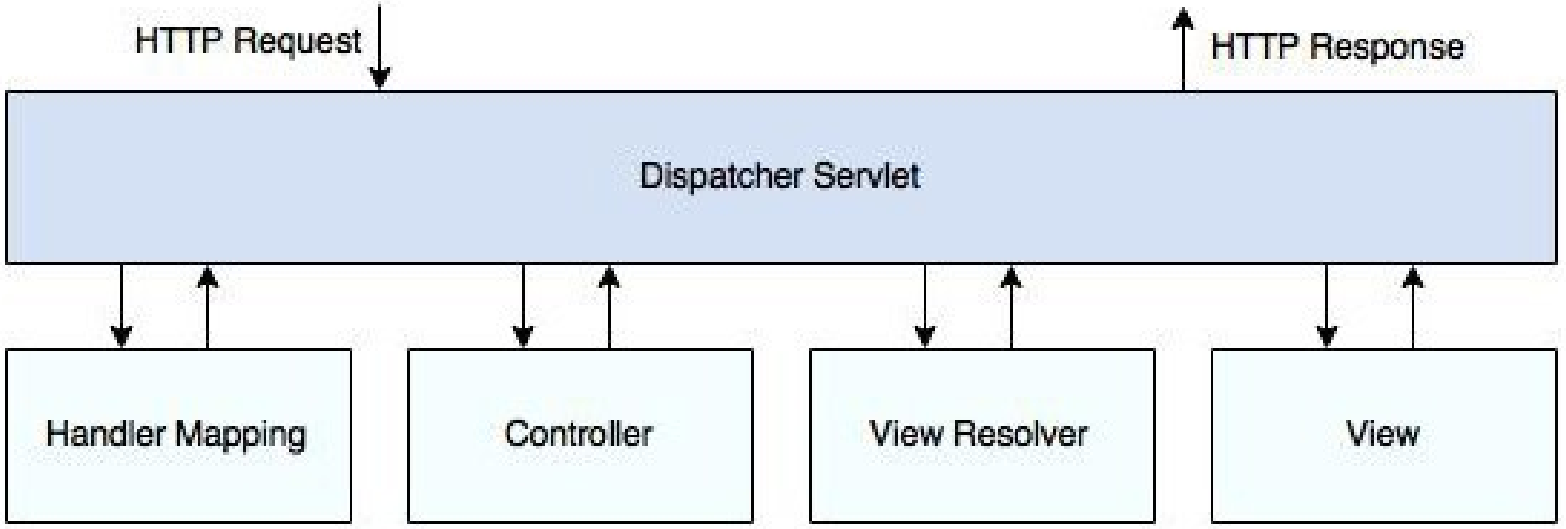
Web MVC Interactions Sequence Diagram



Web MVC Project Structure

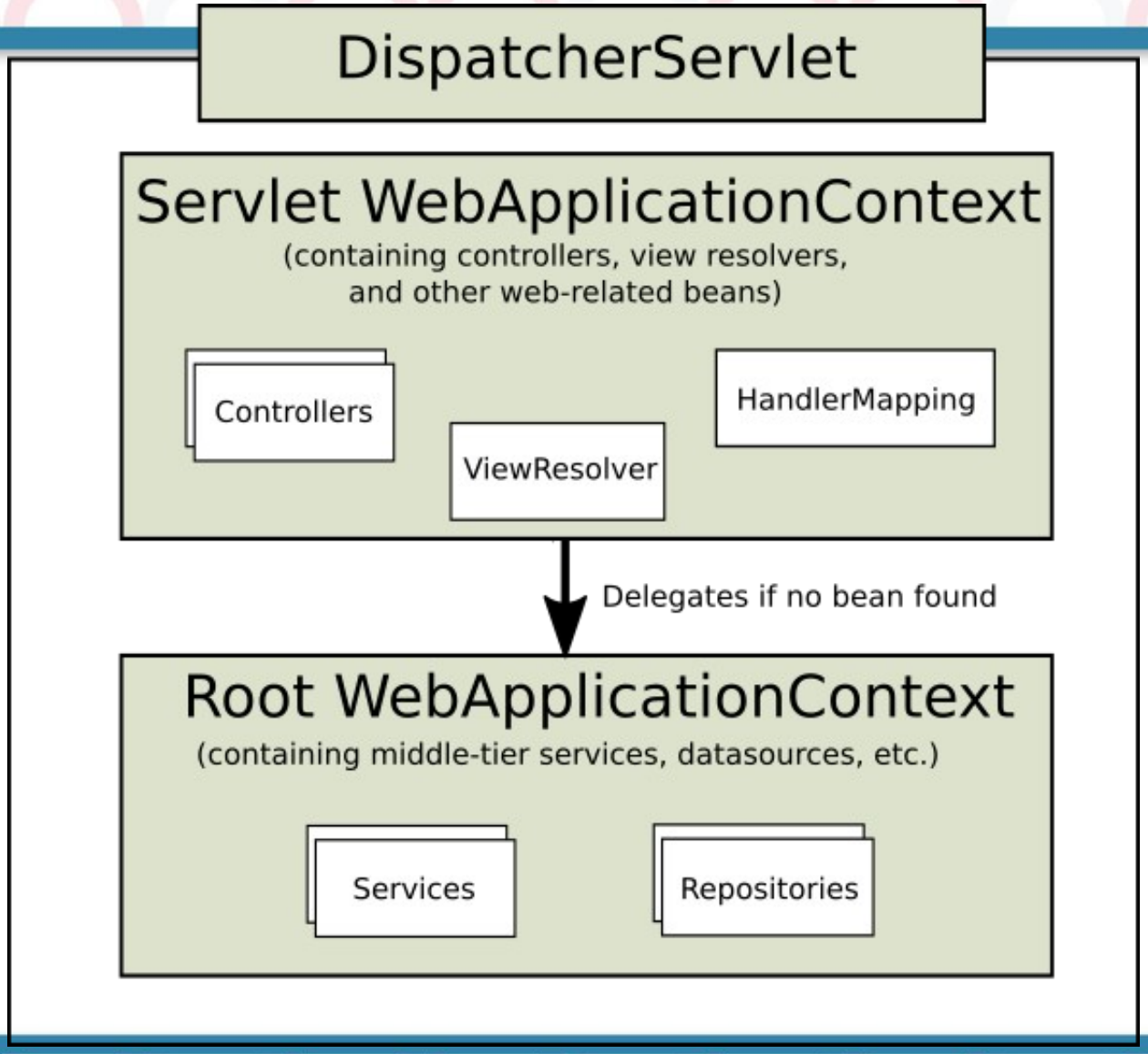


Web MVC Main Components



Source: https://docs.spring.io/spring-framework/docs/

How Dependency Injection Works?



Source: https://docs.spring.io/spring-framework/docs/

Web MVC Request Handling

- Handler mapping maps all the request mappings in the controllers. This happens at server startup.
- Based on the request that gets hit, dispatcher servlet asks the handler mapping as to which equivalent controller method needs to be invoked.
- The controller after completing it's task finally returns a view, model and view or a simple string to the dispatcher servlet.
- The dispatcher servlet then checks with the view resolver and finally resolves the view returned in step 3, integrates the model to the view and renders it on the browser.

Source: https://docs.spring.io/spring-framework/docs/

Web MVC Main Annotations

- @Controller – marks class as Spring MVC controller
- @RequestMapping – Annotation for mapping web requests onto methods in @Controller classes: path (value), method, params, headers, consumes, produces, name
- @GetMapping – shortcut for @RequestMapping(method = RequestMethod.GET)
- @PostMapping = @RequestMapping(method = POST)
- @PutMapping = @RequestMapping(method = PUT)
- @DeleteMapping = @RequestMapping(method = DELETE)
- @PatchMapping = @RequestMapping(method = PATCH)

Source: https://docs.spring.io/spring-framework/docs/

Web MVC Main Annotations - Example

```
@RestController
@RequestMapping("/persons")
class PersonController {

	@GetMapping("/{id}")
	public Person getPerson(@PathVariable Long id) {
		// ...
	}

	@PostMapping
	@ResponseStatus(HttpStatus.CREATED)
	public void add(@RequestBody Person person) {
		// ...
	}
}
```


Source: https://docs.spring.io/spring-framework/docs/

Problem 2 - REST API for Comments

We want to develop a simple REST/JSON API for comments service from Problem 1 with Create-Read-Update-Delete (CRUD) functionality.

