



# Spring Framework

Ocak 2023

**Miraç**  
Instructor



# J2EE, JAVAEE, JAKARTAEE

Enterprise applications: J2EE, (1999)

J2EE -> JavaEE (2006)

JavaEE -> JakartaEE (2018)

EJB(Enterprise Java Bean)

EJB development için aşırı zor vir yapıya sahipdi



# J2EE, JAVAEE, JAKARTAE



Rod Johnson (2002):

~

Book: J2EE Development without EJB

~

Book: Java Development with the Spring Framework



Server tarafında development daha kolay ve hızlı

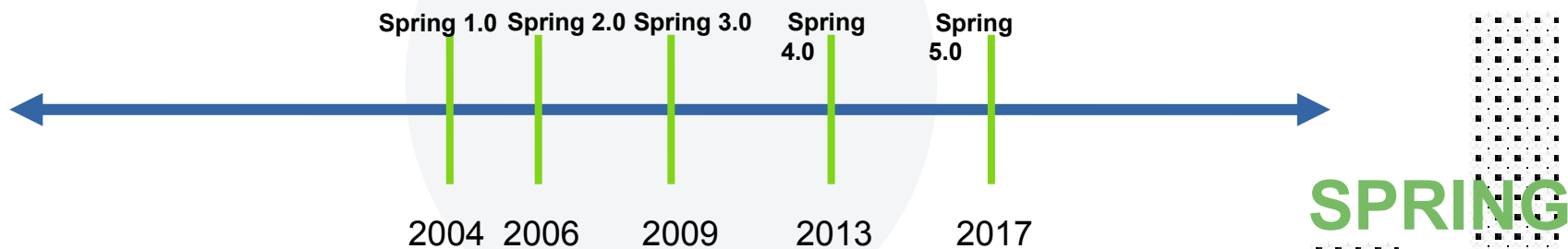
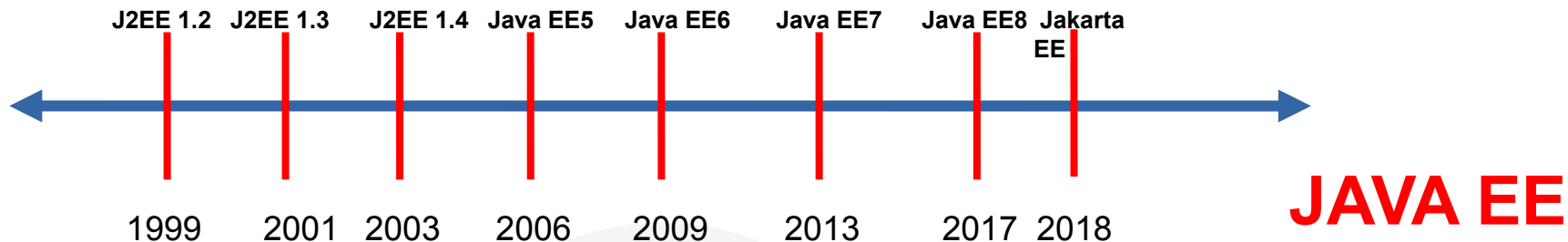


1.0 version release: (2004)



Spring JavaEE için açık kaynak bir Frameworkdür

# JAVA EE ve SPRING




# SPRING FRAMEWORK

**Spring kurumsal uygulamalar geliřtirmek için kullanılan en popüler frameworklerden birisidir.**

**Bağımlılık enjeksiyonu tasarım kalıbı üzerine oturtulmuş “Spring framework” çekirdek yapısı bize birbirinden bağımsız sistemler inşa etmemizi sağlar.**



# SPRING FRAMEWORK

- 🎬 Java POJOs. (Plain- Old – Java- Objects)
  - 🎬 boilerplate Java codeları sayısı oldukça azdır.
  - 🎬 Loose coupling için Dependency Injection mekanizmasını kullanır
- 

