

Spring Framework (Part 1)

Gold - Chapter 4 - Topic 1

Selamat datang di **Chapter 4 Topic 1**
online course **Back End Java** dari
Binar Academy!



Selamat datang di level Gold! ✨

Pada level silver sebelumnya, kita udah belajar dasar-dasar dari BackEnd Javascript. Pada level gold **chapter 4 ini kita lanjut ke cara menerapkan Spring Framework dalam membuat ERD dan CRUD data.**

Nah, berhubung banyak materi praktikal, kita bakal kenalan dulu tentang konsep si **Spring Framework** pada topik 1. Gimana? kamu udah siap?



Dari sesi ini, kita bakal bahas hal-hal berikut:

- Pengantar Framework
- Jenis Framework
- Cara membuat Hello World dengan Spring MVC
- Cara membuat Hello World dengan Spring Boot
- Konsep Project Structure
- MVC Design Pattern
- Application Server



Spring framework, disini ada kata framework yang udah sering banget kita denger di pembahasan kita.

Sebelum masuk lebih detail, gimana kalau kita jabarin dulu makna dari framework secara terpisah?

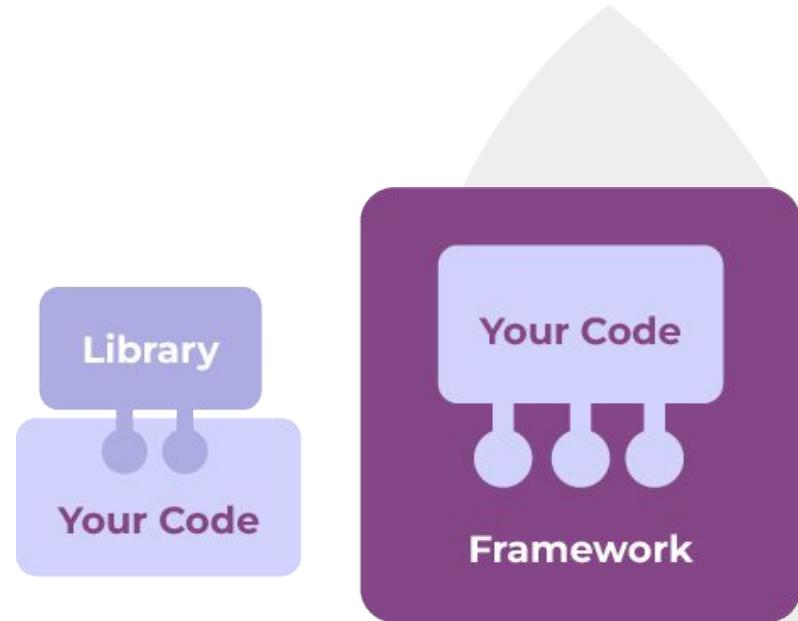


“So, Framework itu apa, sih?”

Secara singkat, framework adalah kerangka kerja.

Kalau kita ngomongin definisi ini dalam software engineering, framework berarti sebuah **template atau kerangka kerja yang terdiri dari berbagai library**.

Dengan kata lain, framework merupakan sebuah kerangka dasar dalam membuat suatu program.



Biar ada gambaran lebih jauh, kita pakai contoh, ya!

Misalnya nih kamu abis nonton video dekorasi meja belajar dan pingin bikin meja belajar baru. To make it happens kamu ada dua pilihan opsi, yaitu:

1. Kamu beli meja belajar rakitan di toko furniture. Nanti tinggal kamu rakit sendiri.
2. Kamu beli bahan-bahannya sendiri, kayak kayu, paku, dan alat pertukangan. Di mana kamu benar-benar buat meja belajar dari awal.



- Kalau kamu ambil opsi pertama, berarti kamu menggunakan framework.

Jadi, kamu nggak perlu lagi bikin dari awal karena kamu udah pakai yang udah jadi.



- Kalau kamu ambil opsi kedua, berarti kamu menggunakan library.

Bahan untuk membuat meja kayak kayu dan paku adalah library-nya.

Waktu library-library ini dikumpulkan dan membentuk suatu fungsi dasar, kumpulan library ini disebut dengan framework.

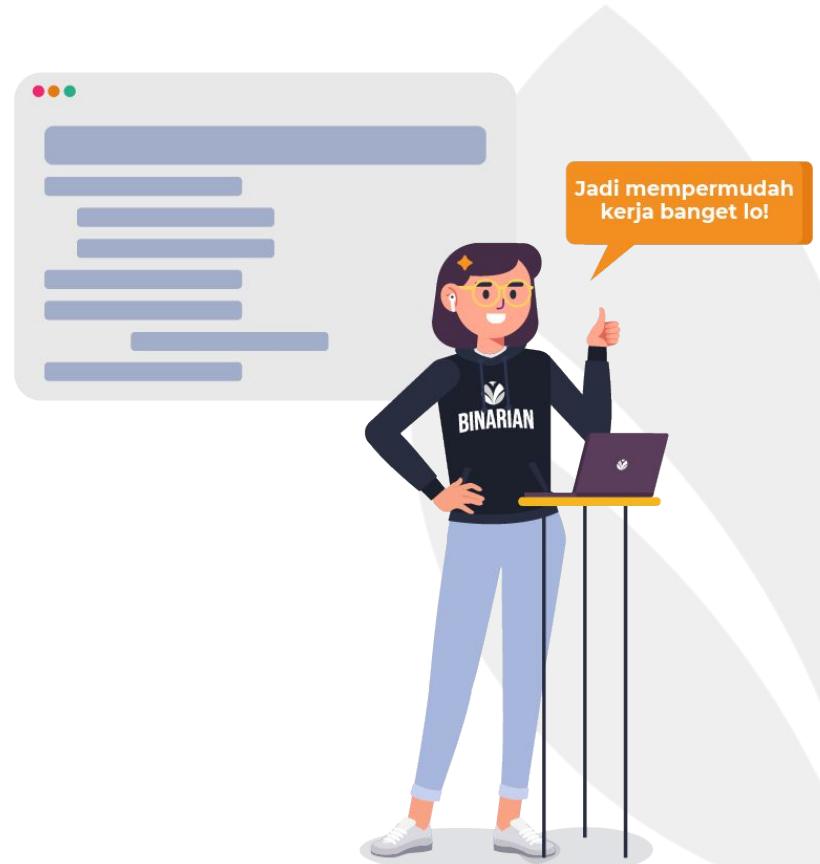


Dari analogi tadi, kesimpulannya kayak gini nih, gengs~

Kalau pakai framework, berarti kita memakai solusi yang pernah dibuat sebelumnya.

Framework ini biasanya memang **dirancang untuk menjadi komponen dasar dari pembuatan berbagai jenis program**.

Dari situ, kita bisa simpulin. Menggunakan framework, berarti secara langsung kita bisa membuat suatu program desktop, web ataupun system back end.