- Implement a collection level resource: **/api/posts**, serving collection GET (GET all posts) and POST requests. In response of a POST request the REST endpoint should return HTTP status: 201 Created with a Location header containing the individual level resource URI of newly created resource – Eg.: /api/posts/16, if new resource has ID=16.
- Implement individual level resource: **/api/posts/{postId}**, serving read (GET), update (PUT) and delete (DELETE) requests.

Spring MVC Special Bean Types - I

- HandlerMapping – map a request to a handler along with a list of HandlerInterceptor`s for pre- and post-processing
- HandlerAdapter – helps the DispatcherServlet to invoke a handler mapped to a request (shields it from details of resolving @Controller annotations, etc.)
- HandlerExceptionResolver – strategy to resolve exceptions possibly mapping them to handlers, or to HTML error views
- ViewResolver – resolves logical String-based view names returned from a handler to an actual View to render
- LocaleResolver & LocaleContextResolver – Resolves the Locale a client is using and possibly their time zone, in order to be able to offer internationalized views

Source: https://docs.spring.io/spring-framework/docs/

Spring MVC Special Bean Types - II

- ThemeResolver – resolves themes your web application can use, for example, to offer personalized layouts
- MultipartResolver – abstraction for parsing a multi-part request (e.g. browser form file upload) with the help of some multipart parsing library.
- FlashMapManager – Stores and retrieves the "input" and the "output" FlashMap that can be used to pass attributes from one request to another, usually across a redirect.
- For each type of special bean, the DispatcherServlet checks for the WebApplicationContext first. If there are no matching bean types, it falls back on the default types listed in DispatcherServlet.properties

Source: https://docs.spring.io/spring-framework/docs/

DispatcherServlet Processing Sequence - I

- The WebApplicationContext is searched for and bound in the request as an attribute that the controller and other elements in the process can use. It is bound by default under the key:
  DispatcherServlet.WEB_APPLICATION_CONTEXT_ATTRIBUTE
- The locale resolver is bound to the request to enable elements in the process to resolve the locale to use when processing the request (preparing data, rendering the view, and so on).
- The theme resolver is bound to the request to let elements such as views determine which theme to use. If you do not use themes, you can ignore it.

Source: https://docs.spring.io/spring-framework/docs/

DispatcherServlet Processing Sequence - II

- If you specify a multipart file resolver, the request is inspected for multiparts; if multiparts are found, the request is wrapped in a MultipartHttpServletRequest for further processing by other elements in the process.
- An appropriate handler is searched for. If a handler is found, the execution chain associated with the handler (preprocessors, postprocessors, and controllers) is executed in order to prepare a model or rendering. Or alternatively for annotated controllers, the response may be rendered (within the HandlerAdapter) instead of returning a view.
- If a model is returned, the view is rendered. If no model is returned, no view is rendered, because the request could already have been fulfilled.

Source: https://docs.spring.io/spring-framework/docs/

DispatcherServelt **init-params**

- contextClass – class implementing WebApplicationContext, (XmlWebApplicationContext by default)
- contextConfigLocation – String indicating where context(s) can be found (using a comma as a delimiter)
- namespace – WebApplicationContext namespace ([servlet-name]-servlet by default)
- throwExceptionIfNoHandlerFound – whether to throw a NoHandlerFoundException if no request handler was found. Exception can be caught by HandlerExceptionResolver, e.g. via an @ExceptionHandler controller method, and handled. By default is set to "false" → response status to 404 (NOT_FOUND), without rising an exception.

