Effective Spring Development

Property files, logging and profiles

Property files

- application.properties is the default properties file in Spring Boot
- Typically it is found in the src/main/resources directory and this is because Spring Boot will look for it there, it will be auto-detected
- The application.properties file can be used to override default configuration
- For example, change the port of a webapp from default 8080 to 8081:
- server.port=8081

Logging

- Logging is an important aspect of any software application
- It helps troubleshooting and debugging issues
- Spring Boot comes pre-configured with logging support
- Spring Boot offers simple configuration options for logging in the application.properties file
- Spring Boot provides support for multiple log levels such as TRACE, DEBUG, INFO, WARN and ERROR

Demo 1 - Logging and properties

- Using the application.properties file
- Logging and logging levels
- Using profiles

Profiles

- Profiles in Spring Boot are a powerful feature that allows developers to configure and manage different environments or deployment scenarios
- With profiles you can easily specify different sets of configuration options for different environments, for example use different databases when running the project in development or production
- The active profile can be set with spring.profiles.active in application.properties or as an environment variable
- The configuration for the profile prod is specified in the alternative property file application-prod.properties

Demo 1 - Profiles

- Setting up a specific profile
- Setting the active profile

Exercise 1 - server.port

- Use the CreateASpringBootAppSolution application (the solution application from the module Creating a Spring Boot Application)

- Change the server port for the web application from the default port 8080 to another port, like 8081 by using this property in application.properties (restart the application and see if it works):

- server.port=8081

Exercise 2 - Logging, part 1

- Setup logging in the Controller class by adding this line in Exercise1Controller (the class in the end is the class the line is in):

- Logger logger = LoggerFactory.getLogger(Exercise1Controller.class);

- Then inside the methods, use the log in different levels, like:

- logger.info(”Info!"); or use another logging level like debug, warn or error

- Set the logging level in application.properties to any logging level, like:

- logging.level.root=info

Exercise 2 - Logging, part 2

- Try to run the application and run a few methods with logging (remember the port could be set to 8081) and you should see logging in the console

- Add this line in application.properties:

- logging.file.name=log.txt

- Restart the application and run a few methods, now you should see logging in the console but also a file called log.txt in the project

Exercise 3 - Profiles

- Now create two profiles, dev and prod, by adding two new properties files next to application.properties in the resources directory. The name of the two properties files should be application-dev.properties and application-prod.properties

- Move the configuration of the port and the logging to these files but make the values them different in each file

- Now select the active profile with this line in application.properties and run the application, also change this and restart the application to see that the configuration really change with the profiles:

- spring.profiles.active=prod