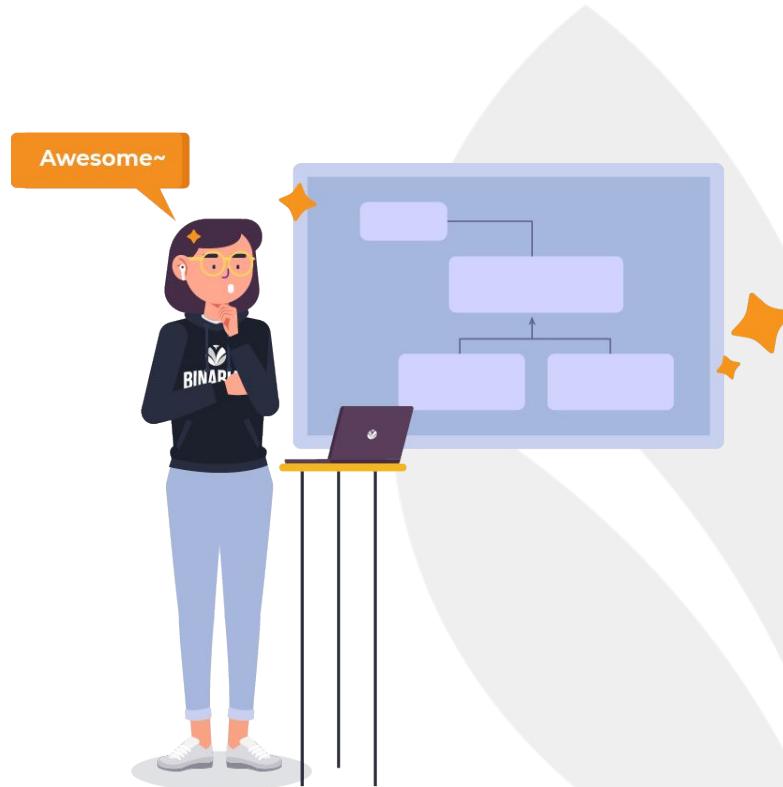


Framework ini ternyata menguntungkan banget buat kita lho~

Kalau kita pakai framework, **proses development bakal jadi lebih singkat** karena kita nggak perlu lagi bikin code dari nol.

Selain itu, kita juga bisa melakukan standarisasi ketika mulai mengembangkan program lain.

Ibaratnya nih, nantinya kita sebagai developer cuma tinggal fokus di business logic aja~



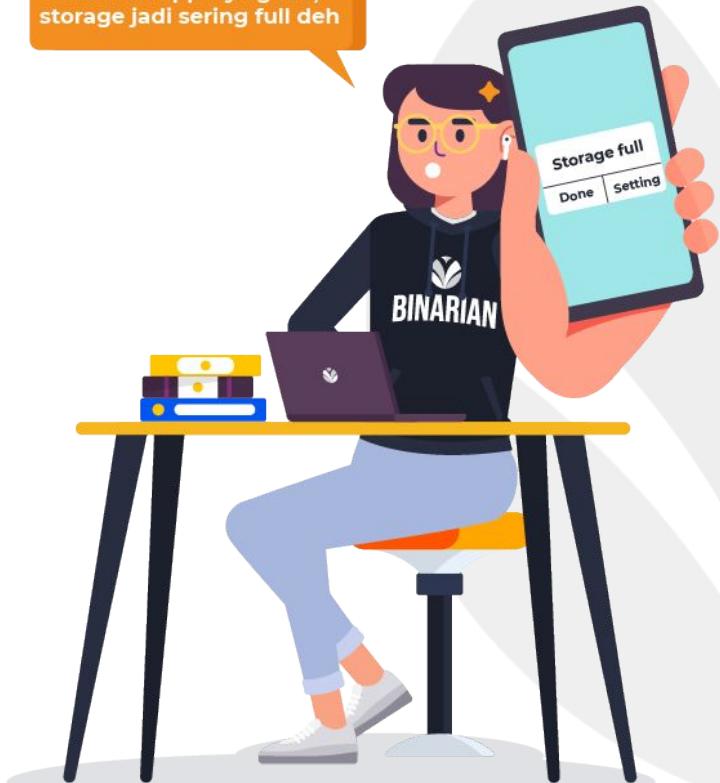
Eits, meskipun bisa mempermudah pekerjaan kita, tapi framework juga punya kekurangan nih, gengs~

Yepp, true. Berhubung framework ini dirancang untuk menciptakan berbagai macam program, bisa jadi ada library-library yang nggak terpakai.

Library yang nggak terpakai inilah yang bikin memory aplikasi jadi besar.

Selain itu, bisa aja library tersebut ternyata memakai suatu resource yang bisa mempengaruhi performance jadi menurun.

Memori app-nya gede,
storage jadi sering full deh



Berikut kesimpulan tentang kelebihan dan kekurangan dari Framework!

Sekalian kita recall, kita coba rangkum kelebihan dan kekurangan jadi beberapa poin berikut.

Kelebihan:

- Proses development nggak dimulai dari nol.
- Developer bisa lebih fokus di business logic aja.



Kekurangan:

- Ada library yang nggak digunakan.
- Memory aplikasi jadi lebih besar.
- Bisa menurunkan performance.

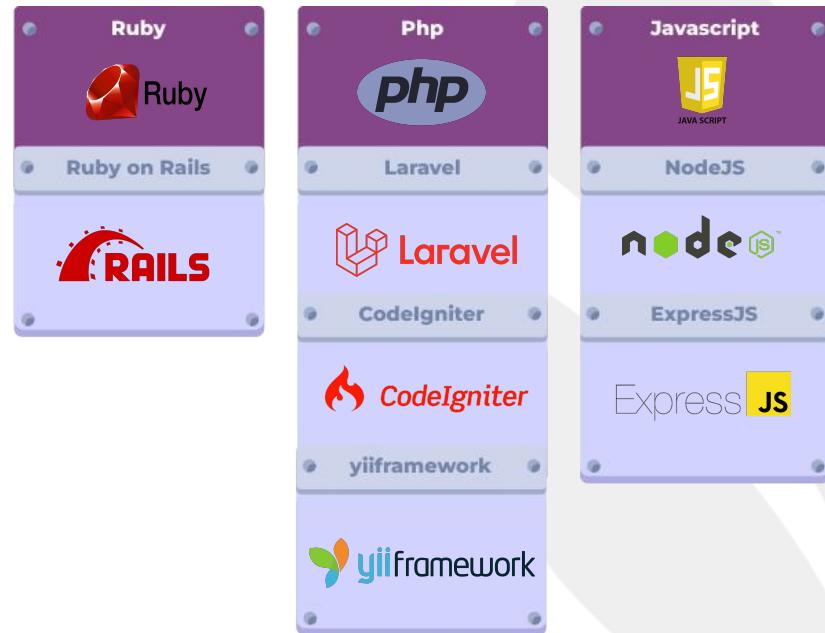


Ada berbagai framework yang bisa digunakan nih, sob!

Setelah memahami konsep framework, siapa yang sangka ternyata framework sangat dekat dengan bahasa pemrograman.

Konsepnya gini, setiap bahasa pemrograman bisa terdiri dari beberapa framework. Misalnya:

- Ruby: Ruby on Rails
- Php: Laravel, Code Igniter, yii
- Javascript: NodeJS, Express JS



“Apakah sebuah program bisa menggunakan lebih dari satu framework?”

Melanjutkan dari penjelasan sebelumnya, jawabannya adalah bisa jadi.

Iya, nggak secara tepat bisa dijawab dengan iya dan tidak. Melainkan bisa jadi.

Bisa jadi sebuah program punya beberapa framework karena jumlah framework ini tergantung sama arsitektur dari program tersebut.



Tadi udah disebutin tuh, kalau satu bahasa pemrograman bisa memiliki banyak framework.

Kalau gitu gimana kalau kita coba pilih satu bahasa yang bakal kita fokusin?

Yepp, pilihan kita jatuh ke Java dan selanjutnya materi ini akan membahas Java Framework.



