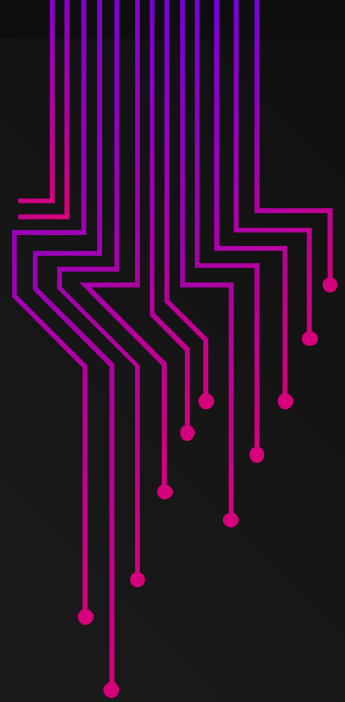


# Java Programming Spring Boot

Java (SE, EE) >> EJB vs Spring >> Spring Boot



# My Programming experience

**VB6**

Basic  
Book (Deitel & Deitel)

—• **.net (VB, C#, J#)** —•

OOP  
(CD > MSDN)

**PHP**

Wordpress  
Web

**Java**

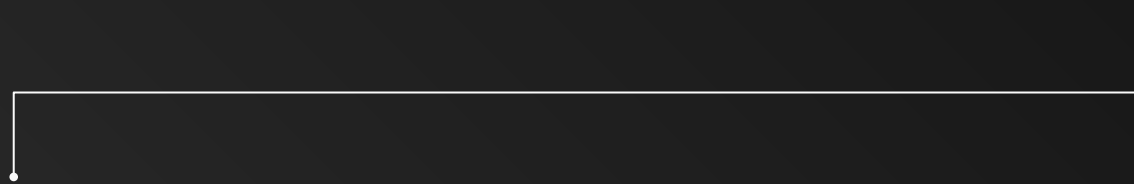
SE, EE  
Class (ACM) & Web

—• **Spring** —•

Web  
Google, spring.io

**Spring Boot**

Web  
Google, spring.io



# What?

## VB6

Basic concepts  
IF, Loop, Exception

## .net (VB, C#, J#)

OOP  
Class, Method

## PHP

Network  
Rest, soap, network

## Java

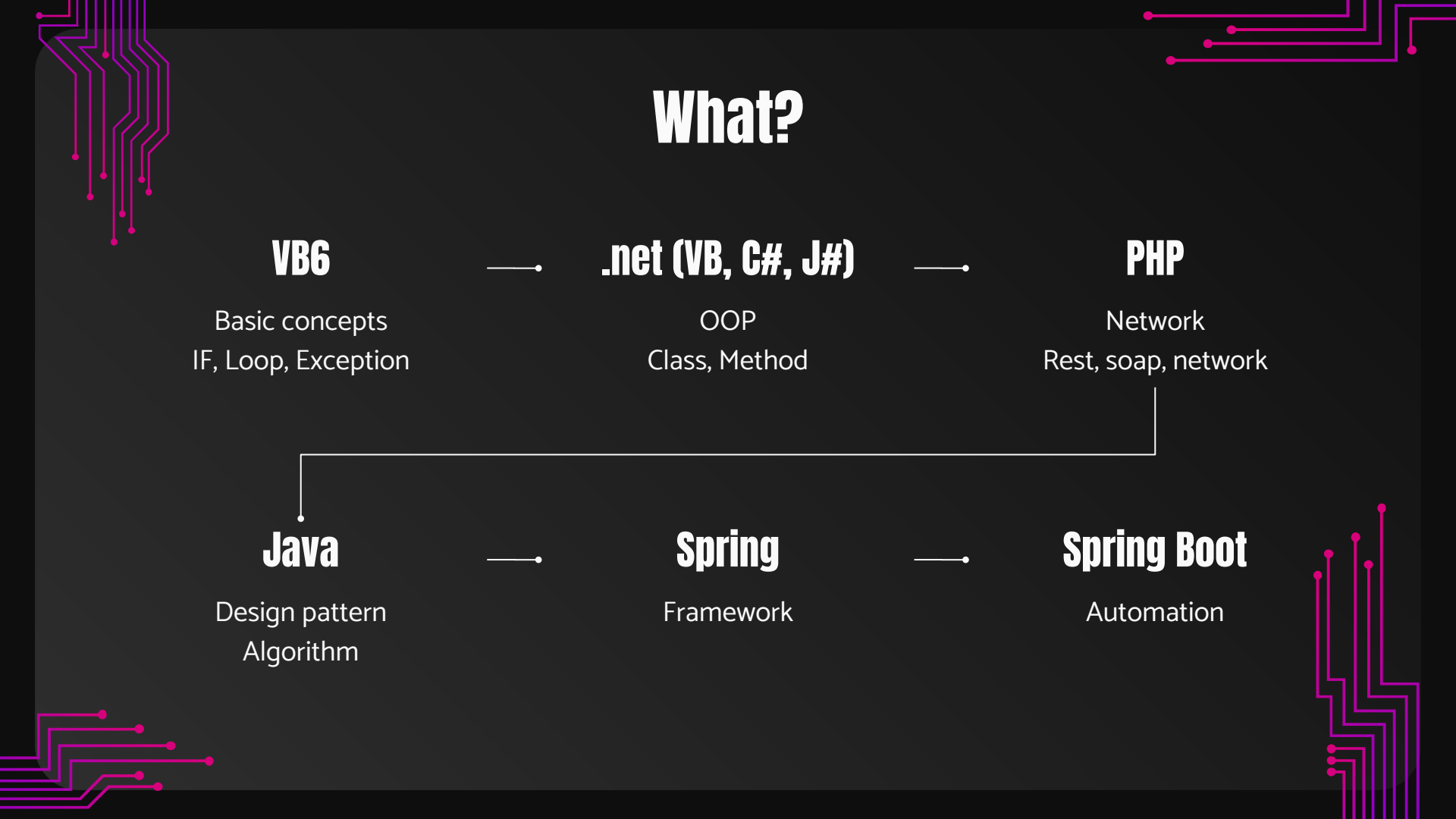
