Staff & Salary Neo4j Project

Educational Java Project for Teaching Architecture, Factories, and Graph Databases

Prepared for Students (Group 8 PPR Tutorial): Prepared by Solomon

November 25, 2025

Contents

1 Introduction

This project is a minimal but realistic Java application designed to teach students:

- how to structure a Java project cleanly,
- how to use a factory pattern to create domain objects,
- how to interact with Neo4j using nodes and relationships,
- how to separate model, persistence, factory, and REST layers,
- how to build a small web interface using Javalin, FreeMarker, and jQuery.

The domain is intentionally simple: employees (Staff) with associated salary information (Salary). Data is persisted in a Neo4j graph.

2 Project Overview

The project models:

- **Staff** :an employee with ID, first name, last name, nickname.
- **Salary** : salary amount and currency.

Each staff node in Neo4j has:

- properties (id, first name, last name, nickname)
- an outgoing relationship `HAS_SALARY` to a Salary node

Students learn:

- graph modeling,
- node creation,
- relationships,
- REST endpoints,
- template rendering (FTL),
- AJAX requests (jQuery).

3 Package Structure

The real project contains these core packages:

- **database** :manages Neo4j connection
- **interfaces**: Staff and Salary domain interfaces
- **implementations** : Neo4j-based concrete implementations
- **factory** : StaffFactory and SalaryFactory (factory pattern)
- **rest**: REST routes using Javalin
- **templates** : FreeMarker templates includes jQuery(it can be separate .js file created)

This structure demonstrates clean separation of concerns.

4 The `database` Package

4.1 Neo4jConnection

This class wraps the embedded or file-based Neo4j database.

Responsibilities:

- open the database
- expose a `GraphDatabaseService`
- ensure safe shutdown

Important Concept for Students: A single connection object is passed into factories so they can create nodes and relationships.

5 The `model` Package

Contains the domain interfaces:

5.1 Staff

- `String getId()`
- `String getFirstName()`
- `String getLastName()`
- `String getNickname()`
- `Salary getSalary()`

5.2 Salary

- `double getSalary()`
- `String getCurrency()`

6 The `impl` Package

Concrete Neo4j-backed implementations.

6.1 StaffNeo4jImpl

Represents a single Staff node in Neo4j.

Responsibilities:

- wrap a Neo4j `Node`
- provide getters for staff properties
- resolve linked Salary node

6.2 SalaryNeo4jImpl

Represents a Salary node in Neo4j.

Responsibilities:

- wrap Neo4j node
- return salary properties

7 The `factory` Package

This is the architectural highlight of the project.

7.1 StaffFactory

- takes a `Neo4jConnection` in the constructor
- can create new staff nodes
- can attach salary to staff
- can search by ID
- returns `StaffNeo4jImpl` objects

Key teaching point: Students learn dependency injection and factory responsibility.

7.2 SalaryFactory

Symmetric to StaffFactory, responsible for creating salary nodes.

8 REST Layer (Javalin)

Routes include:

- `GET /staff` : renders staff table (FTL)
- `GET /staff/{id}` :renders single staff (FTL)
- `GET /api/staff` : returns JSON list of all staff
- `GET /api/staff/{id}` : returns single staff as JSON

9 Frontend: FreeMarker + jQuery

9.1 FreeMarker Template (`staff.ftl`)

Contains:

- staff table
- search bar
- reset button

9.2 jQuery Search

Example:

```
$.get("/api/staff/" + id, function(data) {
    // replace table content
});
```

Students learn:

- asynchronous requests
- updating HTML dynamically

10 Learning Goals

By completing this project, students learn:

10.1 Backend Concepts

- What a model is
- What a factory is
- Clean separation of layers
- How graph databases work
- Relationship creation and traversal

10.2 Frontend Concepts

- Templating with FreeMarker
- AJAX using jQuery
- Rendering dynamic content

10.3 Architectural Skills

- correct use of packages
- avoiding mixing logic and UI
- writing clean, testable code

11 Conclusion

The project provides a complete mini-architecture using:

- Java + Neo4j
- Factory Pattern
- Javalin REST API
- FreeMarker frontend
- AJAX search (jQuery)