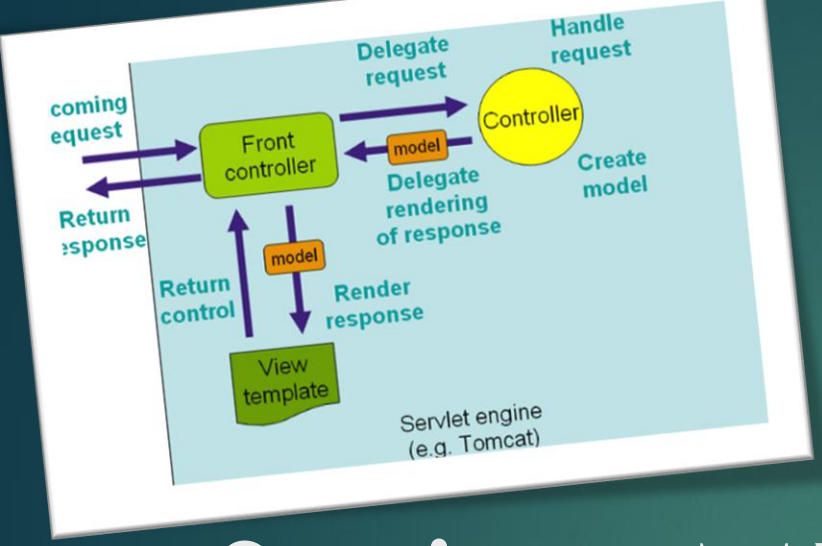


Spring MVC Notes

USEFUL FOR REVISION

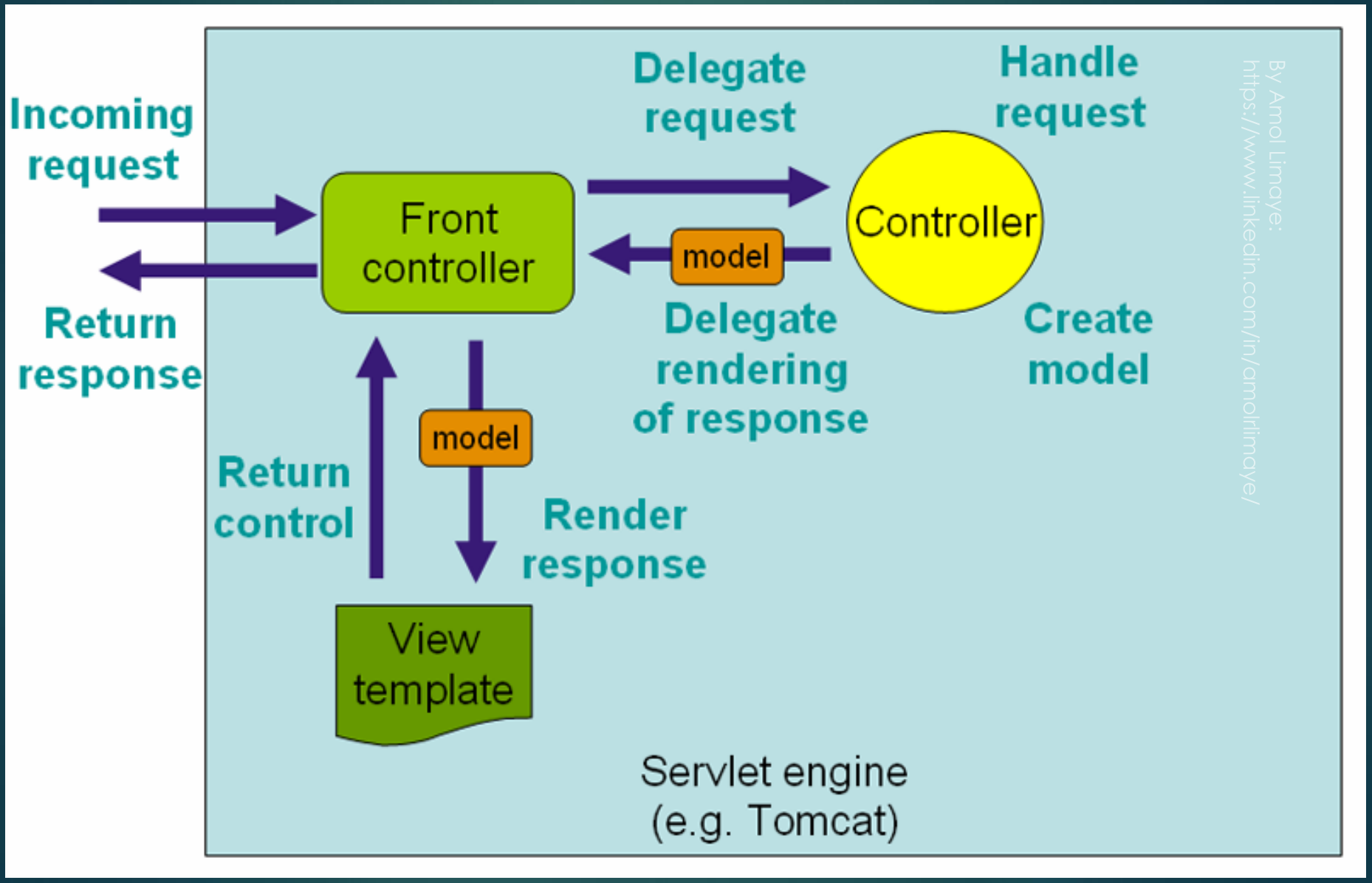
Index

By Amol Limaye:
https://www.linkedin.com/in/amolrlimaye/

What is Spring MVC

- Spring MVC is the Spring module to build web applications –> MVC = Model-View-Controller
- It uses 'Front Controller Pattern' , where a central Servlet – **DispatcherServlet** - provides the algorithm to process the requests.
- Actual work of request processing is performed by configurable delegate components