Design pattern  
Algorithm

## Spring

Framework

## Spring Boot

Automation



# Where ?

**01**

**Book, Multimedia, CD**

Algorithm, Language, Topic

**02**

**Class**

Standard, Certificate

**03**

**Web**

Google, stackoverflow, mkyong,  
dzone, tutorialspoint

**04**

**AI**

Chatgpt, copilot, gemini, grok

# Java Edition

## JAVA SE

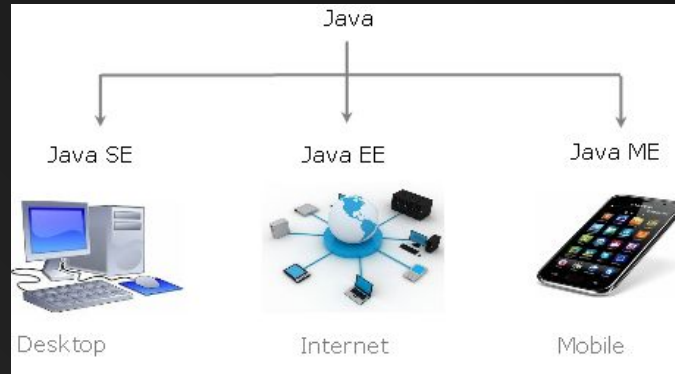
Standard  
Edition

## JAVA EE

Enterprise  
Edition

## JAVA ME

Micro  
Edition

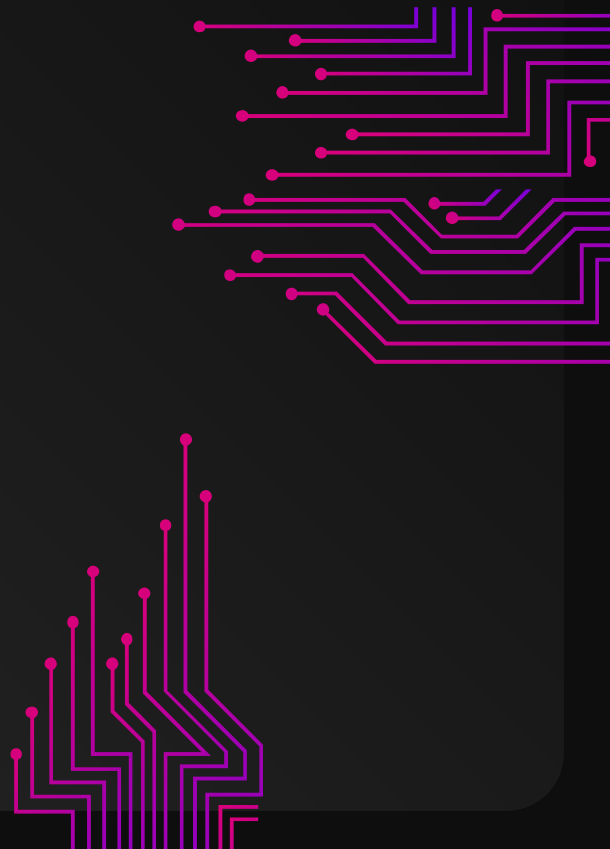


# Java SE

- SE یعنی "Standard Edition" یا نسخه استاندارد.
- شامل امکانات پایه و اصلی جاواست.
