endava

UNIVERSITATEA DIN BUCUREȘTI
VIRTUTE ET SAPIENTIA

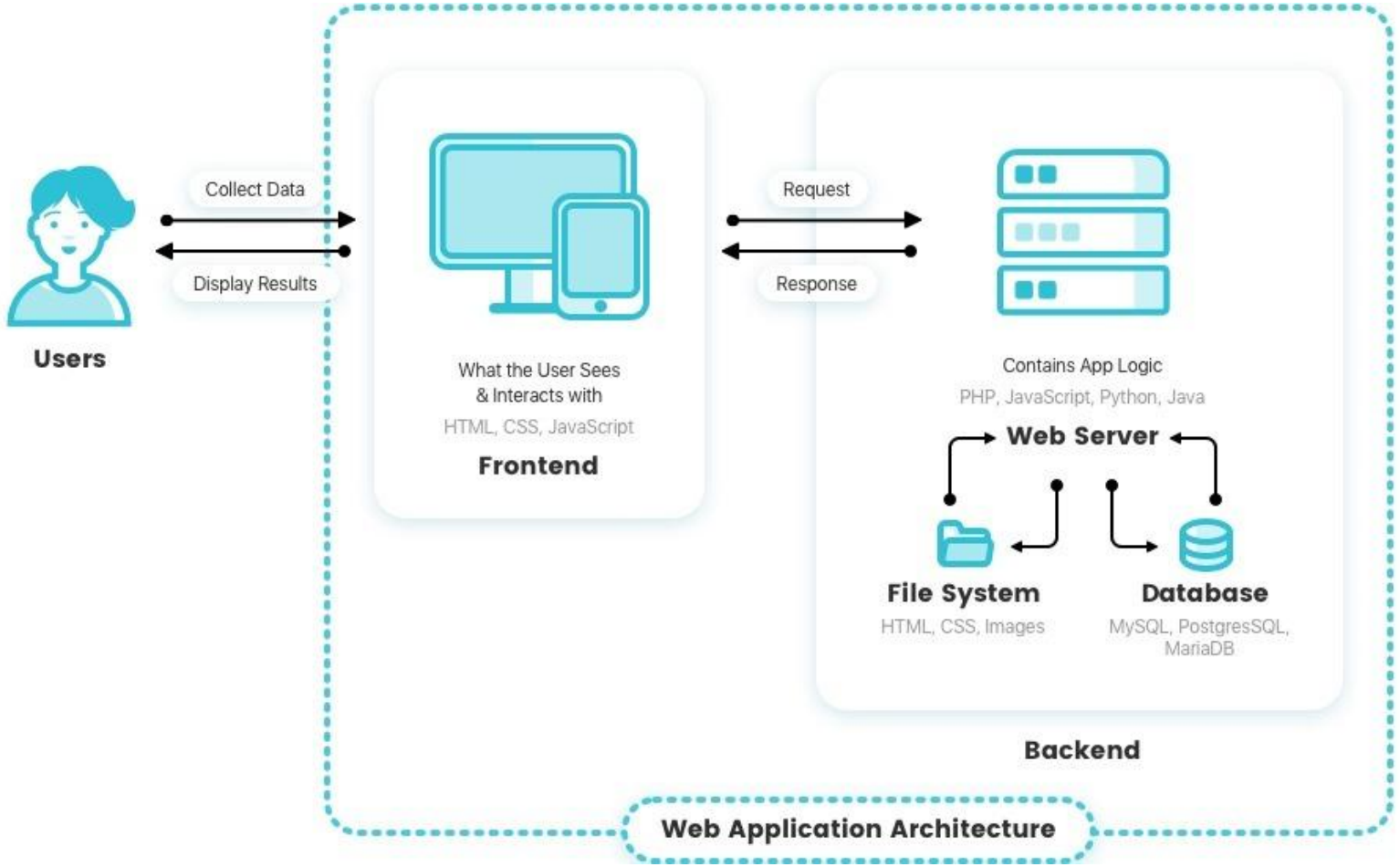
WEB TECHNOLOGIES USING JAVA

COURSE 2 – SPRING CORE.