Source: https://docs.spring.io/spring-framework/docs/

View Resolvers

- AbstractCachingViewResolver
- XmlViewResolver
- ResourceBundleViewResolver
- UrlBasedViewResolver
- InternalResourceViewResolver
- FreeMarkerViewResolver
- ContentNegotiatingViewResolver

Source: https://docs.spring.io/spring-framework/docs/

Spring MVC Models

- Model - holder for model attributes. Primarily designed for adding attributes to the model. Allows for accessing the overall model as a java.util.Map.
- ModelMap – implementation of Map for use when building model data for use with UI tools. Supports chained calls and generation of model attribute names.
- ModelAndView – holds both Model and View, making possible for a controller to return both model and view in a single return value.
- @ModelAttribute – binds a method parameter or method return value to a named model attribute and then exposes it to a web view

Source: https://docs.spring.io/spring-framework/docs/

Handler Method Parameters - I

- WebRequest, NativeWebRequest
- javax.servlet.ServletRequest, javax.servlet.ServletResponse
- javax.servlet.http.HttpSession
- javax.servlet.http.PushBuilder
- java.security.Principal
- HttpMethod
- java.util.Locale
- java.util.TimeZone + java.time.ZoneId
- java.io.InputStream, java.io.Reader
- java.io.OutputStream, java.io.Writer

Source: https://docs.spring.io/spring-framework/docs/

Handler Method Parameters - II

- @PathVariable
- @MatrixVariable
- @RequestParam
- @RequestHeader
- @CookieValue
- @RequestBody
- HttpEntity<B>
- @RequestPart
- RedirectAttributes

Source: https://docs.spring.io/spring-framework/docs/

Handler Method Parameters - III

- @ModelAttribute
- Errors, BindingResult
- SessionStatus + class-level @SessionAttributes
- UriComponentsBuilder
- @SessionAttribute
- @RequestAttribute
- Any other argument

Source: https://docs.spring.io/spring-framework/docs/

Handler Mapping Interceptors

- Applications can register any number of existing or custom interceptors for certain groups of handlers, to add common pre-processing behavior without needing to modify each handler implementation.
- Interceptors must implement HandlerInterceptor interface:
  - preHandle(..) – before the actual handler is executed, returns boolean value indicating if the request handling should proceed
  - postHandle(..) – after the handler is executed, can expose additional models to the view via ModelAndView
  - afterCompletion(..) – after the request completion (after rendering the view), allows for proper resource cleanup

Source: https://docs.spring.io/spring-framework/docs/

Exceptions Handling

- If exception is thrown by a @Controller, DispatcherServlet delegates it to a chain of HandlerExceptionResolver beans to provide alternative handling, typically an error response.
- SimpleMappingExceptionResolver - mapping between exception class names and error view names (error pages)
- DefaultHandlerExceptionResolver – maps exceptions to HTTP status codes.
- ResponseStatusExceptionResolver – resolves exceptions with the @ResponseStatus annotation and maps them to HTTP status codes based on the value in the annotation.
- ExceptionHandlerExceptionResolver – @ExceptionHandler method in @Controller / @ControllerAdvice class.

Source: https://docs.spring.io/spring-framework/docs/

@ExceptionHandler

```
@ExceptionHandler ({CustomValidationException.class,
                    FileSystemException.class})
@Order(10)
public ResponseEntity<String> handle(Exception ex) {
    LOG.error("Article Controller Error:",ex);

    ...
    return ResponseEntity.badRequest().body(ex.toString());
}
```