- ابزارهایی برای ساخت برنامه‌های دسکتاپ، کنسول یا اپ‌های ساده.
- شامل کتابخانه‌هایی مثل:
  - `java.lang`, `java.util`, `java.io`
  - شبکه (`java.net`)
  - رابط گرافیکی (Swing, AWT)
  - `.Collections`, `Streams`, `Threads`, etc

✓ مثال استفاده:

نوشتن ماشین حساب دسکتاپ، اپلیکیشن CLI، یا اپ‌های کوچکی که روی یک کامپیوتر اجرا می‌شن.



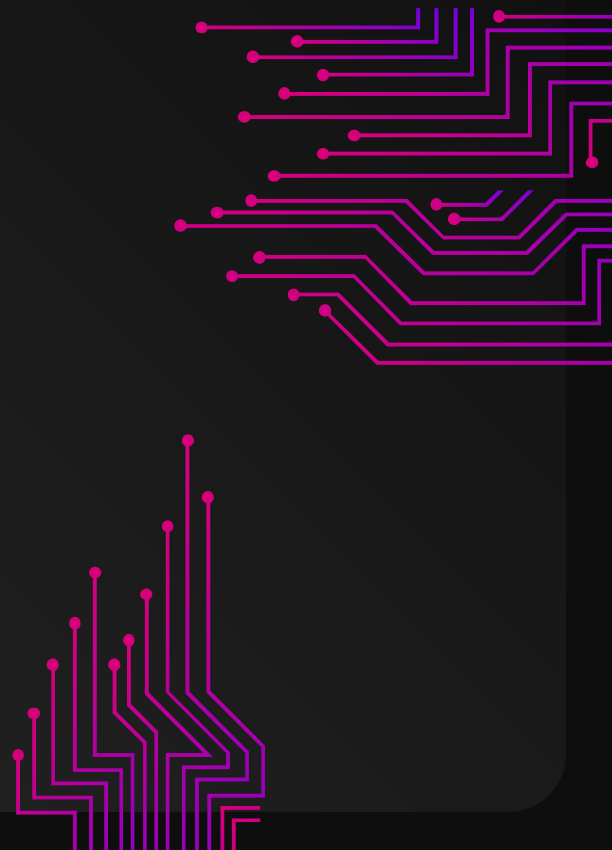
# Java EE

- EE یعنی "Enterprise Edition" یا نسخه سازمانی.
- بر پایه Java SE ساخته شده ولی کلی امکانات اضافه‌تر دارد.
- مناسب برای ساخت برنامه‌های وب، API، میکروسرویس‌ها، سیستم‌های بزرگ سازمانی.
- شامل کتابخانه‌ها و تکنولوژی‌هایی مثل:

