Java Backend Development Live-85

lecture-9

Agenda

- Spring Boot Auto-Configuration
- Request Param, Path Params, Request Body
- Spring Logging and Request Tracing
- HttpFilter
- Spring MVC Architecture
- Controller

Spring Boot Auto-Configuration

- A feature that automatically configures Spring applications based on the libraries present on the classpath.
- Reduces the need for explicit configurations in `application.properties` or Java-based configurations.
- Enabled by default with the `@SpringBootApplication` annotation, which includes `@EnableAutoConfiguration`.
- **Classpath Detection:**
  a. Detects libraries (e.g., Hibernate, Thymeleaf, Kafka) on the classpath.
- **Conditional Beans:**
  a. Uses `@Conditional` annotations to configure only required beans.
  b. Examples: `ConditionalOnProperty` `@ConditionalOnClass`.

Sending Data in Request

- Request Param/ Query Param
- Path Params,
- Request Body
- Request Header

Spring Logging

2022-02-07 18:59:41.257 INFO 477255 --- [main] c.gfg.springdemo.SpringDemoApplication   : Started SpringDemoApplication in 2.004 seconds (JVM running for 2.489)

Date and Time — Millisecond precision and easily sortable.

Log Level — ERROR, WARN, INFO, DEBUG or TRACE.

Process ID.

A --- separator to distinguish the start of actual log messages.

Thread name — Enclosed in square brackets (may be truncated for console output).

Logger name — This is usually the source class name (often abbreviated).

The log message.

Http Filter

A filter is an object used to intercept the HTTP requests and responses of your application. By using filter, we can perform operations at two instances.

- Before sending the request to the controller
- Before sending a response to the client.

```
public class Requestfilter extends HttpFilter {

        @Override

        public void doFilter(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws
    IOException, ServletException {

            filterChain.doFilter(request,response);

        }

}
```

We can define order of filter with @Order annotation.

Tracing with Logging

We can put a unique value in request header (e.g requestId, transactionId etc) and get this value print with every log. Logger Library provides MDC(Mapped Diagnostic Context) to support this.

Just add value in MDC before processing the http request.

MDC.put(“requestId”, servletRequest.getHeader(“requestId”));

Logging pattern

logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} [**[requestId=%X{requestId}]**] - %msg%n

logging.pattern.file= %d{yyyy-MM-dd HH:mm:ss} [%thread | **[requestId=%X{requestId}]**] %-5level %logger{36} - %msg%

#Logging File

logging.file.name=/tmp/app.log

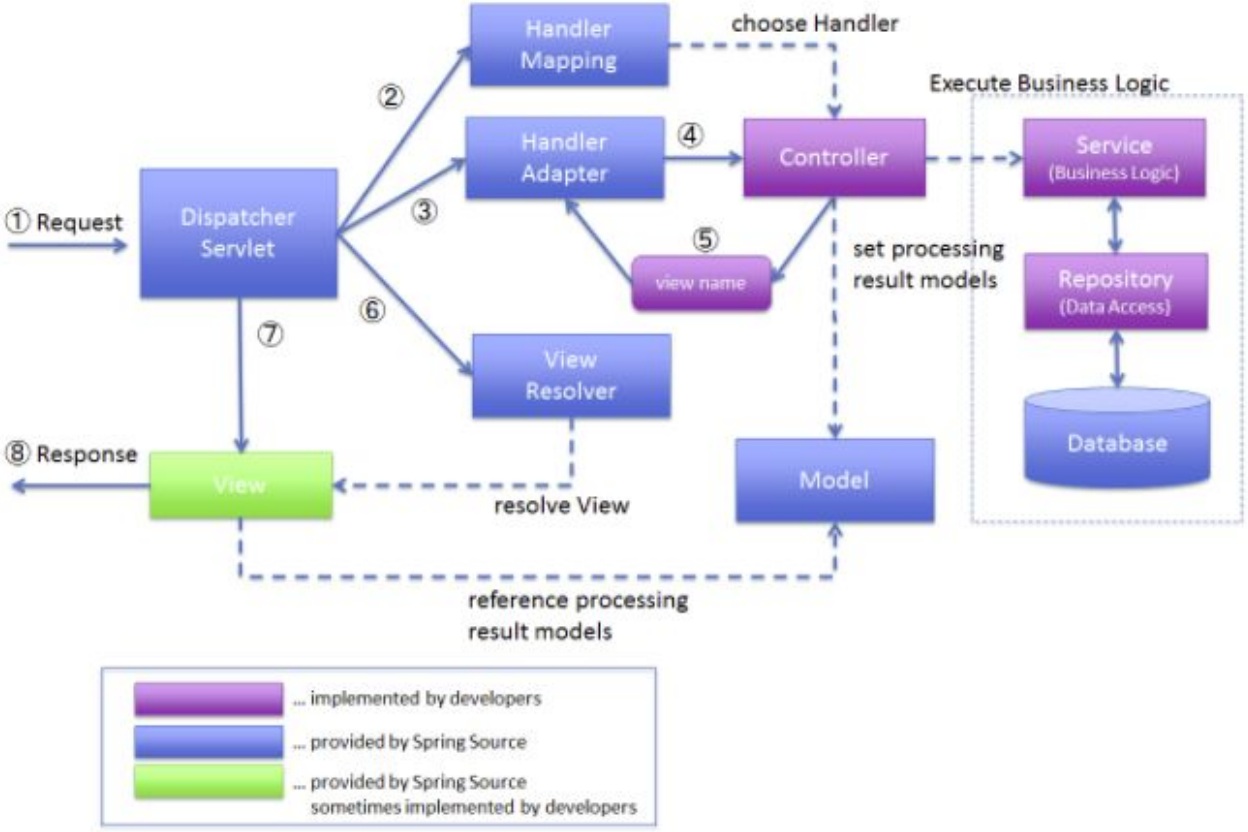
Spring MVC (Model-View-Controller)

- Spring MVC is a web framework that simplifies the development of web application by separating the application into three components.
  - The **Model** encapsulates the application/business data.
  - The **View** is responsible for rendering the model data and it generates HTML output that the client's browser can interpret.
  - The **Controller** is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.
- Spring Boot enhances Spring MVC by providing auto-configuration, embedded servers, and starter dependencies.

Key Concepts of Spring MVC

- **DispatcherServlet**: The central component of Spring MVC. It acts as a front controller that handles all HTTP requests and responses.
- **Handler Mapping**: Maps incoming requests to the appropriate controller methods.
- **Controller**: Contains business logic and processes user requests.
- **View Resolver**: Resolves view names into actual view implementations (e.g., Thymeleaf templates, JSP).

Spring MVC Architecture



Controller vs RestController

@Controller create a Map of Model Object and find a view(jsp,html).

@RestController simply return object and object data directly written into http response as JSON or XML.

@Controller is used to serve HTML, JSP, etc pages. We can pass data as a Model to be filled in these pages (views).

Nowadays frontend code (HTML, js, etc.) is served by different servers (e.g nginx). Backend APIs are used to get data only.