Ternyata Java punya beberapa framework, gengs!

Yes, ada banyak banget framework yang bisa dipakai. Namun, pada pembelajaran ini kita pilih beberapa framework yang terbaik dipakai. Di antaranya:

- **Spring**
- **Quarkus**
- **Struts**

Yuk kita bahas satu-satu!



Spring

Framework yang paling popular di kalangan Java developer.

Kalau temen kita populer, biasanya karena doi orangnya easy going. Spring juga sama.

Spring sendiri jadi populer karena memiliki dokumentasi yang mudah ditemukan. Selain itu banyak komunitas Spring yang tersedia sebagai wadah buat bertanya para developers.



Framework ini meng-cover banyak sekali fitur, mulai dari:

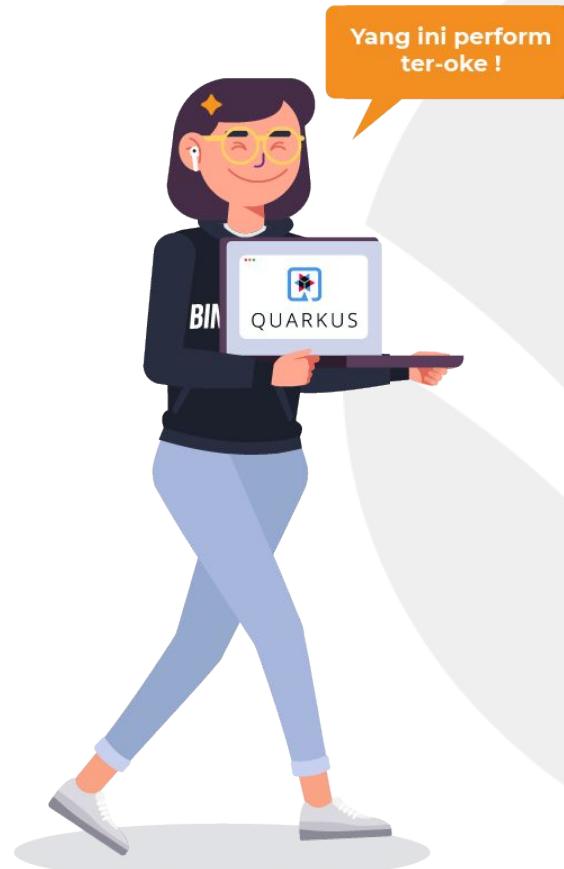
- Fitur untuk menghubungkan aplikasi dengan database.
- Fitur untuk membuat Rest API.
- Fitur untuk membuat tampilan web.
- Sampai dengan fitur untuk membentuk suatu security.



Quarkus

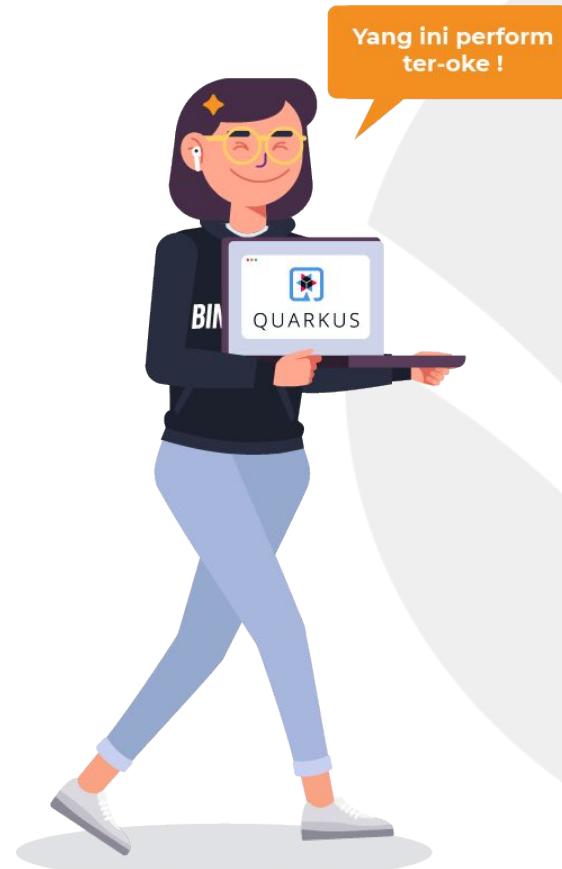
Merupakan framework Java yang dirancang oleh Red Hat pada tahun 2019.

Framework ini **punya performance yang tinggi banget** karena initial releasenya masih belum terlalu lama.



Framework ini belum punya banyak fitur kayak Spring. Bahkan beberapa fitur yang dibuat pun masih ada bug.

Walaupun begitu, **framework ini bisa jadi pilihan untuk membuat program dengan tugas sederhana. Yaitu program yang membutuhkan performance sangat cepat.**

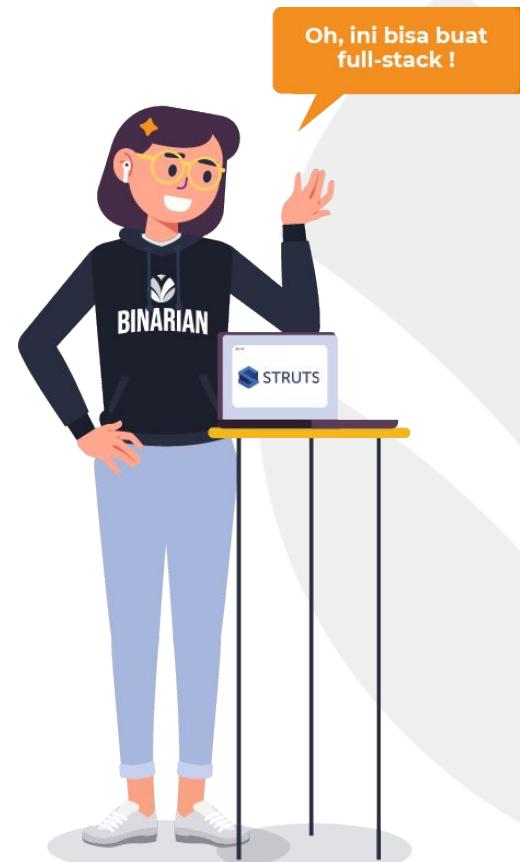


Struts

Framework MVC yang memungkinkan developer untuk **membuat program secara full-stack**.

Framework ini dikembangkan oleh Apache.

Meskipun fitur dari struts ini cukup banyak, tapi Struts nggak se-populer kayak String, jadi dokumentasinya nggak terlalu banyak, deh~



Berikut adalah beberapa framework lainnya~

Framework tersebut, yaitu Play, Vaadin, Spark, Grails, zkoss, dan masih banyak lagi~~

“terus kita bakal pelajari semua nih?”

Kalem sob, di course ini **kita cuma bakal fokus sama framework Spring aja**, yaaaa~



LANJUTTT



Sipp deh! udah mulai tahu kan Spring Framework
itu kayak gimana?

Pada beberapa materi ke depan, kita bakal mulai
detail nih bahas Spring.

Mulai dari Spring Framework, Spring MVC dan
Spring Boot.

Di dalam penggunaan Spring, kita bakal menemukan terminologi **Spring MVC** dan **Spring Boot** lho, gengs!

“Hmm, itu apa, ya?”

Pada dasarnya **Spring MVC** dan **Spring Boot** adalah bagian dari **Spring framework** yang semuanya punya fitur-fitur utama dari Spring.