- Servlets, JSP, JSF
- JPA برای ارتباط با دیتابیس
- EJB برای پردازش‌های سازمانی
- SOAP و REST برای JAX-RS, JAX-WS
- Dependency Injection (CDI)
- Transaction Management و Security

✓ مثال استفاده:

ساخت سیستم بانکی، فروشگاه اینترنتی، backend برای اپلیکیشن موبایل، RESTful API با امنیت و مقیاس‌پذیری بالا.



# JAVA SE VS EE

مقایسه خلاصه: **vs**

Java EE (Jakarta EE)

Java SE

ویژگی

سازمانی، وب، API

دسکتاپ، ساده

نوع برنامه

حرفه‌ای، پیشرفته

پایه‌ای

امکانات

بیشتر

کمتر

پیچیدگی

JPA , Servlets , JAX-RS

java.util , java.io

کتابخانه‌های مهم

پروژه‌های بزرگ، سیستم‌های توزیع‌شده

شروع یادگیری، اپ‌های کوچک

مناسب برای



## EJB چیه توی Java EE؟

- EJB یعنی Enterprise Java Bean.
- EJB یک نوع کلاس جاواست که منطق اصلی برنامه (Business Logic) رو توش می‌نویسی.
- اما فرقش با به کلاس معمولی چیه؟
- اینه که EJB رو توی یه EJB Container اجرا می‌کنی، که خودش کارای سخت رو انجام می‌ده.

مثلاً:

- خودش تراکنش رو مدیریت می‌کنه
- خودش امنیت و دسترسی‌ها رو بررسی می‌کنه
- خودش Bean رو ساخته و نابود می‌کنه
- خودش منابع (مثل دیتابیس) رو تزریق می‌کنه



تو فقط می نویسی:

```

@Stateless
public class HelloService {
    public String sayHello(String name) {
        return "Hello, " + name;
    }
}

```







و باقی کارها رو EJB Container انجام می‌ده!





## What does the EJB Container do?

Here's a breakdown of key responsibilities:

Responsibility	Description
 Lifecycle Management	Automatically creates, pools, and destroys beans as needed.
 Security	Controls access to methods based on user roles or permissions.
 Transaction Handling	Manages transactions (e.g., begin, commit, rollback) for you.
 Dependency Injection	Injects other beans or resources (like DB connections) into your bean.
 Concurrency Control	Manages multi-threading so your code stays thread-safe.
 Remote Communication	Supports RMI so beans can be called remotely (like microservices).

# Lifecycle Management

```
4 public class ApplicationRunner {  
5     public static void main(String[] args) {  
6         try (ConsoleUI consoleUI = new ConsoleUI()) {  
7             consoleUI.startMenu();  
8         } catch (Throwable ex) {  
9             ex.printStackTrace();  
10            System.out.println("System error!");  
11        }  
12    }  
13  
14 }
```

```
4 public class ConsoleUI implements AutoCloseable {  
5  
6     @Override  
7     public void close() {  
8         scannerWrapper.close();  
9     }  
10 }  
11
```

# Dependency Injection (DI)

```
20 public class AccountFacadeImpl implements AccountFacade {
21     private ValidationContext<AccountDto> validationContext;
22     private AccountService accountService;
23
24     private final CustomerService customerService;
25     private final AccountMapstruct accountMapstruct;
26     private static final AccountFacadeImpl INSTANCE;
27
28     > public static AccountFacadeImpl getInstance() { return INSTANCE; }
29
30     public static AccountFacadeImpl getInstance(AccountService accountService){
31         INSTANCE.accountService = accountService;
32         return INSTANCE;
33     }
34
35     static {
36         INSTANCE = new AccountFacadeImpl();
37     }
38
39     private AccountFacadeImpl() {
40         this.accountMapstruct = Mappers.getMapper(AccountMapstruct.class);
41         this.accountService = AccountServiceImpl.getInstance();
42         this.customerService = CustomerServiceImpl.getInstance();
43         this.validationContext = new AccountValidationContext();
44     }
45 }
```