Request Flow



M - The Model

- Model represents the holder or container for data. This is used to pass values to render the view.
- In Spring Web MVC you can use any object as Model; you do not need to implement a framework-specific interface or base class
- The Model is a 'Map' which supports complete abstraction of view technology
- Model is also known as the Command or Form-backing object

V – The View

By Amol Limaye:
https://www.linkedin.com/in/amolrlimaye/

- A View is generally a templating engine to render the webpage that displays data returned by controller
- For View layer, many template based rendering technologies are supported like Velocity, JSP, Freemarker etc.
- The output can also be generated directly in the form of XML, JSON, Atom etc.
- The Model 'Map' is converted by Controller into the appropriate format of the View, like JSP attributes or Velocity template model.

C – The Controller

- A controller accepts user inputs and is typically responsible to prepare the Model and select a View to forward that model to
- Controller can also directly write to the response stream and complete the request – like JSON, XML or a filestream
- A controller can be created using the @Controller stereotype annotation

Important Annotations

- **@Controller** - This stereotype bean annotation indicates that a given class instance is a controller bean
- **@ResponseBody** – When a method is annotated with this, the return type is written to the response body. Spring converts returned object to the response body using *HttpMessageConverter*
- **@RequestBody** – This indicates that a method parameter should be bound to the HTTP request body. Again, *HttpMessageConverter* converts the request body to the object of specified type.

Important Annotations

- **@RestController** – This stereotype annotation combines **@Controller** and **@ResponseBody**
- **@RequestMapping(value="/url", method=RequestMethod.GET)** – This specifies the URL to be handled by controller or method. The 'method' part of the annotation specifies which type of HTTP method will be handled
- **@GetMapping** – This is a *composed annotation* which acts as a shortcut for *@RequestMapping (method = RequestMethod.GET)* . This is available since Spring 4.3 . Similar shortcut annotations are available for other HTTP methods like POST, PUT, DELETE etc.

Important Annotations

- **@PathVariable** – You can use this annotation on method argument to bind it to a value of URI template variable.

Example:

```
@GetMapping("/owners/{ownerId}")
public String findOwner(@PathVariable String ownerId)
{
```

- **@RequestParam** – Use this to bind request parameter to method parameter in your controller

Example:

```
@GetMapping
public String setupForm(@RequestParam("petId") int
petId) {
```

Important Annotations

- **Consumable media types** (**consumes**) – You can narrow the controller method primary mapping by listing the consumable media types. The request will be matched only if *Content-Type* header of the request matches the given type.

```
@PostMapping(path = "/pets", consumes = "application/json")
public void addPet(@RequestBody Pet pet, Model model) {
```

- **Producible media types (produces)** – Similar to above, the request will be matched only if 'Accept' request header matches the given type

```
@GetMapping(path = "/pets/{petId}", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
public Pet getPet(@PathVariable petId, Model model){
```

Important Annotations

- **@ExceptionHandler** -- Use this annotation on methods to be invoked to handle an exception. This method can be defined locally in a @Controller or would apply to many @Controller when defined in @ControllerAdvice annotated class
- **@ControllerAdvice** – Methods defined in class annotated with this will apply to multiple controllers. This is a specialization of @Component
- **@RestControllerAdvice** = @ControllerAdvice + @ResponseBody

Using Themes

- Spring MVC gives ability to change the look and feel of your application – example, offer personalized views
- It can be typically done by supplying custom stylesheets and images in a folder named 'themes' in resources
- 'themeSource' and 'themeResolver' are the bean names which DispatcherServlet looks for to detect any themes related code
- **themeSource** – specifies the source of theme, like resources folder or any other custom location
- **themeResolver** – specifies how to detect theme for a particular request

Supported View technologies

- Thymeleaf
- JSP
- JSTL
- Velocity (Deprecated as of Spring 4.3)
- Freemarker
- Groovy markup
- Script views (e.g. React running on Nashorn)
- XML, XSLT
- Document views (PDF, Excel)
- Feed Views (RSS, Atom)
- JSON, XML and many more..

Spring MVC Test framework

- Based on 'Servlet API mock objects' and do not use a running servlet container
- Uses *DispatcherSerlet* to provide full Spring MVC runtime behavior
- Provides support to load actual Spring configuration during tests using *TestContext framework*
- MockMvc is an important component provided by this module for testing your Spring MVC code

By Amol Limaye:
https://www.linkedin.com/in/amolrlimaye/

Important beans and interfaces

- **ViewResolver** – Provides mapping between view names and actual views
- **View** - Prepares the request and hands it to proper view technology
- **LocaleResolver** – Resolve client's locale and possibly show internationalized views
- **ThemeResolver** – Resolve themes your application can use
- **HandlerExceptionResolver** – Deals with unexpected exceptions occurring during controller execution
- **MultipartResolver** – Supports multipart request, e.g. file upload from HTML form

Save this PDF for quick reference

- Source: Official Spring documentation
- Follow Amol Limaye to more see such content in your feed

https://www.linkedin.com/in/amolrlimaye/