Selanjutnya, kita bakal bahas satu-satu istilah baru ini~



Yang pertama adalah Spring MVC~

Spring MVC adalah modul dari Spring yang **dipakai untuk membuat program yang bersifat fullstack.**

Hal ini karena konfigurasinya udah diperuntukan buat bikin aplikasi full stack.



Sebenarnya Spring MVC ini merupakan teknologi yang udah lawas, gengs~

Walaupun lawas, ternyata Spring MVC masih banyak digunakan, lho.

Kayak yang udah pernah dijelaskan di awal course ini kalau banyak banget bank, insurance, ataupun fintech yang udah menggunakan Java sejak lama.

Berhubung sering berkaitan dengan keuangan, aplikasi yang mereka buat nggak cukup agile buat beradaptasi sama teknologi yang baru.



Aplikasi lama yang udah dibuat biasanya udah bugs free (bebas dari bugs) dan dianggap cukup robust buat menangani segala request yang ada.

Kalau dilakukan perubahan, maka bisa memakan biaya yang besar banget.

Misalnya kayak cost berupa resource untuk melakukan regression testing, ataupun cost yang perlu dibayar kalau aplikasi gagal menggantikan aplikasi yang lama.



Oleh karena itu di dalam topic ini, selain membahas tentang Spring MVC, kita juga bakal mempelajari Spring Boot supaya pas di dunia kerja nanti bisa beradaptasi dengan lebih mudah~

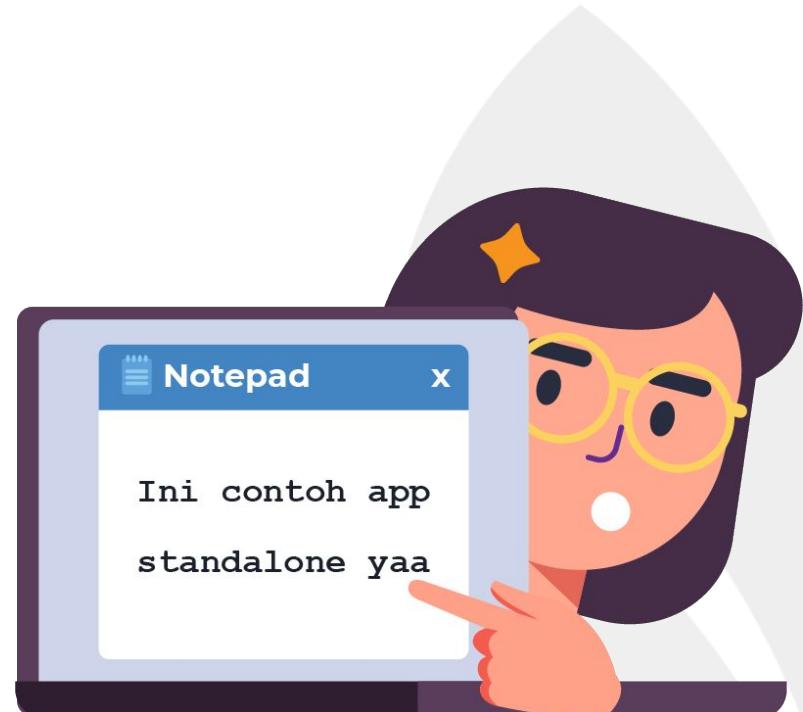


Yang kedua adalah Spring Boot~

Spring Boot merupakan modul dari Spring framework yang biasanya **dipakai untuk membuat aplikasi yang bersifat standalone.**

Biasanya sih aplikasi tersebut digunakan untuk bikin REST API.

Eh iya, tapi Spring Boot nggak terbatas pada REST API aja, kok. Melainkan bisa dipakai juga buat bikin aplikasi yang full stack.



Trus, hubungan antara Spring Framework dengan Spring Boot kayak gimana?

Kamu masih ingat dengan analogi framework dan library yang tadi udah kita bahas di awal topic? Nah mirip-mirip kayak gitu konsepnya.

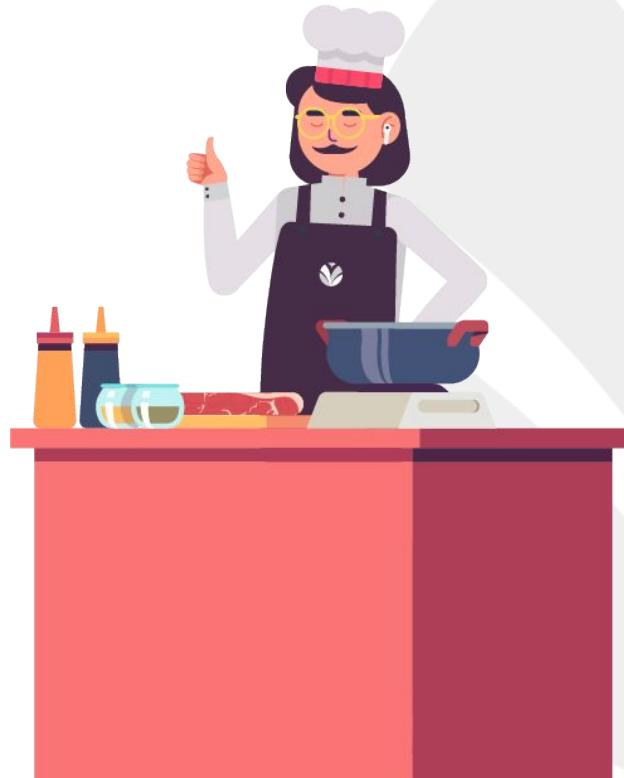
Misal gini, kamu mau masak rendang. Kalau kamu penyuka tantangan, kamu meracik sendiri bumbu rendang mulai dari serai, daun jeruk, bawang putih, bawang merah dan bumbu lainnya.

Tapi, kalo misal kamu lagi nggak punya waktu untuk meracik bumbu sendiri, kamu beli bumbu rendang yang sudah jadi dalam bentuk sachet di supermarket.



Nah, analogi dari bumbu racikan sendiri itu adalah Spring Framework. Sedangkan bumbu siap saji adalah representasi dari Spring Boot.

Balik lagi, mau kamu mau pake bumbu racikan sendiri atau bumbu siap saji outputnya sama aja kan? Yaitu rendang yanglezattt ☐



Lebih lanjut tentang Spring Boot...

Spring Boot memberikan kenyamanan pada developer karena menyediakan konfigurasi yang sifatnya default, tanpa harus bikin konfigurasi kompleks kayak pakai Spring Framework biasa.