# Security

```
@Override
public Boolean login(String username, String password) {
    try {
        Customer customer = searchCustomersByEmail(username);
        if(Objects.equals(
            customer.getPassword(),
            PasswordEncoderUtil.encodePassword(password, customer.getId()))){
            globalAttributes.setCustomerId(customer.getId());
            return true;
        }else{
            globalAttributes.setCustomerId(null);
            return false;
        }
    } catch (CustomerNotFoundException e) {
        globalAttributes.setCustomerId(null);
        return false;
    }
}
```

```
6 @Setter
7 @Getter
8 public class GlobalAttributes {
9     private static final GlobalAttributes INSTANCE;
10
11 > public static GlobalAttributes getInstance() { return INSTANCE; }
12
13
14
15     private Integer customerId;
16
17     static {
18         INSTANCE = new GlobalAttributes();
19     }
20
21 > private GlobalAttributes() { customerId = null; }
22
23
24
25 }
```

```
38 public void menu() {
39     Integer customerId = globalAttributes.getCustomerId();
40     if(customerId != null) {
41         menuWithUser();
42     }else
43         menuNoUser();
44 }
```

# Concurrency Control

```
194 @Override
195 public synchronized void transfer(int fromAccountId, int toAccountId, Amount amount) throws AccountNotFoundException,
196     Lock fromLock = getLock(fromAccountId);
197     Lock toLock = getLock(toAccountId);
198
199     Lock firstLock = fromAccountId < toAccountId ? fromLock : toLock;
200     Lock secondLock = fromAccountId < toAccountId ? toLock : fromLock;
201
202     firstLock.lock();
203     secondLock.lock();
204
205     try{
206         Account fromAccount = getAccountById(fromAccountId);
207         Account toAccount = getAccountById(toAccountId);
208
209         if(amountUtil.compareTo(amount, fromAccount.getBalance()) > 0){
210             throw new ValidationException("The amount is larger than fromAccount balance!");
211         }
212         fromAccount.setBalance(amountUtil.subtract(fromAccount.getBalance(), amount));
213         toAccount.setBalance(amountUtil.add(toAccount.getBalance(), amount));
214     }finally {
215         firstLock.unlock();
216         secondLock.unlock();
217     }
218 }
```

## EJB Container چیست؟ ✓

EJB Container (مخزن EJB) یک محیط اجرایی (runtime environment) است که:

برنامه‌های EJB (یعنی Enterprise JavaBeans) را اجرا، مدیریت، و پشتیبانی می‌کند.

یعنی وقتی تو یک کلاس EJB مثل `Stateless@` یا `Stateful@` می‌نویسی، این Container مسئول اینه که:

- Bean رو بسازه یا از حافظه پاک کنه
- بهش منابع (مثلاً اتصال دیتابیس) بده
- امنیتش رو کنترل کنه
- تراکش‌ها رو براش مدیریت کنه
- و همه چیز رو پایدار و مطمئن نگه داره



## 🧠 چرا به EJB Container نیاز داریم؟

EJB ها بدون Container کار نمی‌کنن چون:

1. خودشون چیزی نیستن جز یه کلاس ساده.

2. این Container هست که:

- Annotation های مثل `Stateless@` یا `Transactional@` رو می‌فهمه.
- Bean رو در جای درست قرار می‌ده.
- منابع (مثل دیتابیس) رو تزریق می‌کنه.
- امنیت و تراکنش‌ها و... رو اجرا می‌کنه.

پس Container مثل موتور خودروی EJB هست. بدونش EJB فقط یه اسکلت خالیه.

## EJB Container چگونه فراهم می‌شود؟

EJB Container بخشی از Application Server های Java EE هست.