# INVERSION OF CONTROL



**Programlama prensibidir**



**IOC kontrolü developerdan Frameworke alır...**

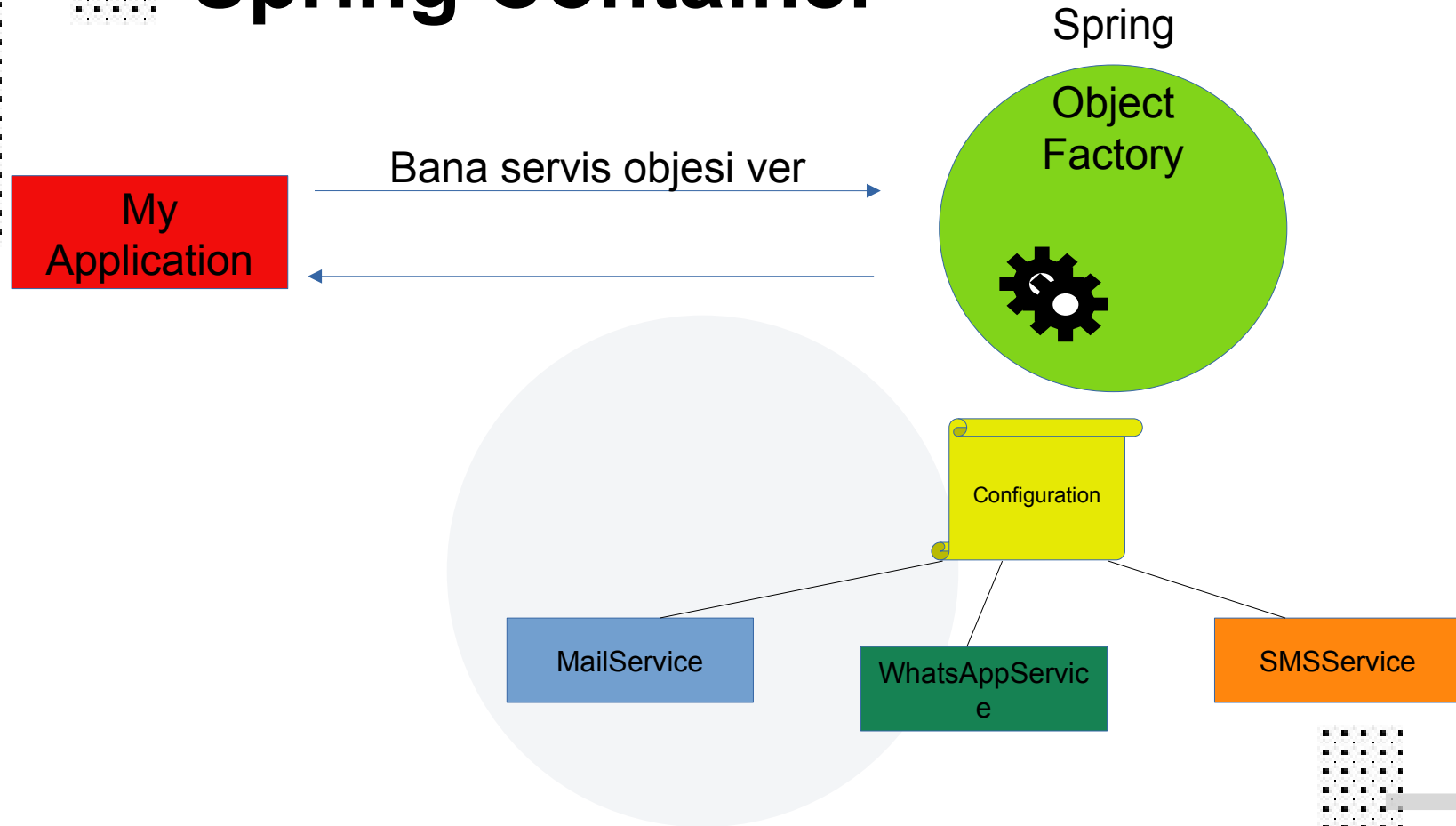


**“New” leme tarzı işlemler bellek yönetimi açısından**



**maliyetli olduğu için obje üretiminin şeklini değiştirir**

# Spring Container





# Spring Container



## Temel Fonksiyonları

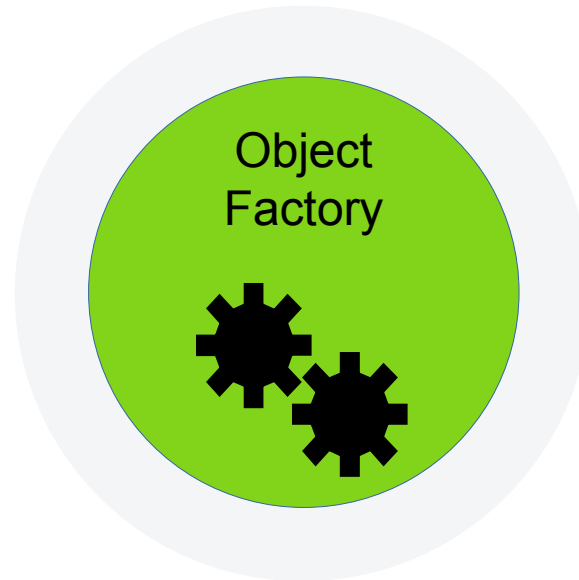
~

\*

Objeleri üretmek ve yönetmek (Inversion of control (IoC))


\*

Bağımlılıkları Objeye enjekte etmek (Dependency Injection (DI))





# Spring Bean Nedir ?

- 1) Bean Nedir ? : basit Java Objeleridir.
  - 2) Spring Container, Java objeleri ürettiği zaman
  - 3) onları “Spring Beans” diye adlandırır.
  - 4) Spring Beans’ler normal Java Class’lardan üretilir.
- 

# Configuring Spring Container

- XML Based (eskidir ve yapılandırması zordur )
- Annotation Based (Spring 2.5)
  - ~ XML configurasyonları minimum düzeyde kullanılır
  - ~ ama “component scan” yapısı xml ile yazılır
- Code Based (Spring 3.2)
  - ~ java configuration class kullanır
  - ~ Bütün konfigürasyonlar kod içinde yapılır

# Spring Annotations?

- @Configuration --> Konfigürasyon için kullanılır
- @Component
  - @Bean
- @ComponentScan --> Component leri taramaya başlar
- @Autowired --> Bağımlılık Enjekte etmek için kullanılır
- @Qualifier --> Seçim yapıcısıdır

# Örnek Proje

Adım1: Java Class oluştur ve @Configuration ekle

Adım2: component scanning için @ComponentScan ekle

Bunu yazmazsanız proje içindeki dosyaları otomatik tarar

Adım3 : Class lara @Component annotation ekle

Adım4: Spring Java configuration class'ını oku

Adım5: Spring Container dan bean talep et



# Injection Tipleri

1) Constructor injection

2) Setter Injection

3) Field Injection



# Bean Scopes

1) Scope = bean ın yaşam döngüsü

2) Tanımladığı alanlar :

- ~ Ne kadar yaşıyacak ??
- ~ Kaç tane instance üretilecek ??
- ~ Nasıl paylaşılacak ??