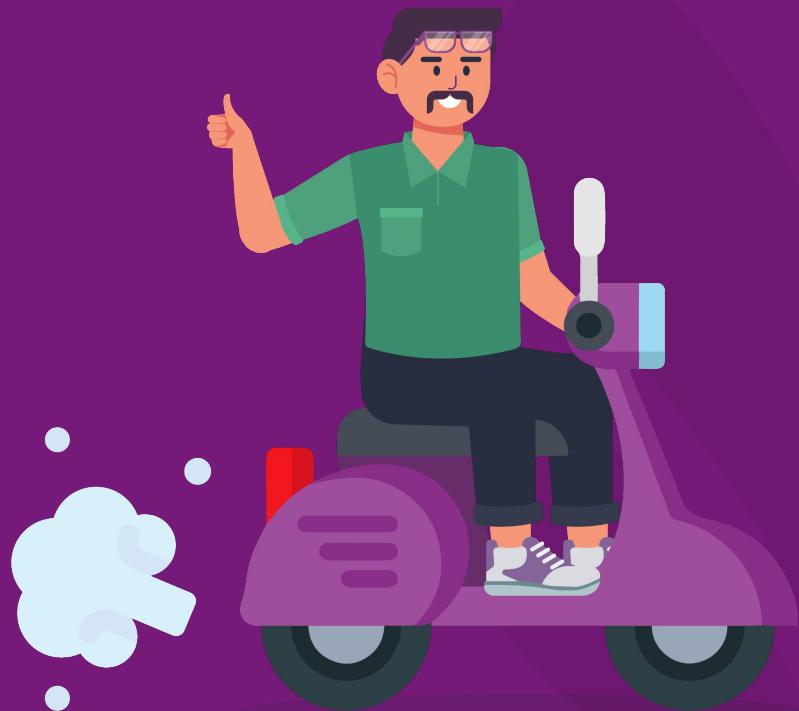
Jadi, **Spring Boot bisa dikatakan sebagai versi dari Spring Framework yang dibuat lebih mudah** karena konfigurasinya diatur secara otomatis.



Gimanaaa?

Sekarang kamu udah ada gambaran tipis-tipis tentang Spring MVC dan Spring Boot?

Biar ada gambaran lainnya, selanjutnya kita bakal kepoin tentang Hello World dengan Spring MVC dan Spring Boot.



“Sebenarnya, apa sih perbedaan Spring MVC dengan Spring Boot?!”

Good question!

Buat tahu perbedaan antara keduanya, kita bakal sedikit melakukan percobaan menggunakan Spring MVC dan Spring Boot untuk membuat Hello world Application.

Mari kita ikuti tutorial di slide selanjutnya, yaaa~



Penasaran sama cara bikinnya? Coba simak tutorialnya, ya!

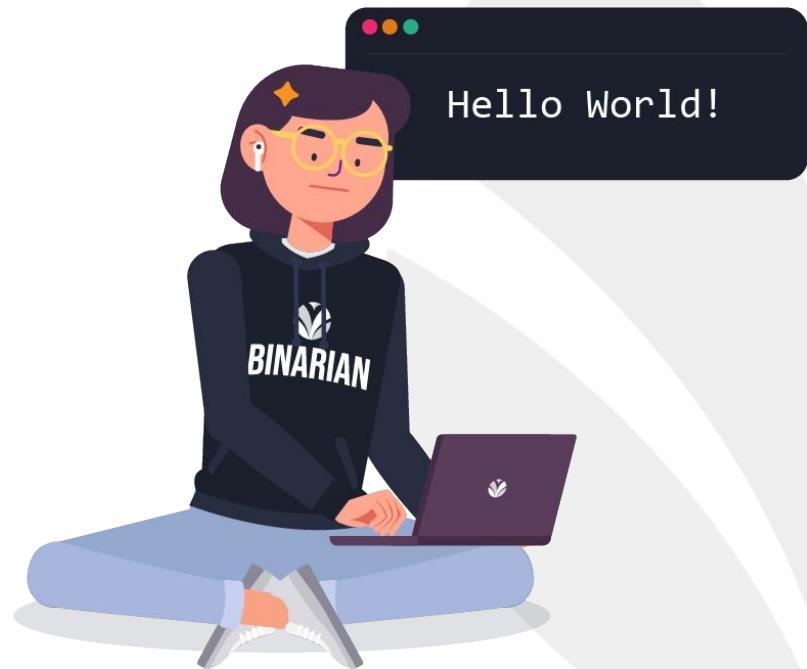
- **Spring MVC**

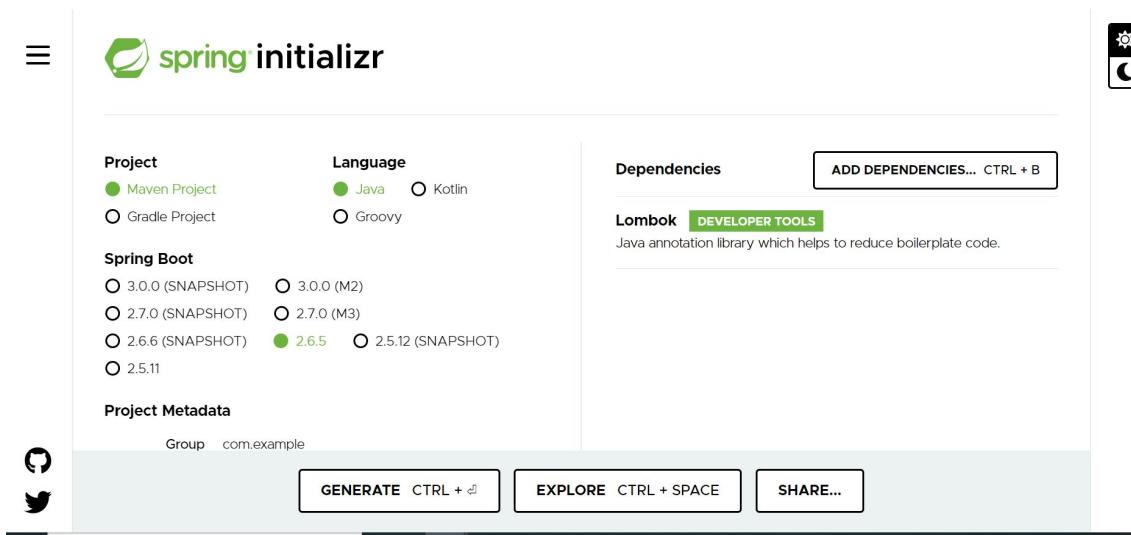
Buat tutorial Spring MVC bisa kamu ikuti dari referensinya [di sini](#), yaaaa~

- **Spring Boot**

Buat tutorial Spring Boot, kita bakal mengikuti tutorial dengan referensi [di sini](#), yaaa~

Setelah menggunakan Spring MVC dan Spring Boot, kira-kira apa sih perbedaan yang kamu rasakan?





The screenshot shows the Spring Initializr interface. On the left, there's a sidebar with a menu icon (three horizontal lines) and social sharing icons for GitHub and Twitter. The main area has sections for 'Project' (Maven Project selected), 'Language' (Java selected), 'Dependencies' (Lombok selected under DEVELOPER TOOLS), and 'Spring Boot' (version 2.6.6 selected). Under 'Project Metadata', the group is set to com.example. At the bottom are buttons for 'GENERATE' (CTRL + ⌘), 'EXPLORE' (CTRL + SPACE), and 'SHARE...'.

Berbeda dengan Spring MVC yang memulai project dengan menggunakan project Maven, **Spring Boot** udah menyediakan project starter dengan menggunakan **Spring Initializr**.

Spring initializr bisa diakses di [sini](#) ya, gengs!

Beberapa IDE juga udah mendukung Spring initializr, lho!

IntelliJ Ultimate bisa bikin project Spring Boot secara langsung.

Oh iya, di Eclipse kita juga bisa melakukan inisiasi project kok, tapi kita harus install plugins Spring Tools-nya dulu.



Meskipun nggak selengkap di Spring initializr, tapi dependency yang disediakan udah cukup buat meng-cover kebutuhan dasarnya. Atau kita juga bisa pakai Spring tool suite yang punya tampilan serupa kayak eclipse.

Wuih, siapa sangka ternyata banyak pilihannya, ya?!



Berikut adalah cara untuk melakukan inisiasi project Spring Boot pada IntelliJ!

Simak cara-caranya ya, gengs.

- Klik **File > New > Project > Spring Initializr**
- Kalau pakai Spring initializr, kita bakal dapat structure project Maven yang udah distandarisasi sama Spring Boot pakai project Default.

Selain itu kita juga bisa menambahkan dependency secara langsung dalam inisiasi project-nya.



Tadi kan kita udah sempet mention tentang project structure dari Spring MVC dan Spring Boot.

Tapi sebetulnya, apa sih konsep dari Project Structure itu sendiri?

Kayaknya kita perlu kupas tuntas yang satu ini dulu, deh~



Catatan dulu, nih. Pada pembahasan project structure disini, pembahasannya cuma terbatas pada penggunaan Maven aja, ya.

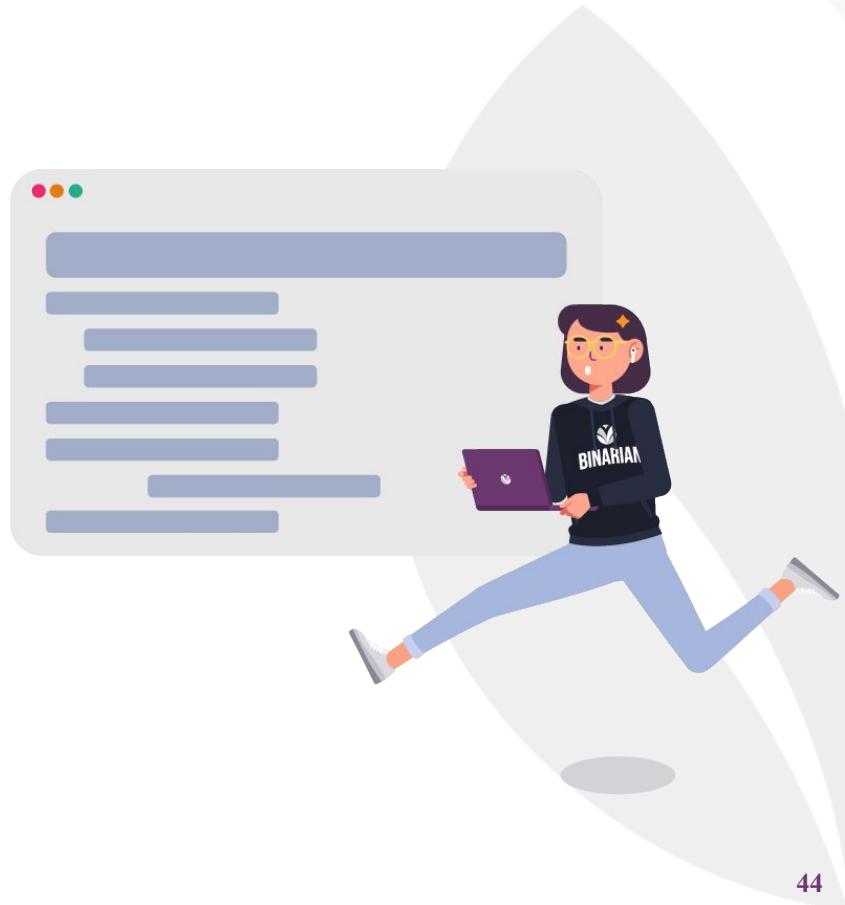
Dari situ, project yang pakai [Gradle](#) nggak bakal dijelaskan di sini. Tapi jangan khawatir. Secara garis besar, implementasi dari Maven dan Gradle sama, kok.



Berikut adalah project structure Spring MVC!

Pada Spring MVC secara default punya project structure kayak di bawah ini:

- **src/main/java**, berisi source code dari business logic.
- **src/main/resources**, berisi file konfigurasi dari project tersebut. Dimana file resource ini biasanya berbentuk xml.
- **src/main/webapp**, berisi resource dari file web kayak HTML, CSS dan web.xml. File web.xml berfungsi untuk memetakan servlet, controller dan juga file dari view yang dipakai.



Kalau yang ini adalah project structure dari Spring Boot~

Sebetulnya tuh, secara default si Spring Boot ini punya project structure yang hampir mirip kayak Spring MVC.

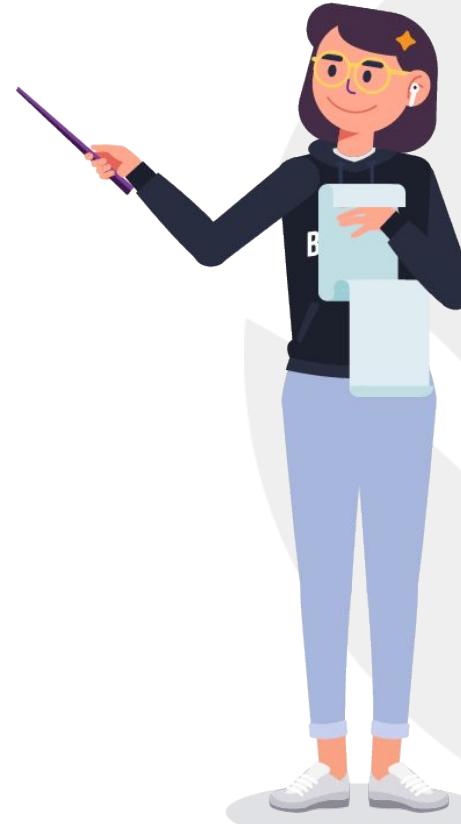
Coba deh kamu lihat project-project structure dibawah ini:

- **src/main/java**, berisi source code dari business logic. Ada class utamanya, yaitu class yang punya annotation `@SpringBootApplication` yang dipakai supaya aplikasi bisa dijalankan.



- **src/main/resources**

yaitu, berisi application.properties dan resource lainnya. Kalau menambahkan file html ataupun css yang mau dipakai untuk membentuk suatu view, maka file tersebut secara default diletakkan di folder resource.



“Kenapa sih Spring Boot nggak punya folder webapp kayak di Spring MVC?”

Jawabannya karena **Spring Boot bisa dipakai untuk kebutuhan yang multipurpose.**

Berikut merupakan contoh sebuah code structure yang best practice dan sering digunakan di Java

Boleh kamu cek [di sini](#) ya, bestie~



Kita udah menuju akhir materi di topic kali ini nih, gengs.

Lastly, kita bakal kepoin tentang **MVC Design Pattern**.

Kalau gitu, kita lanjuuuut!

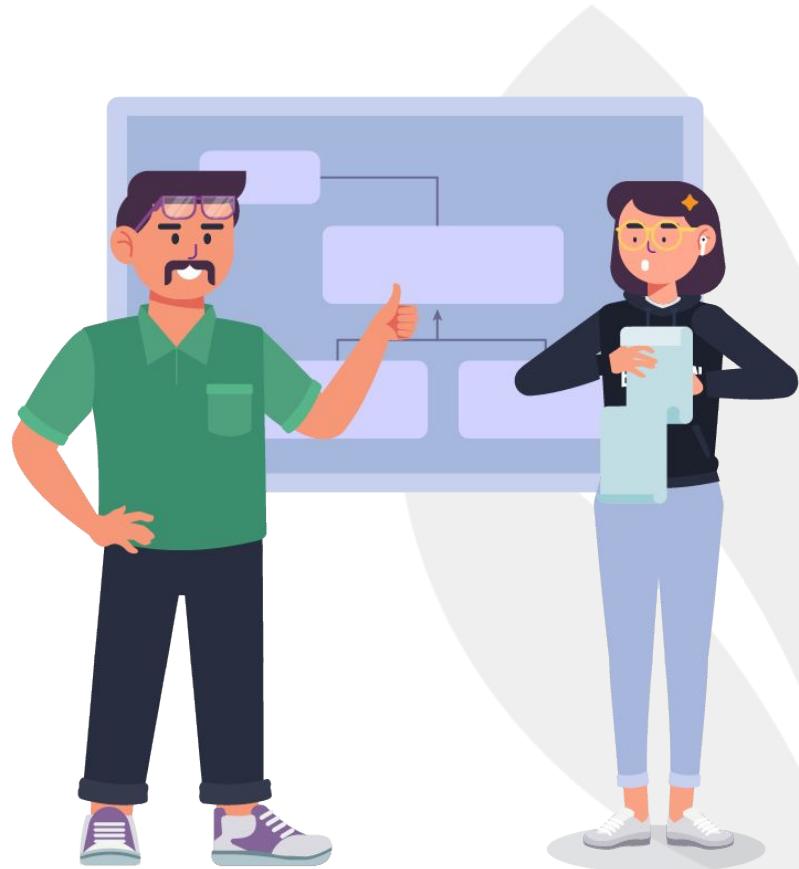


Pada course ini, aplikasi yang bakal kita buat punya arsitektur MVC, lho!

MVC (Model View Controller) merupakan salah satu architectural pattern.

“Hah apaan tuh?”

Architectural pattern adalah pattern atau solusi yang udah pernah dibuat sebelumnya dan dipakai sebagai arsitektur dari sebuah software.



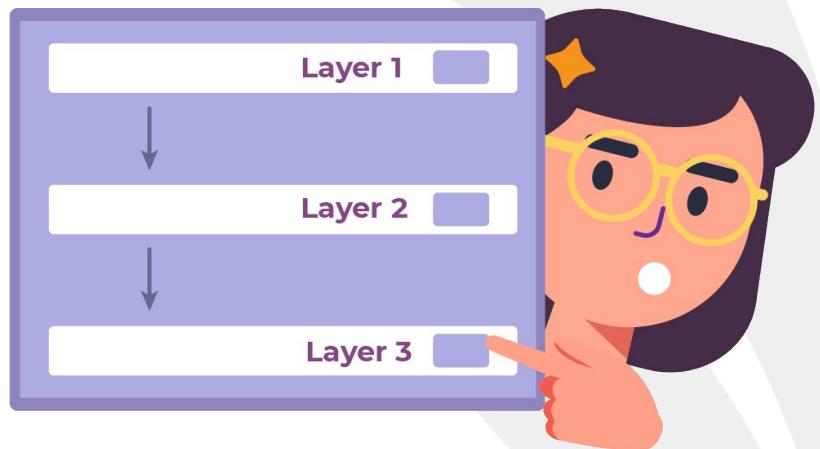
Walaupun udah kenalan, tapi kita belum kenal lebih dalam nih tentang pengertian dari MVC itu sendiri

MVC adalah salah satu turunan dari salah satu architectural pattern lain, yaitu **layered pattern**.

Layered pattern merupakan architectural pattern yang memakai beberapa layer program untuk membentuk suatu software.

Nah, jumlah dari layer tersebut bisa beragam sesuai dengan kebutuhan.

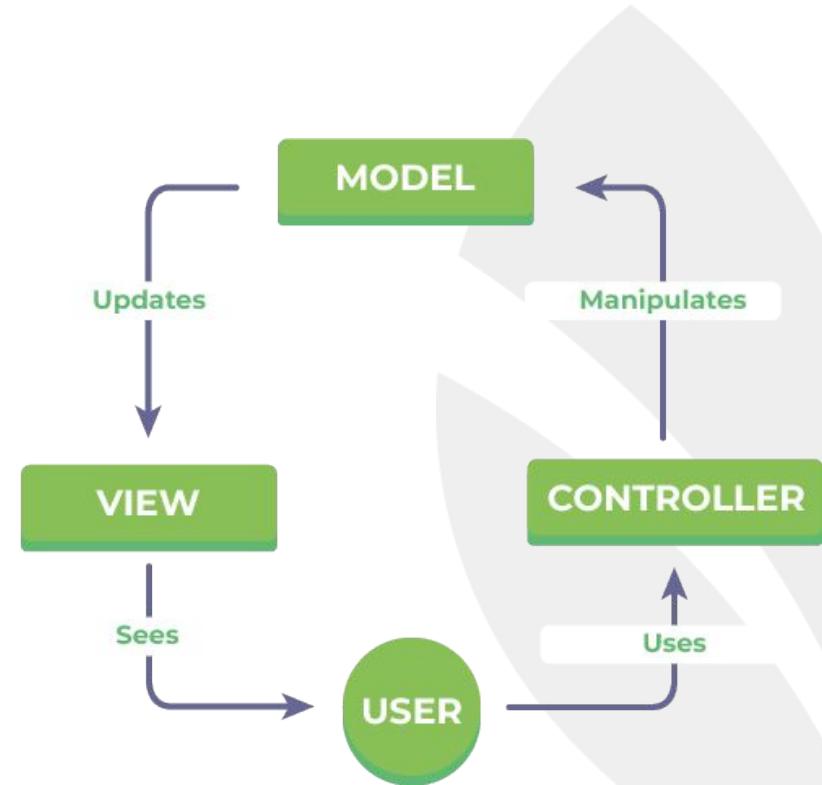
Sistem ini pake architecture tipe layered pattern



Arsitektur MVC terdiri dari tiga layer nih, gengs!

Layer tersebut terdiri dari Model, View dan Controller. Berikut rincian dari fungsi masing masing layer:

- **Model**, yaitu layer data dari aplikasi yang dibuat untuk dikirimkan ke layer lain.
- **View**, yaitu layer untuk menampilkan dan memberikan perintah pada aplikasi.
- **Controller**, yaitu layer yang berisi business logic untuk mengolah request yang berasal dari view.

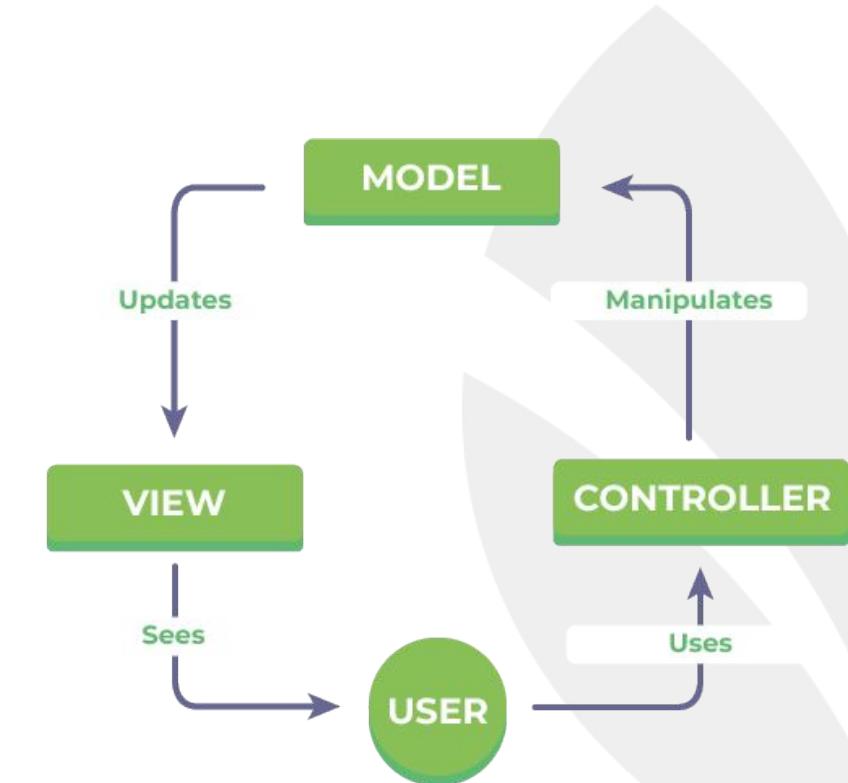


Bukan cuma aku, kamu, dan dia aja yang bestfriend 4ever, ketiga layer tersebut juga saling bestie, lho~

Iya, jadi ketiga layer tersebut akan berinteraksi untuk menjalankan sebuah program.

“Terus, apakah tiap layer harus merupakan layer program yang berbeda?”

Jawabannya tidak. MVC ini merupakan sebuah arsitektur. Kalau nggak ada ketiga layer (model, view dan controller) berarti itu bukan MVC, ya!

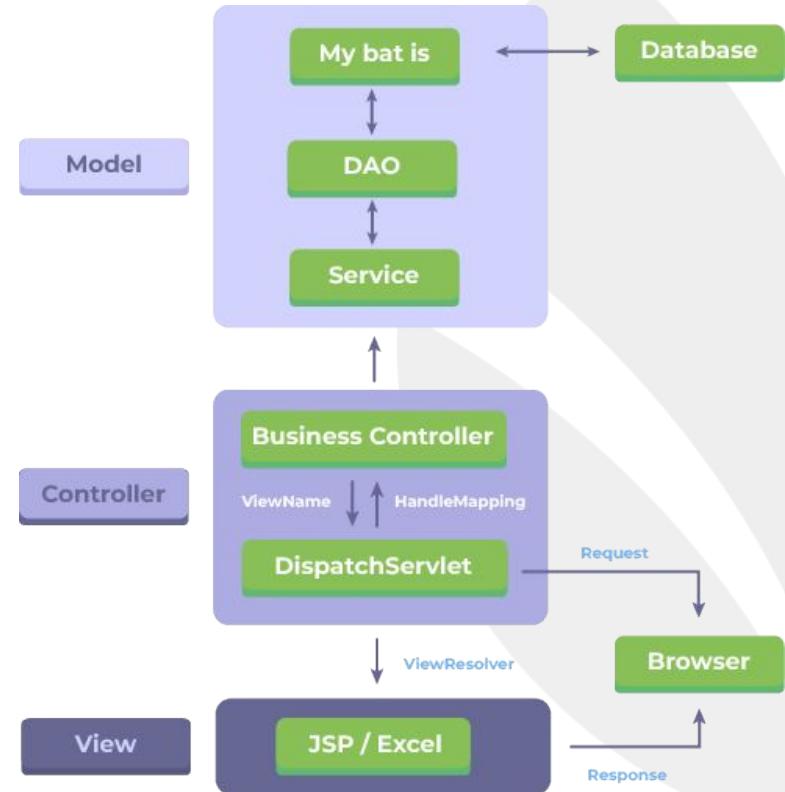


Berikut merupakan contoh salah satu implementasi dari arsitektur MVC dengan menggunakan Spring MVC~

Teknologi yang digunakan untuk menghubungkan program ke database adalah dengan My bat is.

Teknologi lain yang bisa digunakan contohnya Spring Data JPA atau hibernate.

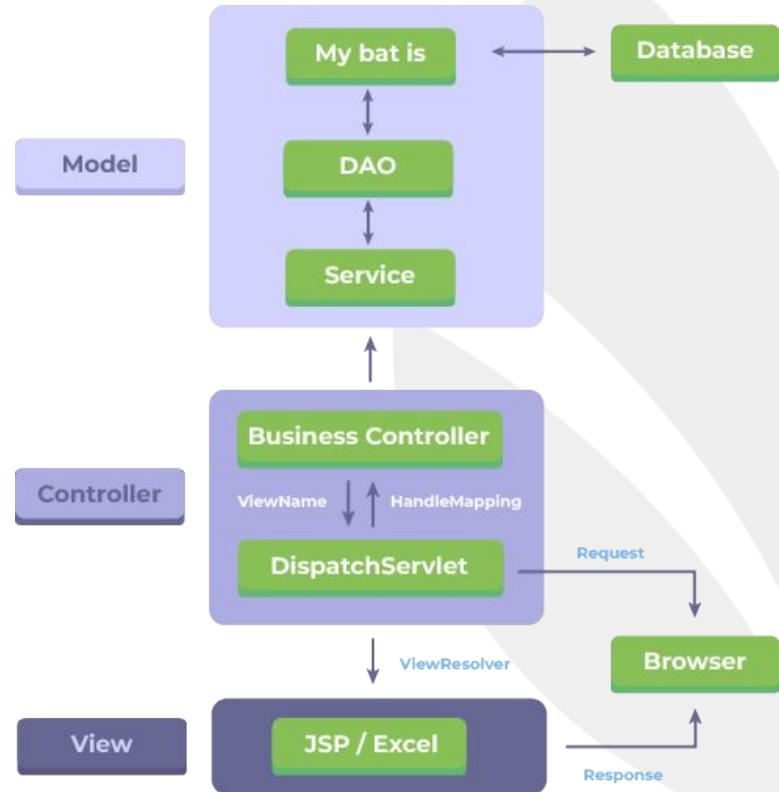
Kemudian, controller dibuat menggunakan library Spring web. Sedangkan pada viewnya menggunakan JSP.



Teknologi lainnya yang dapat digunakan untuk men-develop view layer yaitu [Thymeleaf](#).

Eits... tenang bestie di course ini kita bakal fokus sama Model dan Controller aja, jadi kita nggak perlu men-develop view layer.

Santuyyy~



Good job! Materinya udah hampir selesai nih, gengs.

Untuk menutup Spring Framework pada topik ini, sekarang kita bakal cari tahu tentang **Application Server**. Yuk~



“Jadi, Application Server itu apa dong?”

Application Server merupakan JVM yang bakal menjalankan aplikasi Java di dalamnya.

Application server yang dipakai ini berupa Application webserver karena di course ini kita bakal bikin RESTful API.

Selain itu, Application webserver juga bisa menghandle request dengan koneksi HTTP dan HTTPS.



Sebelum tahu konsepnya, kita harus tahu dulu nih keberadaan dari Application Server ini!

Jadi gini, kalau kita menggunakan Spring Boot, maka si Application Server udah otomatis tersedia di dalamnya.

Application server yang tersedia di dalam Spring Boot secara default ini adalah [Apache Tomcat](#).

Berikut referensi yang membahas secara detail mengenai Apache Tomcat bisa kamu cek [di sini](#), ya.



Nah, sekarang kamu jadi paham deh mengenai hubungan antara **Spring Framework** dan **Spring Boot**~

Menurutmu, apa sih kelebihan dan kekurangan dari masing-masing framework tersebut?



Nah, selesai sudah pembahasan kita di Chapter 4 Topic 1 ini.

Selanjutnya, kita bakal bahas tentang Spring Framework part 2.

Penasaran kayak gimana? Yuk, langsung ke topik selanjutnya~