Error Pages

```
<error-page>
   <location>/error</location>
</error-page>
```

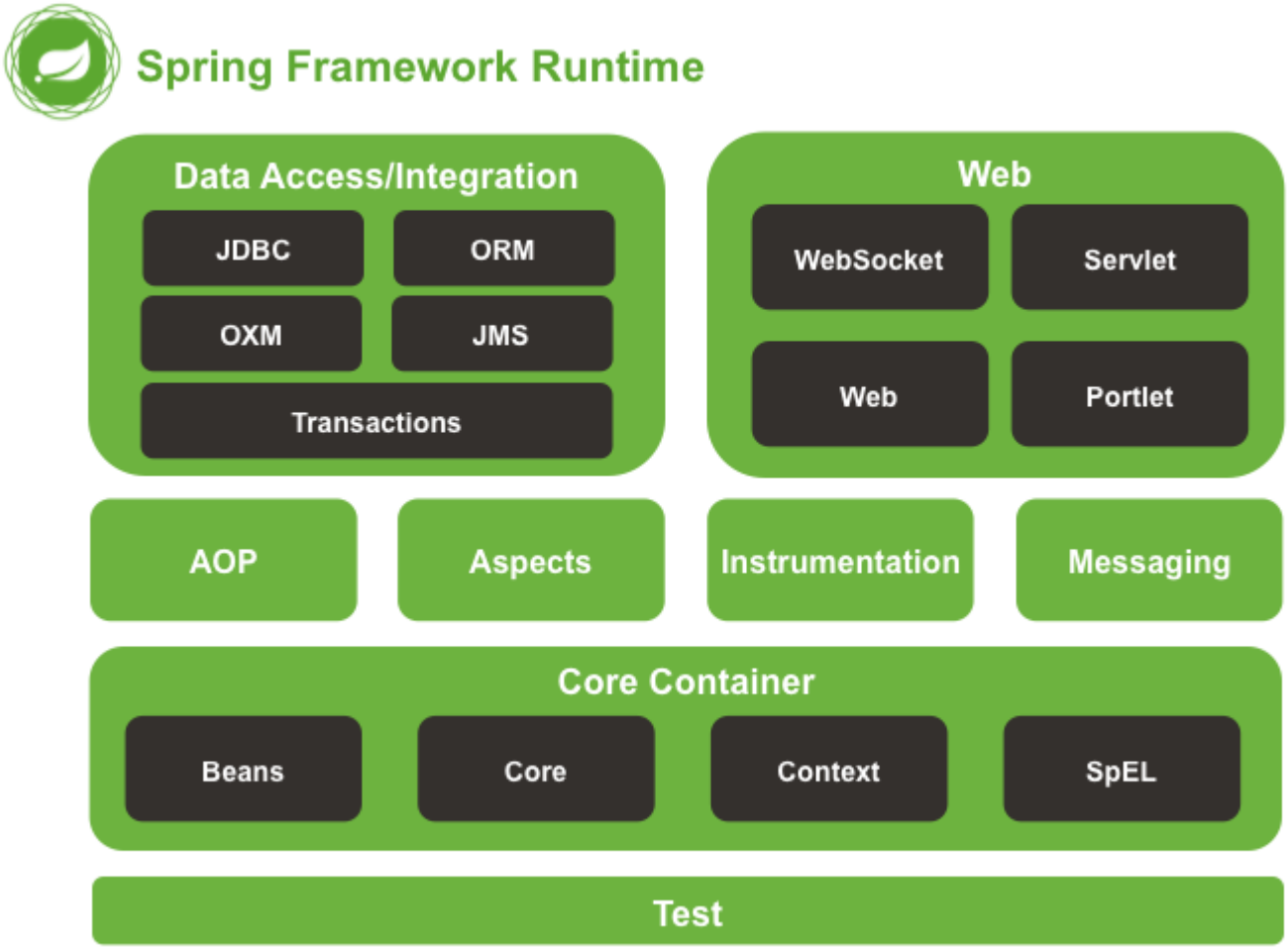
```
@RestController
public class ErrorController {

@RequestMapping(path = "/error")
public Map<String, Object> handle(HttpServletRequest request) {
  Map<String, Object> map = new HashMap<String, Object>();
  map.put("status",
      request.getAttribute("javax.servlet.error.status_code"));
  map.put("reason",
      request.getAttribute("javax.servlet.error.message"));
  return map;
}}
```


Source: https://docs.spring.io/spring-framework/docs/

Spring MVC Configuration

```
@EnableWebMvc
@Configuration
@ComponentScan("org.iproduct.spring.webmvc.web")
public class SpringWebConfig implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(
            ResourceHandlerRegistry registry) {
        registry.addResourceHandler("/resources/**")
            .addResourceLocations("/resources/");
    }
    @Override
    public void addViewControllers(
            ViewControllerRegistry registry) {
        registry.addViewController("/new-article")
                .setViewName("articleForm");
        registry.addViewController("/home").setViewName("home");
    }
    ...
```

Spring Framework 4.2 Main Modules



Source: https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/overview.html

Additinal Examples

Learning Spring 5 book examples are available @ GitHub:

https://github.com/PacktPublishing/Learning-Spring-5.0

Spring 5 Core Referenc Documentation:

https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html

Spring Expression Language Guide (Baeldung):

http://www.baeldung.com/spring-expression-language

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg