UCA Universidad de Cádiz

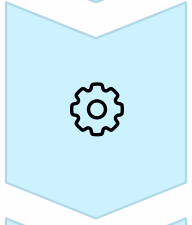
Developing Web Applications with Plain Java

Andrés Muñoz / Iván Ruiz

Contents

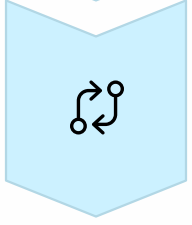
Plain Java Web Development

Introduction to building web apps with Java



Framework & Setup

Vaadin, Spring Boot configuration and project structure



Development & Integration

UI components, routing, data binding and live coding demo

What is "Plain Java" Web Development?

Single Language Development

Build entire web applications without writing HTML, CSS or JavaScript

Full-Stack Java

Develop both frontend and backend using only Java

Clean Architecture

Maintain proper separation of concerns with type-safe code

Why Vaadin?

Component-Based

Create UIs using reusable, high-level Java components that render as standard Web Components.

Enterprise Ready

A production-grade framework with built-in features for routing, security, and more, ready for enterprise applications.

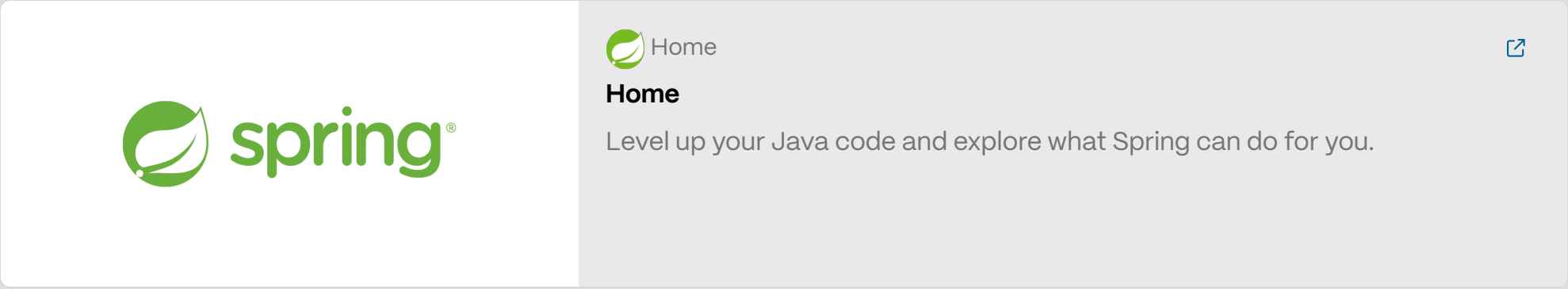
Automatic Data Binding and Validation

Streamline development with built-in data binding and validation, ensuring data integrity and reducing boilerplate.

Active Community & Support

Leverage a large, active community, extensive documentation, and professional support for quick problem resolution and continuous learning.

Why Spring Boot?



Streamlined Configuration

Auto-configuration significantly reduces boilerplate code, allowing you to focus on business logic rather than tedious infrastructure setup.

Comprehensive Module Ecosystem

Access a rich collection of modules for Data Access, Security, Web Services, Transactions, Caching, Data Integration, and more, providing robust support for diverse application needs.

Production-Ready Features

Built-in features like metrics, health checks, and externalized configuration simplify the development and deployment of enterprise-grade applications.

Spring Boot seamlessly complements Vaadin by providing a robust backend foundation, while Vaadin handles the UI layer, creating a complete and streamlined Java-only web development experience.

Why Maven?

Dependency Management

Automatically resolves and downloads required libraries and their dependencies, eliminating manual JAR file management and version conflicts.

Build Standardisation

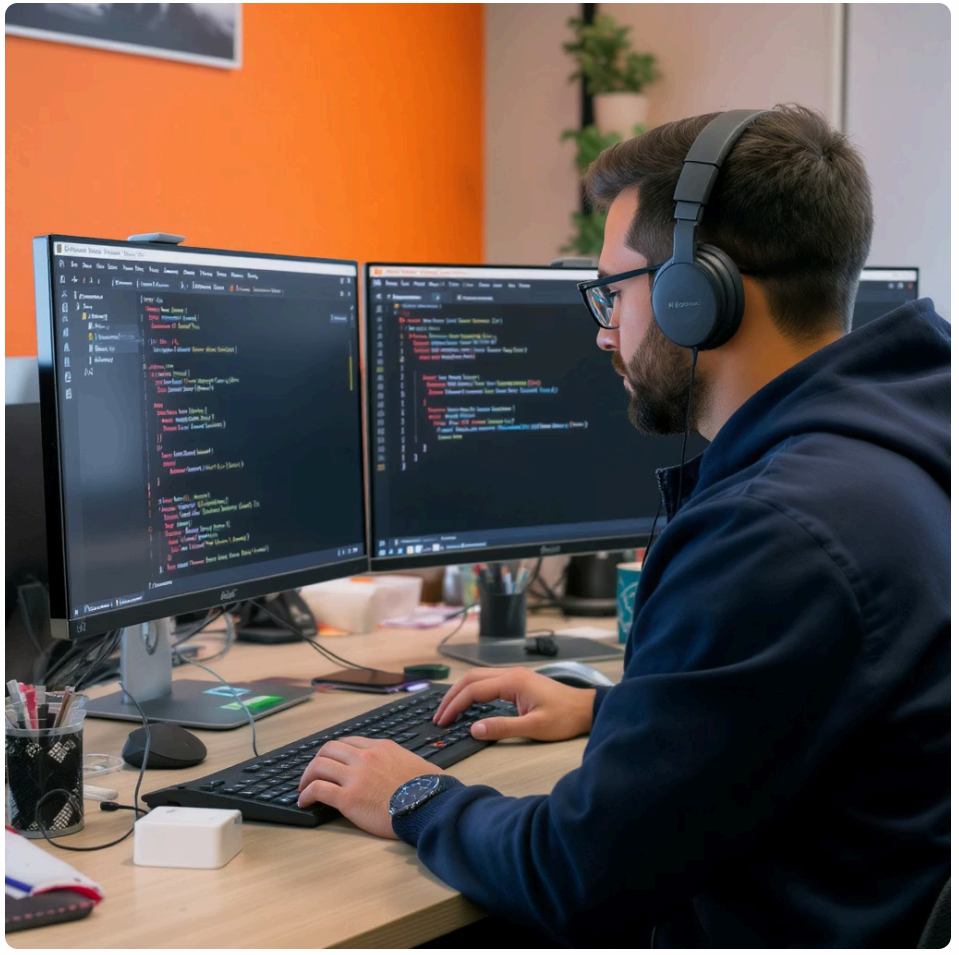
Enforces consistent project structure and build lifecycle across all Java projects, making it easier for developers to switch between projects.

Maven simplifies the build process in Plain Java web development

Vaadin Integration

Seamlessly integrates with Vaadin and Spring Boot, providing plugin support for development workflows, testing, and production builds.

Project Setup



Required Tools

- Java 17 or newer (LTS version)
- **Maven** or Gradle build system
- Spring Initializr or **Vaadin Starter** for project scaffolding
- **IntelliJ IDEA**, VS Code, Eclipse IDE...

Your First Vaadin View

```
@Route("")
public class MainView extends VerticalLayout {
    public MainView() {
        add(new H1("Welcome to Vaadin!"));
        add(new Button("Click me", e ->
            Notification.show("Clicked!")));
    }
}
```

This minimal view demonstrates:

- @Route annotation mapping to root URL
- Layouts for component organisation
- Event handling with Java lambdas

Backend Integration

Vaadin Views

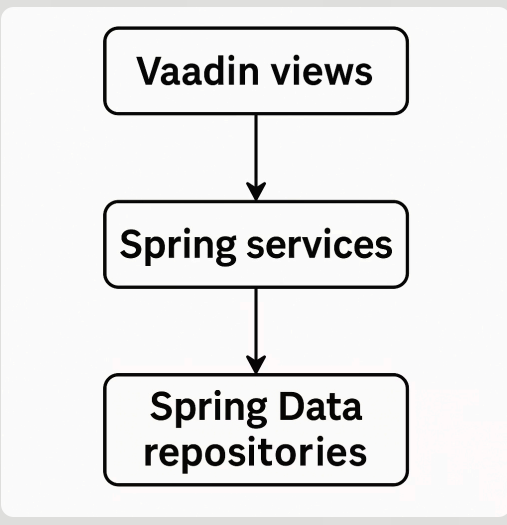
Inject services into views with standard Spring dependency injection

```
@Route("users")
public class RegisterView extends VerticalLayout {

    @Autowired
    private UserService service;

    public RegisterView() {
        TextField username = new TextField("Username");
        PasswordField password= new PasswordField("Password");
        Button register = new Button("Register", e -> {
            if (!username.isEmpty() && !password.isEmpty()) {
                service.registerNewUser(username.getValue(), password.getValue());
                Notification.show("User registered");
            } else {
                Notification.show("Fields cannot be empty");
            }
        });

        add(username, password, register);
    }
}
```



Service Layer

Separate business logic from views with dedicated service classes

```
@Service
public class UserService {
    @Autowired
    private UserRepository repo;

    public User registerNewUser(String username, String pass){
      // TO DO ...
      return repo.save(newUser(username,encodePassword(pass)));
  }
```

Spring Repositories

```
@Repository
public interface UserRepository extends JpaRepository {
   // save, findById, findAll, delete, etc.
}
```

Leverage Spring Data's powerful repository abstractions for database operations

Vaadin Starter

start.vaadin.com

Create a new Vaadin app: configure views and customize the theme

A tool that allows you to visually create a custom Spring Boot based Vaadin Flow or Hilla app starter that you can download and open in your IDE.

Project Structure

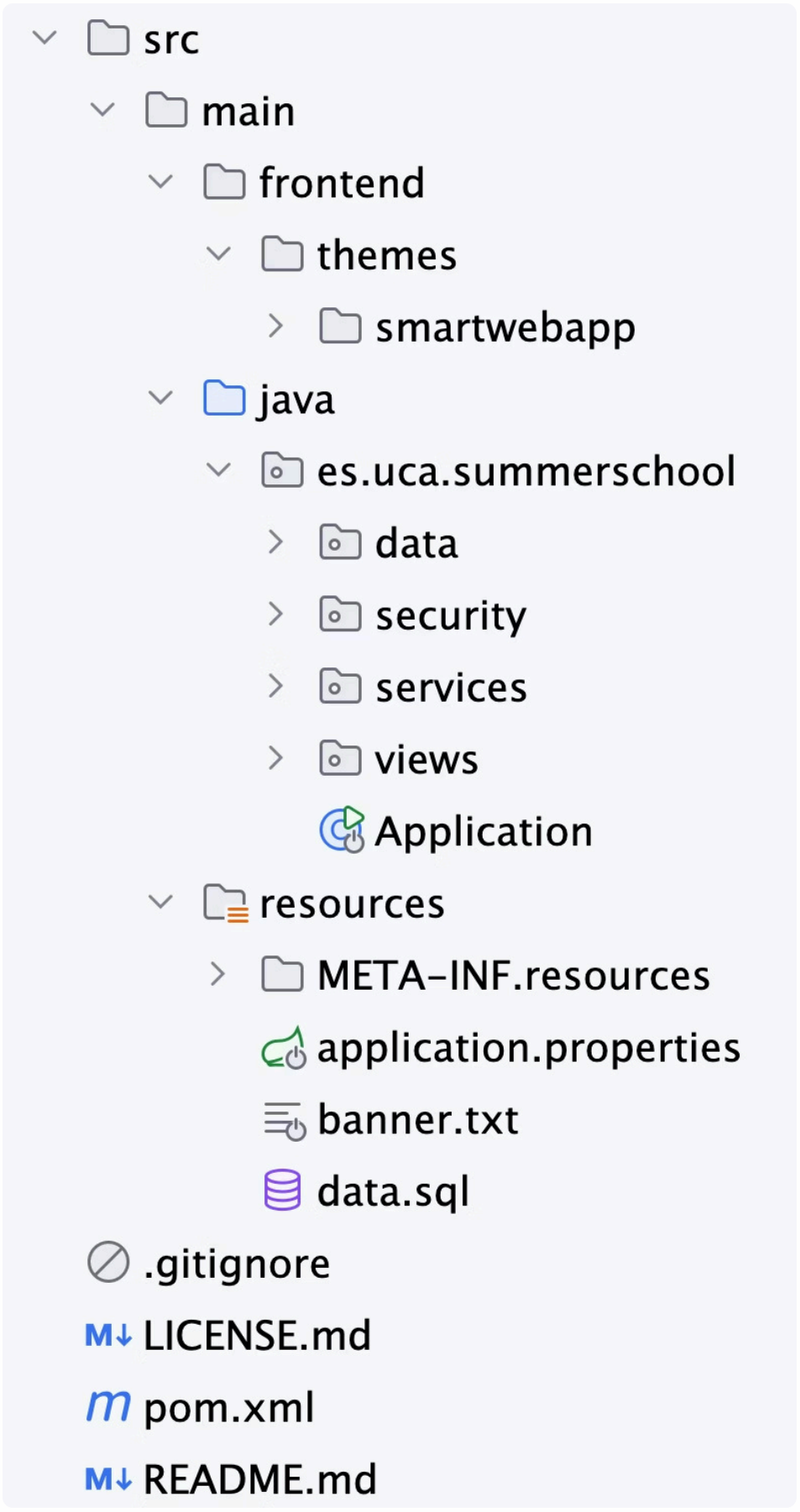
The structure follows Maven conventions with clear separation between application code and resources.

The `frontend` folder contains styles for the visual theme.

The `java` folder contains all Java packages and classes, organized according to their responsibilities (data, security, services, and views).

The `Application` class is the main entry point for the application.

The `resources` folder contains commonly used configuration files.



Deploying the Application

Simple Deployment Options

Vaadin applications are packaged as standard Java web applications, making deployment straightforward.

Build the Application

Package your application using Maven or Gradle

```
mvn clean package
```

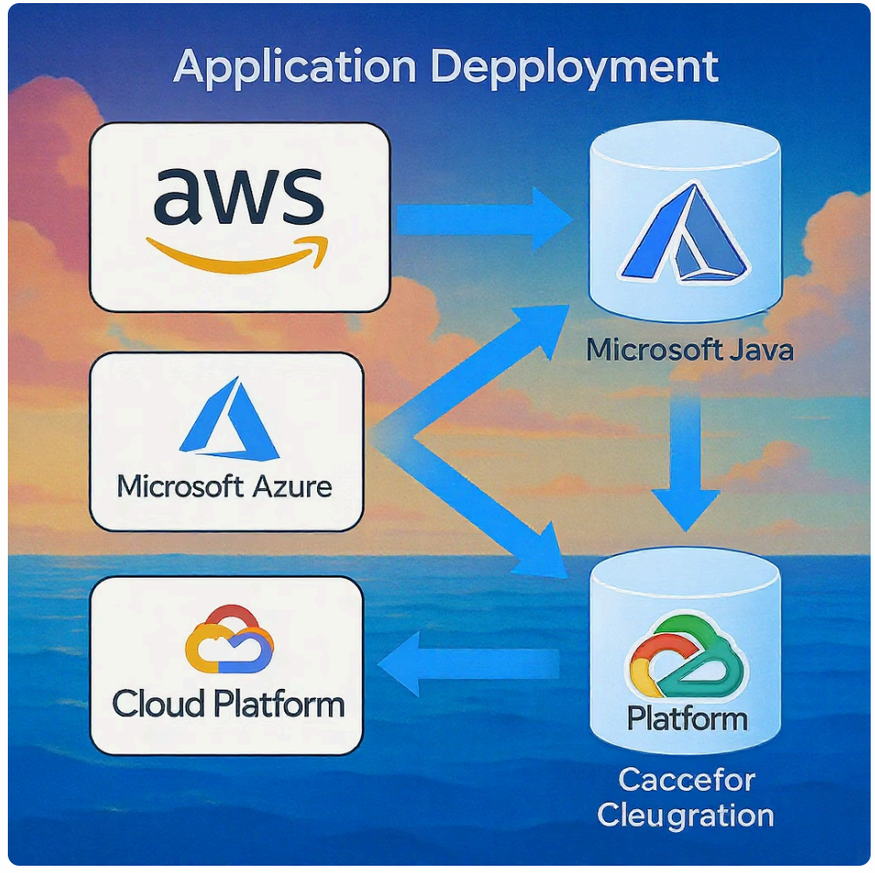
Run the JAR

Execute the self-contained JAR with embedded Tomcat

```
java -jar target/myapp-1.0.jar
```

Cloud Deployment

Deploy to Heroku, AWS, Azure, or Google Cloud



Ex 1. Creating a webapp

Pros & Cons of Plain Java UIs

Pros

- Type-safe development with compile-time checks
- Refactoring support across entire codebase
- Reuse backend skills and knowledge
- Single language throughout the stack
- Integrated security model

Cons

- Less direct control over HTML markup
- Heavier runtime than pure JavaScript frameworks
- Learning curve for component model
- More server resources required

Summary

- Vaadin enables building modern, responsive UIs using pure Java without writing HTML, CSS, or JavaScript
- Spring Boot provides a robust foundation for backend services, security, and data access
- The combined stack offers type-safe, full-stack development with a single language
- Ideal for enterprise applications, internal tools, admin panels, and business applications

Resources & Q&A

Essential Resources

- Official Documentation: https://vaadin.com/docs
- Starter App Generator: https://start.vaadin.com
- Spring Framework: https://spring.io
- Component Directory: vaadin.com/components
- Community Forum: vaadin.com/forum

Any Questions?

Let's review key concepts or jump back into the code examples for clarification.

Start Your Project | Read Documentation