یعنی برای داشتنش باید از یکی از این سرورها استفاده کنی:

توضیح	Application Server
رایگان، پرمصرف، پشتیبانی کامل از EJB	WildFly (قبلاً JBoss)
سرور رسمی مرجع Jakarta EE	GlassFish
نسخه تجاری‌تر و بهینه‌تر از GlassFish	Payara
سازمانی و تجاری، مناسب برای محیط‌های بزرگ	WebLogic (Oracle)
بسیار قوی و گران، مخصوص سازمان‌های بزرگ	WebSphere (IBM)

تو این سرورها، EJB Container به‌صورت پیش‌فرض نصب شده. فقط کافیه برنامه‌ات رو Deploy کنی و تمام.

## EJB Container چطور فراهم می‌شه؟

EJB Container بخشی از Application Server های Java EE هست.

یعنی برای داشتنش باید از یکی از این سرورها استفاده کنی:

توضیح	Application Server
رایگان، پرکاربرد، پشتیبانی کامل از EJB	WildFly (قبلاً JBoss)
سرور رسمی مرجع Jakarta EE	GlassFish
نسخه تجاری‌تر و بهینه‌تر از GlassFish	Payara
سازمانی و تجاری، مناسب برای محیط‌های بزرگ	WebLogic (Oracle)
بسیار قوی و گران، مخصوص سازمان‌های بزرگ	WebSphere (IBM)

تو این سرورها، EJB Container به‌صورت پیش‌فرض نصب شده. فقط کافیه برنامه‌ات رو Deploy کنی و تمام.

## 💰 مقایسه‌ی هزینه و مجوزهای Application Server ها برای EJB

سرور (Application Server)	هزینه	نوع مجوز	ویژگی‌ها	مناسب برای
WildFly (JBoss)	✅ رایگان	Apache License 2.0	متن‌باز، سبک، فعال در جامعه‌ی توسعه‌دهنده	پروژه‌های کوچک تا متوسط، یادگیری و توسعه
GlassFish	✅ رایگان	EPL + GPL	سرور مرجع Jakarta EE، آپدیت‌های رسمی، نه خیلی سریع	آموزش، پروژه‌های تستی، یادگیری
Payara Server (Community)	✅ رایگان	CDDL + GPL	نسخه پایدارتر از GlassFish، بهبود یافته	توسعه رایگان، مناسب‌تر از GlassFish
Payara Server Enterprise	💰 پولی (با پشتیبانی)	تجاری (Subscription)	پشتیبانی رسمی، آپدیت امنیتی، SLA	سازمان‌ها، بانک‌ها، اپ حساس
Red Hat JBoss EAP	💰 پولی (ولی سانس آزاد)	اشتراک سالانه Red Hat	نسخه پایدارتر از WildFly با پشتیبانی حرفه‌ای	سازمان‌ها و شرکت‌های بزرگ
Oracle WebLogic	🌸 خیلی گران	مجوز تجاری Oracle	قوی، سنگین، مناسب سازمان‌های خاص بزرگ، با قابلیت‌های خاص	بانک‌ها، دولت، ERP
IBM WebSphere	🌸🌸🌸 خیلی گران	تجاری	بسیار پیشرفته، امکانات سازمانی کامل، پیچیده	شرکت‌های چندملیتی، سیستم‌های بحرانی

## جزئیات قیمت تقریبی (ممکنه با زمان و شرایط خاص تغییر کنه):

سرور	قیمت پایه سالانه (تقریبی)
WildFly	رایگان
GlassFish	رایگان
Payara Community	رایگان
Payara Enterprise	از حدود \$2,500 تا \$10,000 در سال (بسته به نیاز)
JBoss EAP	حدود \$8,000+ در سال برای هر CPU socket
Oracle WebLogic	حدود \$25,000+ در سال + لایسنس Oracle Java
IBM WebSphere	حتی بیش از \$30,000 در سال (بسته به پیکربندی)

## نتیجه‌گیری برای تو به عنوان برنامه‌نویس یا یادگیرنده: ✓

پیشنهاد من

اگر توی این دسته‌ای...