# Bean Scopes (Singleton)

- 1) Bütün beanler default olarak singleton üretilir
- 2) Spring IoC Container her beandan sadece 1 tane
- 3) instance oluşturur.
- 4) Beans'ler memory de tutulur

```
MessageService service1=  
context.getBean("mailService",MessageService.class);
```

```
MessageService service2=  
context.getBean("mailService",MessageService.class);
```

Aynı Obje



# Bean Scopes ( Singleton )

```
@Component
```

```
@Scope("singleton") //default scope
```

```
public class MailService implements MessageService {  
  
}
```

# Bean Scopes ( Prototype )

- New object is created for each request

```
@Component
@Scope("prototype")
public class MailService implements MessageService {
}
```

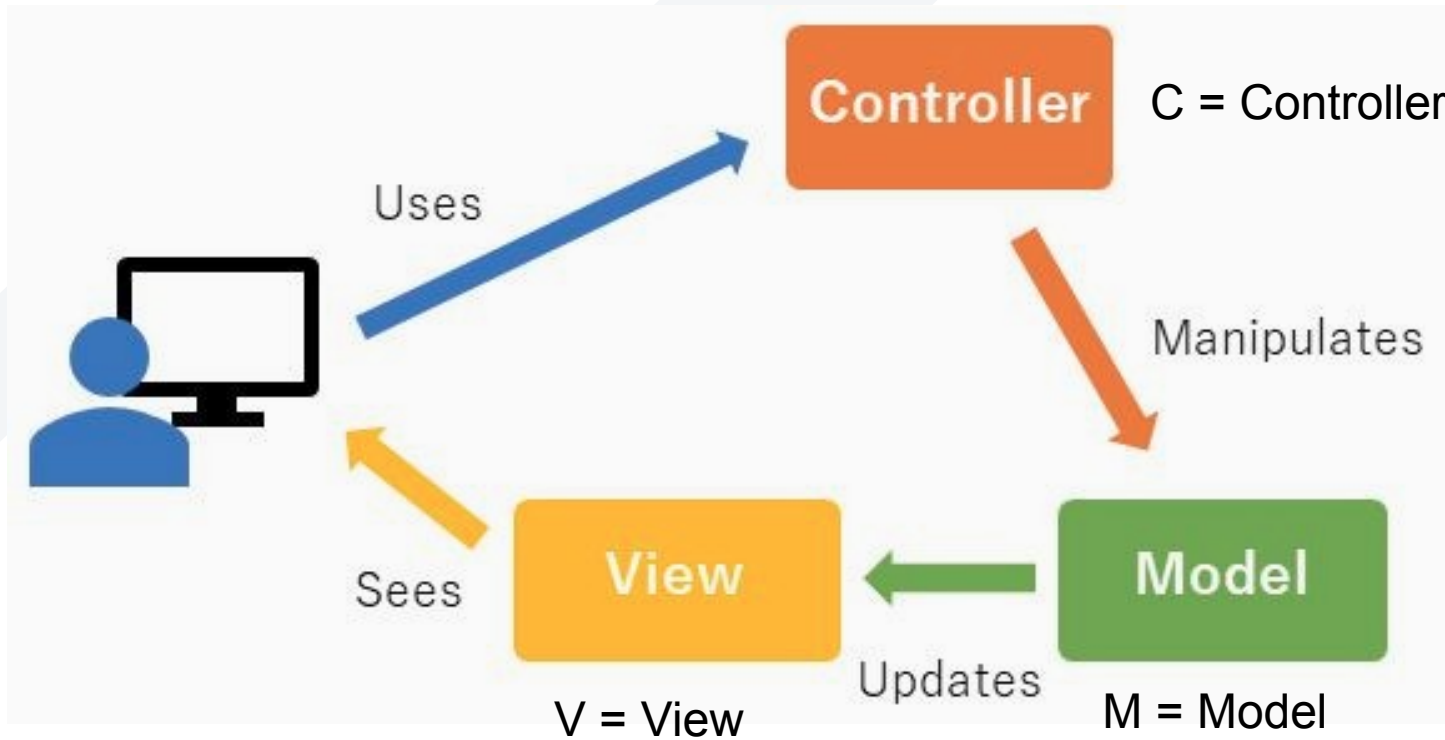
```
MessageService service1=
context.getBean("mailService",MessageService.class); —————> Reference1
```

```
MessageService service2=
context.getBean("mailService",MessageService.class); —————> Reference2
```

# Spring MVC

- ~ Java ile web uygulamaları yapmak için bir Framework'dür
- ~ Model-View-Controller design pattern üzerine inşa edilmiştir
- ~ Dynamic Web Application oluşturulabilir
- ~ RESTFUL service üretmek için kullanılabilir
- ~ Core Spring Framework(IOC, DI) ün getirdiğin kolaylığı sağlar

# Spring M-V-C



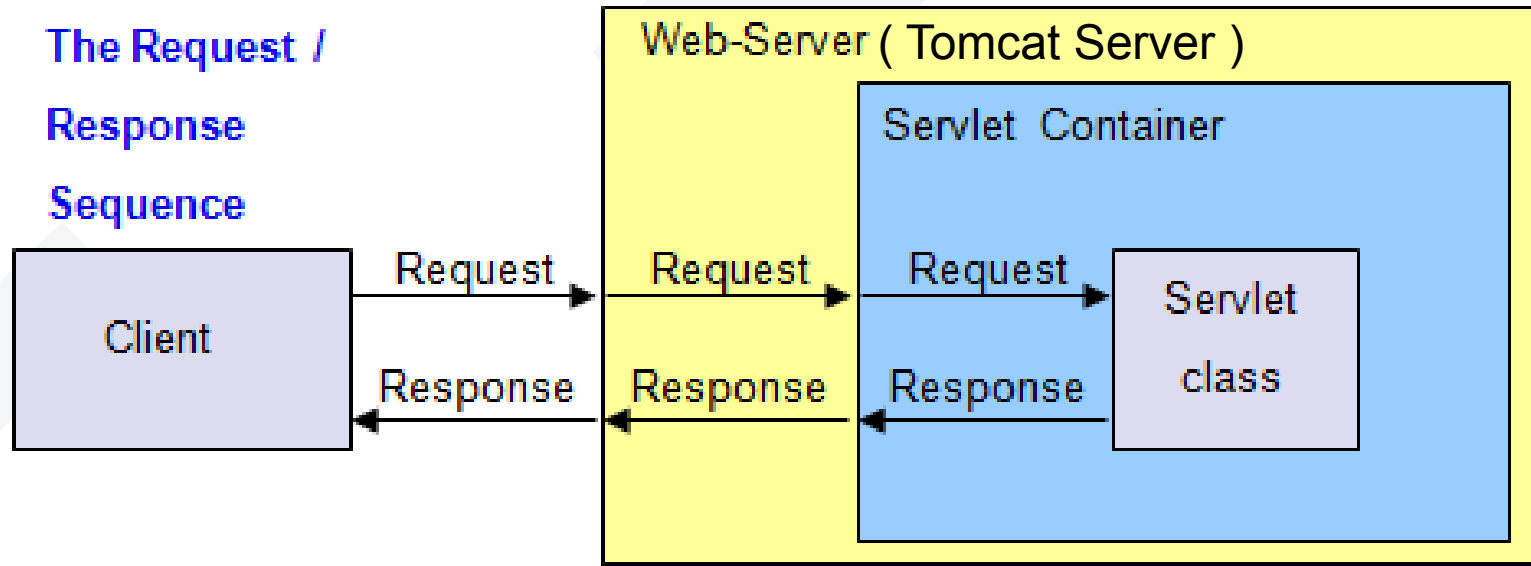
# Spring MVC Configurations

- Spring MVC supports 2 farklı konfigürasyonu destekliyor
  - ✓ XML Based Configuration
  - ✓ Java Based Configuration

# Önemli Annotation'lar

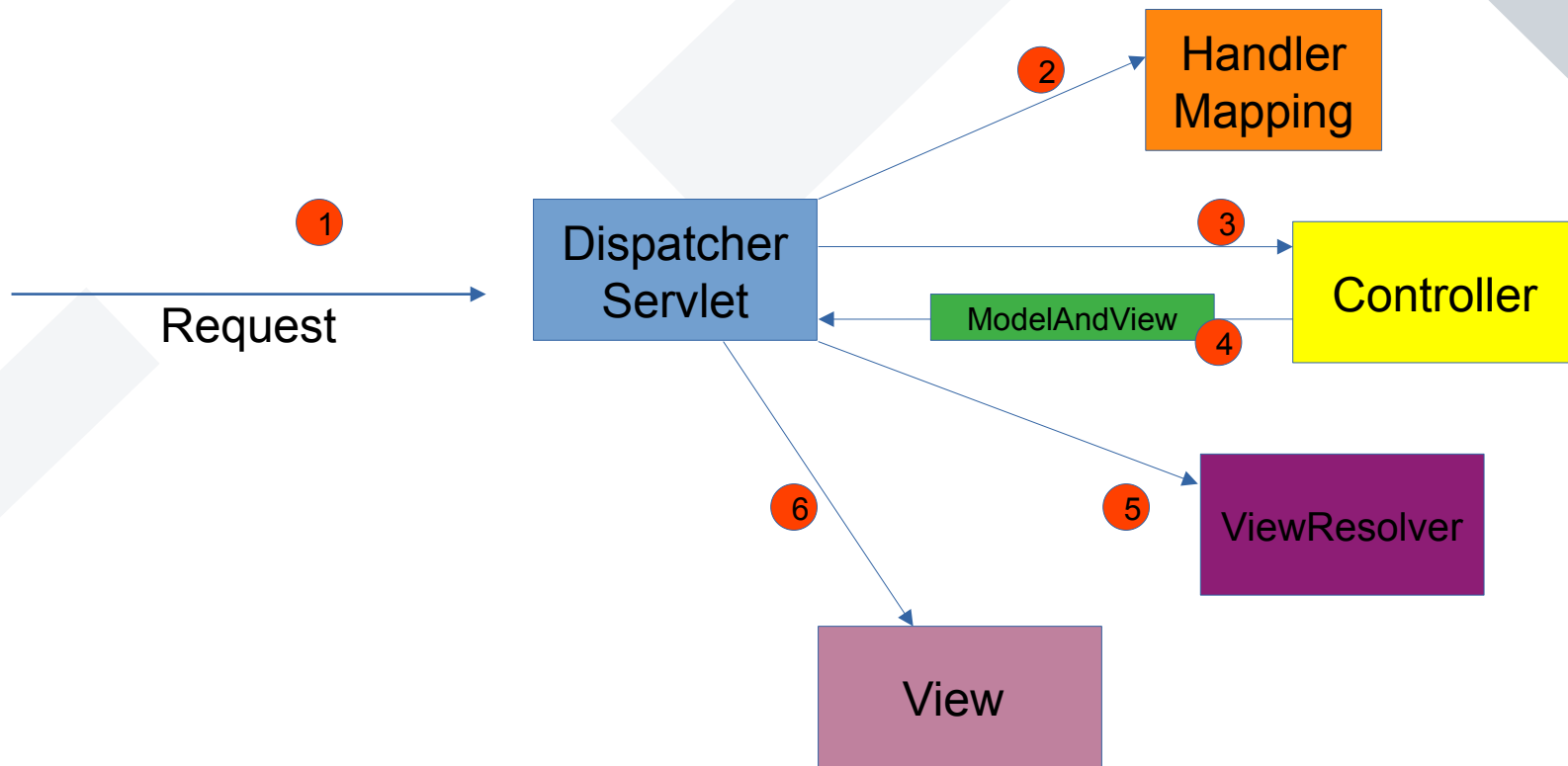
- @Controller and @RestController
- @RequestMapping
  - @GetMapping, @PostMapping
  - @PutMapping, @DeleteMapping
- @RequestParam, @PathVariable
- @RequestBody
- @ResponseBody

# WebServer - Servlet



**Servlet** = Gelen HTTP request'leri, uygun şekilde cevaplanılmasını sağlayan Java Sınıfıdır.

# Spring MVC





# Spring MVC - URL

- HTTP URL
  - `http://serverAddress:portNumber/resourcePath?queryString`

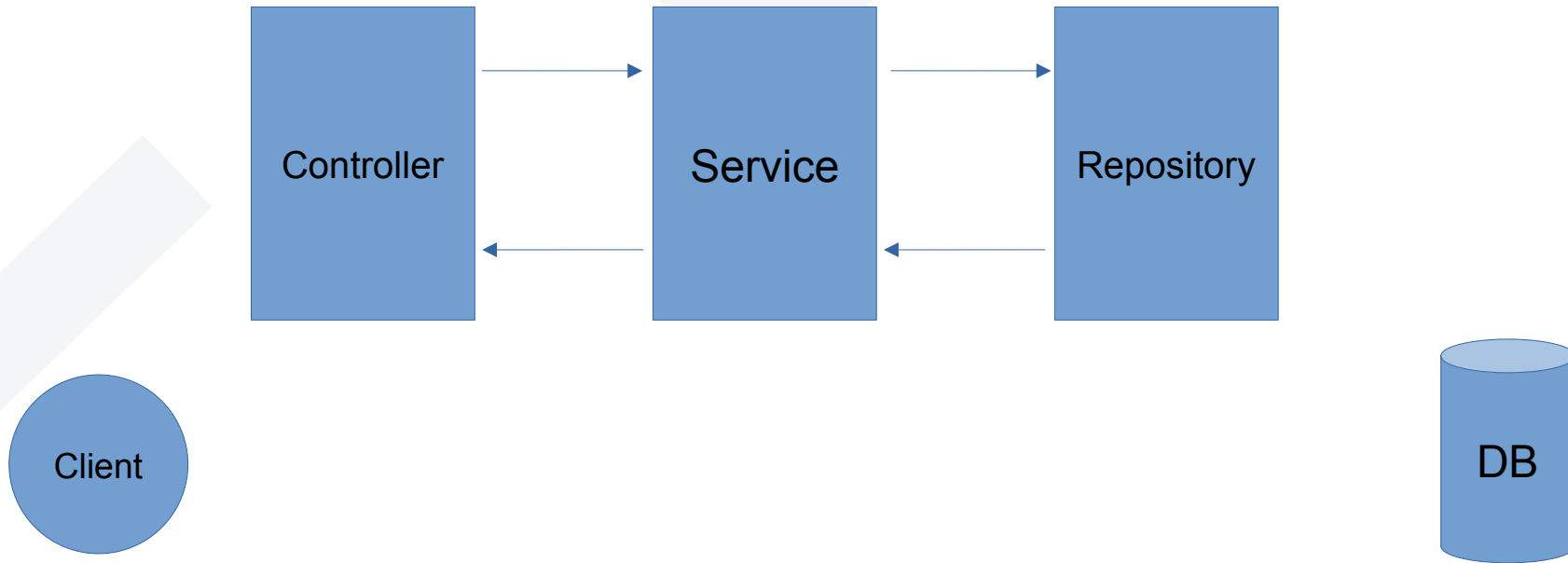
`https://127.0.0.1:8080/customer?id=1`

The diagram illustrates the components of the URL `https://127.0.0.1:8080/customer?id=1` using arrows and labels:

- `https` points to **protocol**
- `127.0.0.1` points to **server address**
- `:8080` points to **Port number**
- `/customer` points to **Resource path**
- `?id=1` points to **Query param**

Örnek = `https://127.0.0.1:8080/data/2.5/weather?q=London,uk&appid=2b1fd2d7f77ccf1b7de9b441571b39b8`

# 3-Layer Architecture



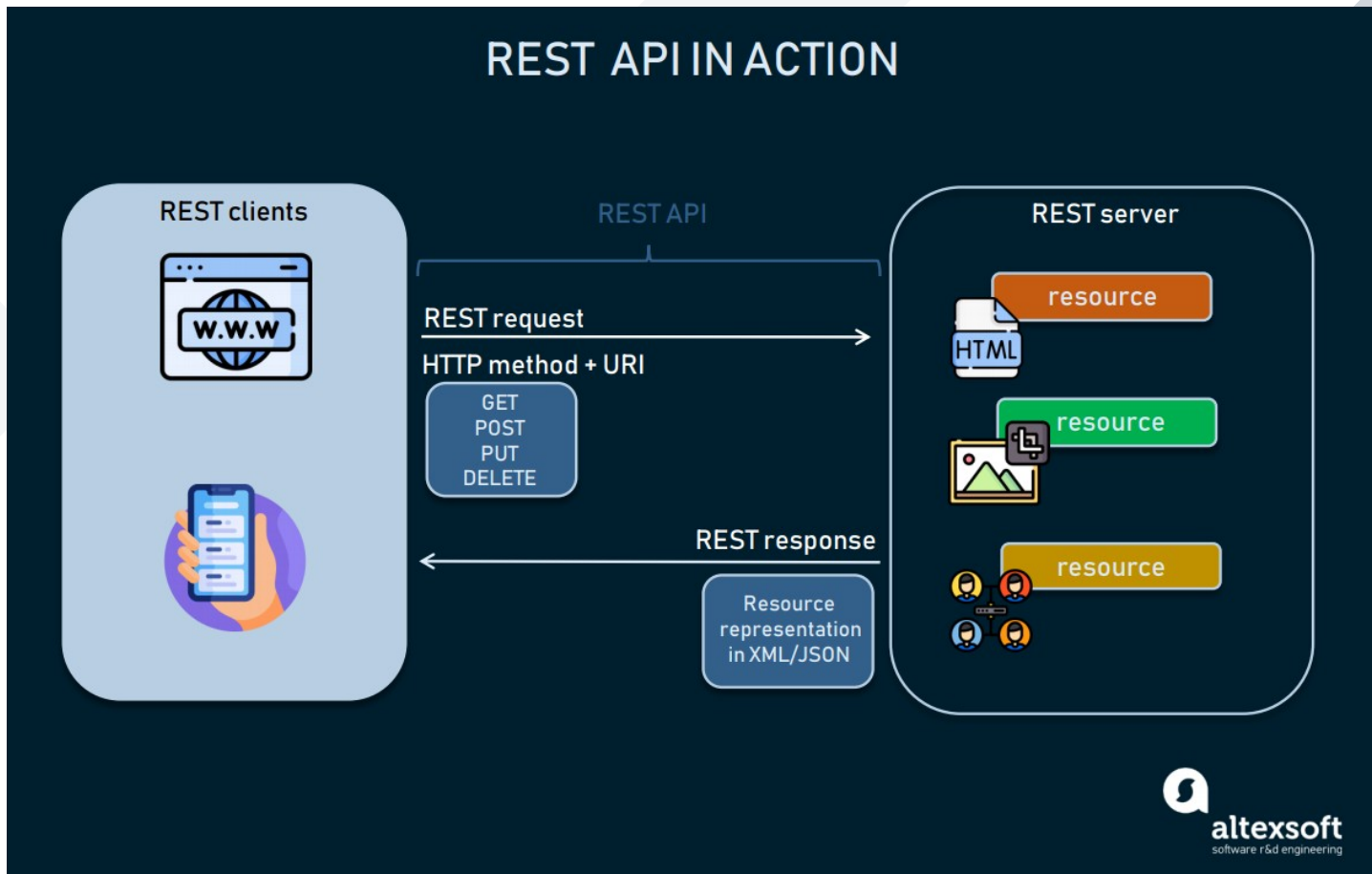
# REST nedir ??

- Stateless
- Server tarafındaki kaynaklara URI ile ulaşılır

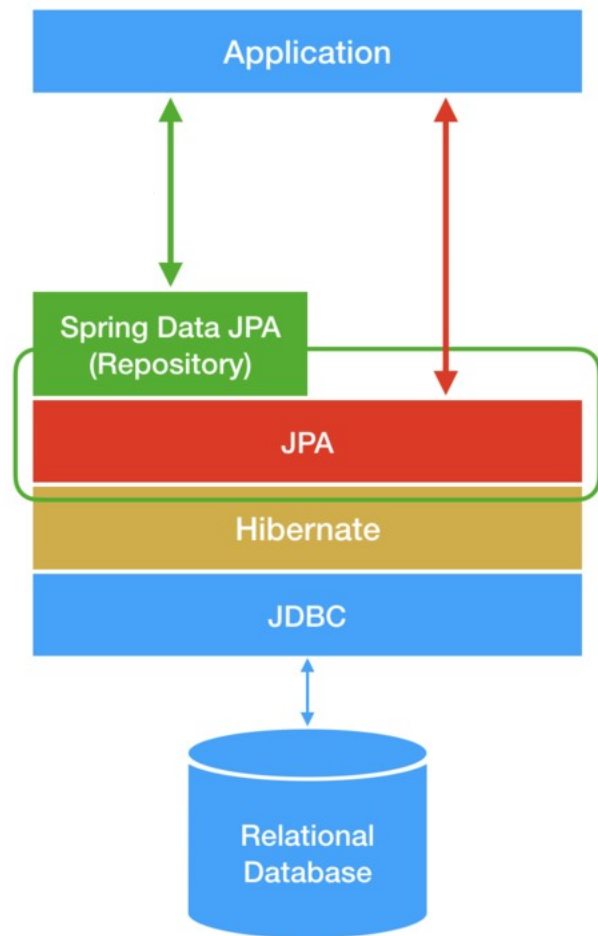
`https://127.0.0.1:8080/customer?id=1`

- Session yoktur, bilgi taşınmaz
- HTTP protokollerini kullanır
- CRUD operasyonları için ,HTTP methodları kullanılır; get, post, put and delete
- Response olarak bilinen formatlar kullanılır JSON veya XML, vb.

# REST nedir ??



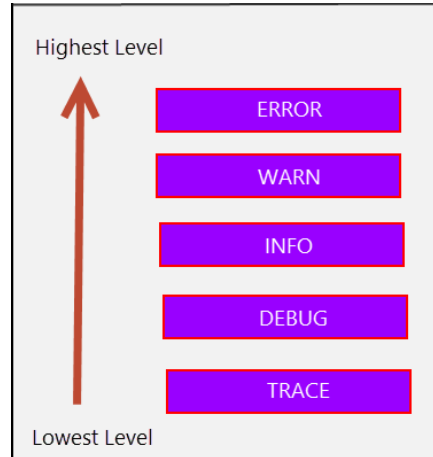
# Spring Data JPA nedir ??



# Log İşlemleri

- **Loglama Seviyeleri**

- Trace : Uygulamanın davranışı ile ilgili bütün bilgileri görmemizi sağlar
- Debug : Debug seviyesindeki bilgileri gösterir.
- Info : Systemimizin anlık olarak ne yaptığı hakkında bilgi verir
- Warn : Kritik olmayan ama aplikasyonumuzu etkileyen uyarıları gösterir
- Error : Kritik seviyedeki uyarıları gösterir



# Spring Boot **Actuator**

- **Monitörleme işlemlerimi yapmamı sağlar**

Base path: /actuator

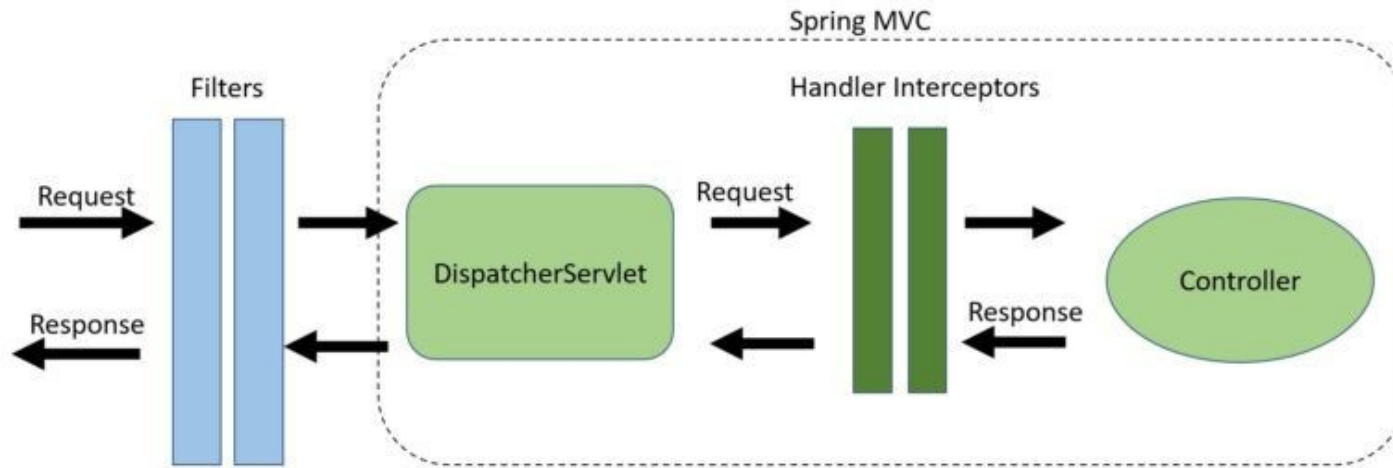
/info	: Uygulama bilgisi
/health	: Query kontrolü
/threaddump	: Thread dump durumu kontrolü
/trace	: En son girilen 100 HTTP requests
/beans	: Oluşturulan Spring Bean'ler
/loggers	: <i>Oluşturulan Logger Objeleri</i>
/env	: <i>enviroment değişkenleri</i>

# Spring Security

- Enterprise uygulamalar temelde 2 seviyede güvenlik
- mekanizmasına ihtiyaç duyarlar:
  - Authentication
    - Kimsin ??
  - Authorization
    - Yetkin var mı ???
- Spring Security is a framework for security.



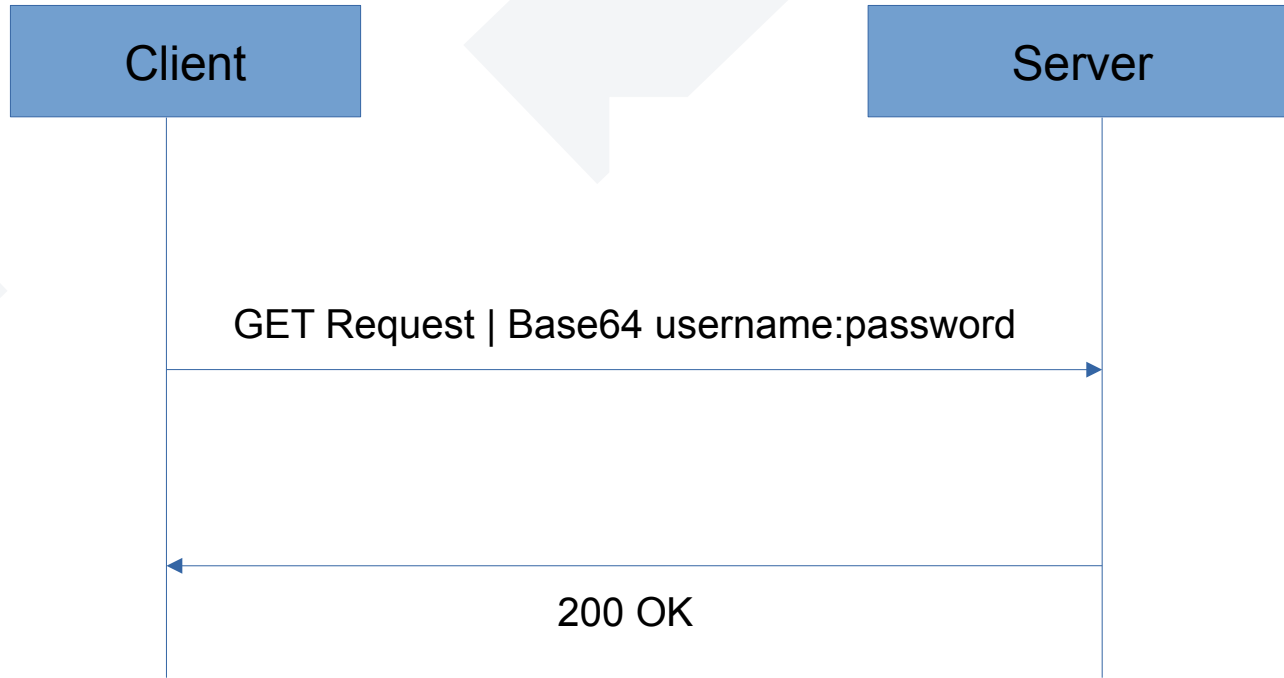
# Spring Security



# Spring **Security**

- Birden fazla Authentication Methodlarını destekler
  - Form Based Authentication
  - Basic Authentication
  - .
  - .
  - **JWT Based Authentication (JSON WEB TOKEN)**

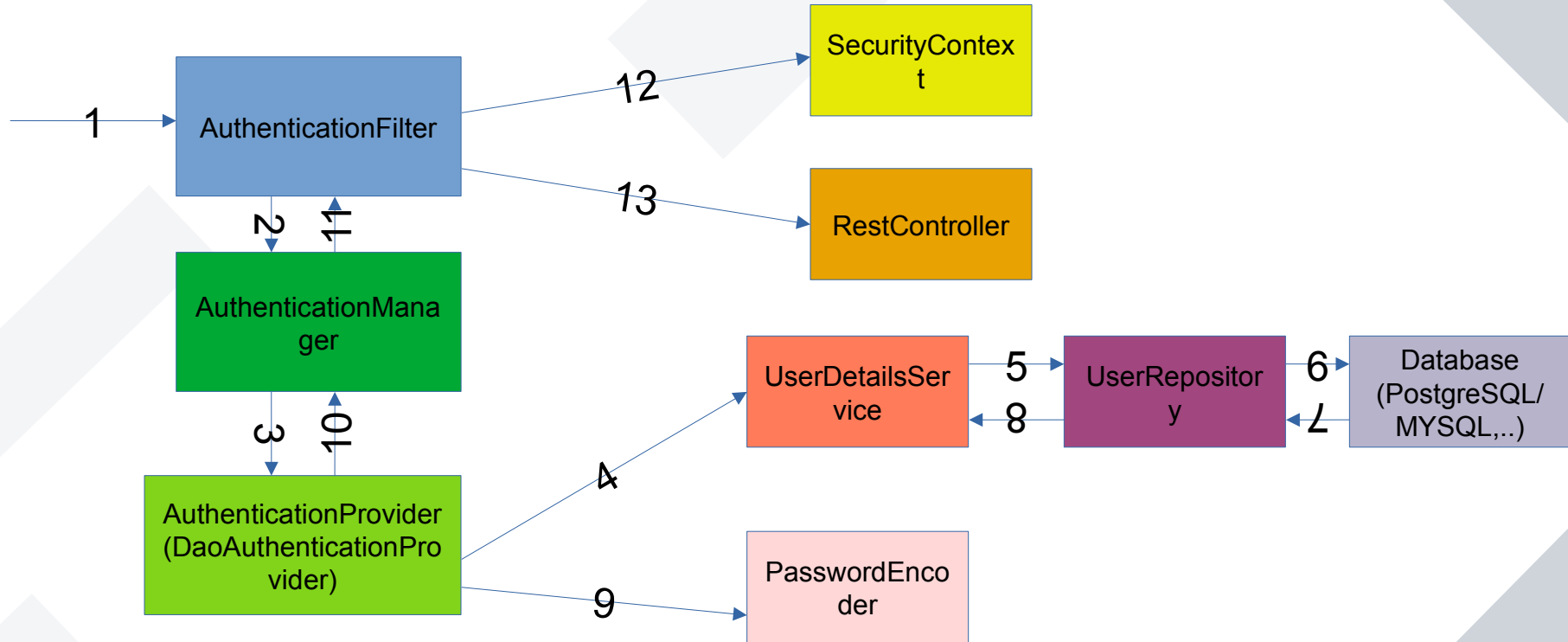
# Spring Security – Basic Auth



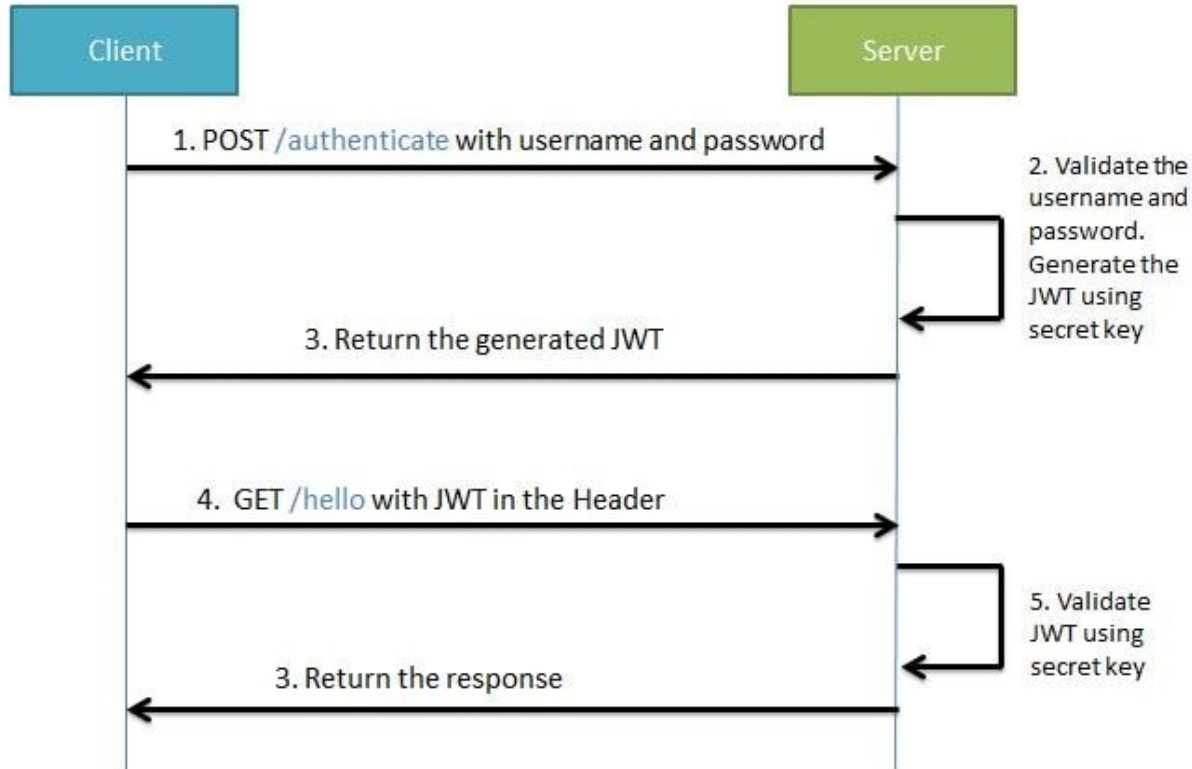
\*  
\*

Her requestde Kullanıcı adı ve şifre bilgisi gönderilmeli  
Kullanıcı adı ve şifre bilgisi Base64 ile encode edilir

# Spring Security – Basic Auth



# Spring Security – JWT Based



**iyi alıřmalar ...**