AGENDA

- BASIC WEB APPLICATION STRUCTURE
- WHAT IS SPRING FRAMEWORK
- MAVEN
- SPRING CONTEXT
- HOW TO PUT A BEAN IN THE CONTEXT
- HOW TO RETRIEVE A BEAN FROM THE CONTEXT

BASIC WEB APPLICATION STRUCTURE



WHAT IS SPRING FRAMEWORK

- application framework: a set of commonly-used software functionalities that provides a foundation for developing an application.
- saves time and helps ensure you have fewer chances of generating bugs.
- technical code is larger than the business logic code
- modular framework
- real world usage:
  - backend apps
  - automation testing frameworks
  - desktop apps
  - mobile apps

WHAT IS SPRING FRAMEWORK

- Objectives of this course using Spring:
  - use the Spring context and understands aspects (Spring Core)
  - implement data exchange between apps using REST APIs (Spring Web)
  - implement the mechanism of a Spring app to connect to a database and work with the persisted data (Spring Data JDBC and Spring data JPA)
  - build basic apps that use the convention over configuration approach (Spring Boot)
  - testing Spring apps
- Spring modules we will discuss further:
  - Spring Core
  - Spring AOP
  - Spring Web
  - Spring Boot
  - Spring Data JDBC
  - Spring Data JPA
  - Spring Test

MAVEN

- building tool
- common building tasks:
  - downloading the dependencies needed by your app
  - running tests
  - validating that the syntax follows some rules that you define
  - checking for security vulnerabilities
  - compiling the app
- pom.xml file:
  - dependencies
  - plugins
  - building configurations

SPRING CORE

- Provides the fundamental mechanisms used by Spring to integrate into apps:
  - Spring Context
  - Dependency Injection (DI)

DEPENDENCY INJECTION

- Design principle in Spring Framework that helps manage object dependencies in a clean, decoupled way.
- It’s part of the broader **Inversion of Control (IoC)** pattern: the control of object creation, configuration, and management is handled by the Spring IoC container rather than the application itself.

SPRING CONTEXT

- a place in the memory of your app in which we add all the object instances that we want the framework to manage
- beans: object instances we add to the Spring context
- not all objects in your app need to be managed by Spring
- *AnnotationConfigApplicationContext* class is the most used approach to obtain a reference to the context

HOW TO PUT A BEAN IN THE CONTEXT

- Annotation approach versus xml approach
- Annotation approach:
  - @Bean
  - stereotype annotations
    - @Component
    - @Controller
    - @RestController
    - @Service
    - @Repository
  - programmatically

HOW TO PUT A BEAN IN THE CONTEXT

- Steps for using @Bean:
  1. define a @Configuration class
  2. create a method in this class and annotate it with @Bean. This method should return the instance you want to become a bean
  3. make Spring use the configuration class when creating the context

- Steps for using stereotypes annotations:
  1. mark the class of the bean with a stereotype annotation (for example, @Component)
  2. add @ComponentScan to a @Configuration class, to instruct Spring where to look for stereotype annotations
  3. make Spring use the configuration class when creating the context

HOW TO PUT A BEAN IN THE CONTEXT

Using @Bean	Using stereotypes annotations
You have full control over the creation of the bean	You have full control only after the bean is created by Spring
You can add multiple instances of the same type	You can add only one instance of a type
You can create beans of any classes	You can create beans only of classes defined in your app
You need to define one method for each bean -> a lot of code	You only add an annotation for each bean -> less code

HOW TO RETRIEVE A BEAN FROM THE CONTEXT

- Wiring: directly calling a @Bean annotated method from another @Bean annotated method
- Autowiring: you let Spring provide a value for a parameter
  - by name: enables the dependency injection based on bean names
  - by type: enables the dependency injection based on bean types
- Types of dependency injection:
  - on constructor:
    - most used, production-ready apps
    - useful when dependencies are mandatory, as the object cannot be created without them
  - on setters:
    - less used
    - useful when dependencies are optional or can be changed later
  - on attributes:
    - least preferred, proofs-of-concept and tests
    - less recommended because it’s harder to test and violates principles of encapsulation

HOW TO RETRIEVE A BEAN FROM THE CONTEXT

- Choosing from multiple beans in the context:
  - @Qualifier (explicit choice)
  - @Primary (default choice if multiple beans exist)
  - by Type (single bean match)
  - by Name (match field name to bean name)
  - throws exception (if multiple beans match and no qualifier is provided)

    It's important to resolve this by either:

    - using @Primary to set a default bean,
    - using @Qualifier to specify the exact bean, or
    - refactoring to ensure only one eligible bean exists in the context.

BIBLIOGRAPHY

- Spring in Action, by Craig Walls
- Youtube Spring playlist

THANK YOU!