WildFly یا Payara Community (رایگان و راحت)

تازمکار یا دانشجو

WildFly یا Payara با Docker

تیم کوچک یا استارت‌آپ

Payara Enterprise یا JBoss EAP

شرکت متوسط که دنبال پشتیبانیه

WebLogic یا WebSphere (با پشتیبانی حرفه‌ای)

سازمان بزرگ با سیستم‌های حساس

در اوایل دهه ۲۰۰۰، کار کردن با Java EE (و به خصوص EJB) خیلی سنگین، پیچیده و دست‌وپاگیر بود.

● مثلاً برای اینکه به کلاس ساده‌ی Business Logic بنویسی، باید:

- کلی XML بنویسی
- Bean تعریف کنی
- کانفیگ‌های امنیت و تراکنش انجام بدی
- روی یک Application Server خاص Deploy کنی (که خیلی زمان‌بر بود)

⚠ این پیچیدگی باعث شده بود خیلی از برنامه‌نویس‌ها خسته بشن و دنبال یه راه ساده‌تر باشن.

در دوران طلایی Java EE (اوایل دهه ۲۰۰۰):

- برای استفاده از EJB باید برنامه رو روی یک Application Server سازمانی مثل WebLogic یا WebSphere اجرا می‌کردی.
- این سرورها:
- سنگین بودن 🐢
- کند بودن 🐢
- و مهم‌تر از همه، گران بودن 💰💰

فقط لایسنس WebLogic گاهی ده‌ها هزار دلار در سال هزینه داشت!

برای شرکت‌های کوچک یا استارت‌آپ‌ها، این هزینه‌ها واقعاً غیرقابل قبول بود.



## چه کسی Spring رو ساخت؟ 🧑💻

Rod Johnson، یه برنامه‌نویس و معمار نرم‌افزار استرالیایی، در سال ۲۰۰۲ یه کتاب نوشت به اسم:

*"Expert One-on-One J2EE Design and Development"*

توی اون کتاب:

- نشون داد EJB چقدر زیادی پیچیده‌ست
- و یه راه حل سبک‌تر معرفی کرد

بعدش همون سال، نسخه‌ی ابتدایی **Spring Framework** رو معرفی کرد.

## 🔥 ظهور Spring: نرم افزار سبک، رایگان، انقلابی

وقتی Rod Johnson اومد و Spring رو معرفی کرد:

- همه چی **Open Source** و رایگان بود ✅
- بدون نیاز به Application Server اجرا می شد 😍
- از همون ابتدا روی سادگی، ماژولار بودن و قابل تست بودن تمرکز داشت
- این موضوع مثل یک زلزله توی دنیای Java بود. 🔥

## Spring چی کار می‌کنه و چه ربطی به EJB داره؟ 🛠️

Spring	EJB
می‌تونه روی سرور معمولی یا حتی داخل IDE اجرا بشه	روی Application Server اجرا می‌شه (مثل WebLogic، WildFly)
سبک، ساده، قابل تست، بدون وابستگی سنگین	پیچیده و وابسته به Container
برای پروژه‌های کوچک تا Enterprise قابل استفاده‌ست	برای پروژه‌های بزرگ سازمانی طراحی شده
از Annotation ها و Java Config پشتیبانی می‌کنه	از XML زیاد استفاده می‌کرد (قدیم‌تر)

Spring اومد جایگزین EJB شه با مفاهیمی مثل:

• @Service, @Component, @Repository (معادل Bean ها)

• @Transactional برای مدیریت تراکنش

• Dependency Injection ساده‌تر

• تست‌پذیری بالا

• اجرا بدون نیاز به Application Server

## برخورد جامعه Java EE و سازمان استانداردسازی (JSR)

اینجا داستان جنجالی می‌شه:

در اواخر زمان، Java EE از طریق سازمانی به نام JCP (Java Community Process) استانداردسازی می‌شد. Spring چون مستقل و بیرون از JCP بود، قبولش نداشتن.

● به‌طور خاص، وقتی پروژه‌ای به اسم Java EE 5 اومد، سازمان JCP گفت:

"Spring استاندارد نیست. استفاده ازش ممکنه در آینده مشکل‌ساز بشه."

و حتی جملات تندی گفته شد، مثل اینکه:

"Spring باعث هرج‌ومرج در Java Enterprise می‌شه."

حتی شرکت‌هایی که توی استانداردسازی Java EE بودن (مثل Oracle و IBM) حاضر نشدن Spring رو به عنوان یه فریم‌ورک رسمی قبول کنن.

## 🧠 فلسفه‌ی اصلی Spring:

وقتی Spring اومد، Rod Johnson و تیمش گفتن:

"Java EE (یا همون J2EE اون زمان) یه مجموعه از ابزارها، کتابخونه‌ها و الگوهای طراحی‌ست. ما نیازی نداریم خودمون رو درگیر پیچیدگی و محدودیت‌های اون کنیم. ما می‌تونیم همون مفاهیم رو، با سادگی و انعطاف بالا، در قالب Spring ارائه بدیم."

یعنی چی؟ یعنی Spring اومد بیشترِ نیازهایی که Java EE با کلی پیچیدگی حل می‌کرد رو با استفاده از:

- Design Patterns (الگوهای طراحی)
- و ابزارهای ساده ولی قدرتمند بازطراحی کرد، ساده‌سازی کرد، و ماژولار و قابل تست کرد.

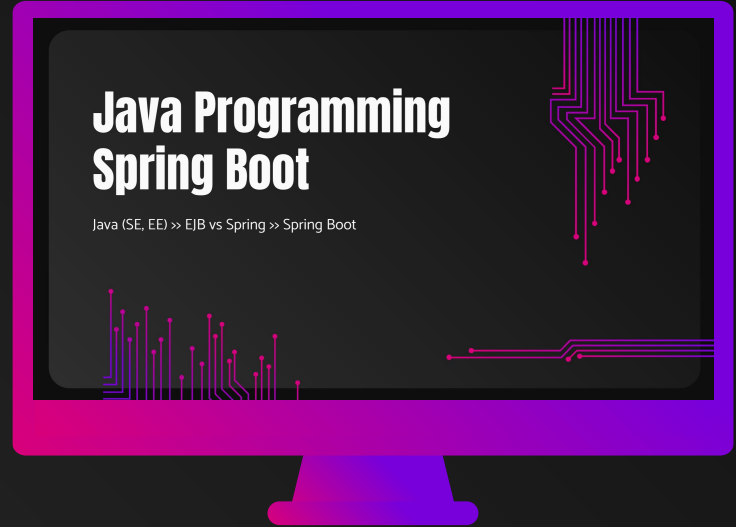
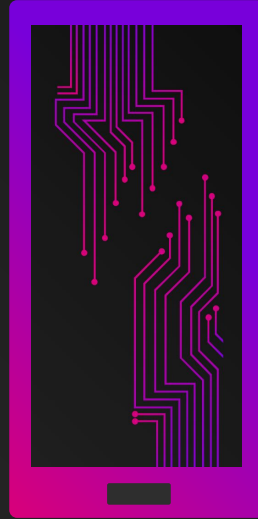
# الگوهای طراحی و ابزارهایی که Spring استفاده کرد

ابزار در Spring	Design Pattern	حوزه
Spring Core / Context	<b>Dependency Injection</b>	Dependency Management
Spring Beans	<b>Service Layer Pattern</b>	Business Logic و Service لایه
Spring Data	<b>DAO Pattern</b>	مدیریت داده / دیتابیس
Spring Security	<b>Interceptor Pattern</b>	امنیت
Spring Boot	<b>Strategy Pattern</b>	پیکر بندی



# First Exercise

Spring boot helloworld  
Push to github



The slide features a dark gray background with decorative circuit-like lines in purple and pink. These lines are arranged in a symmetrical, branching pattern along the left and right edges, resembling a stylized circuit board or data flow. The lines vary in thickness and end in small dots.

# Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**