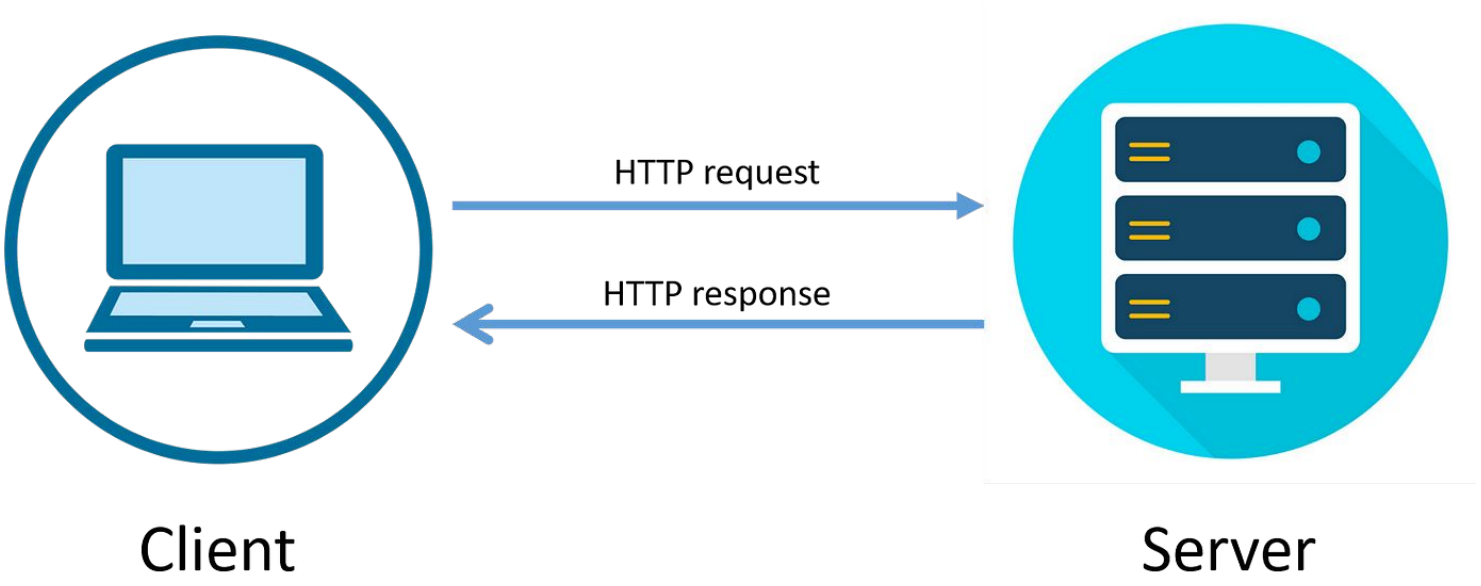
Web Tabanlı Server/Client İletişimi
Java Servlet

Dr.Öğr. Üyesi Ahmet KARADOĞAN

HTTP

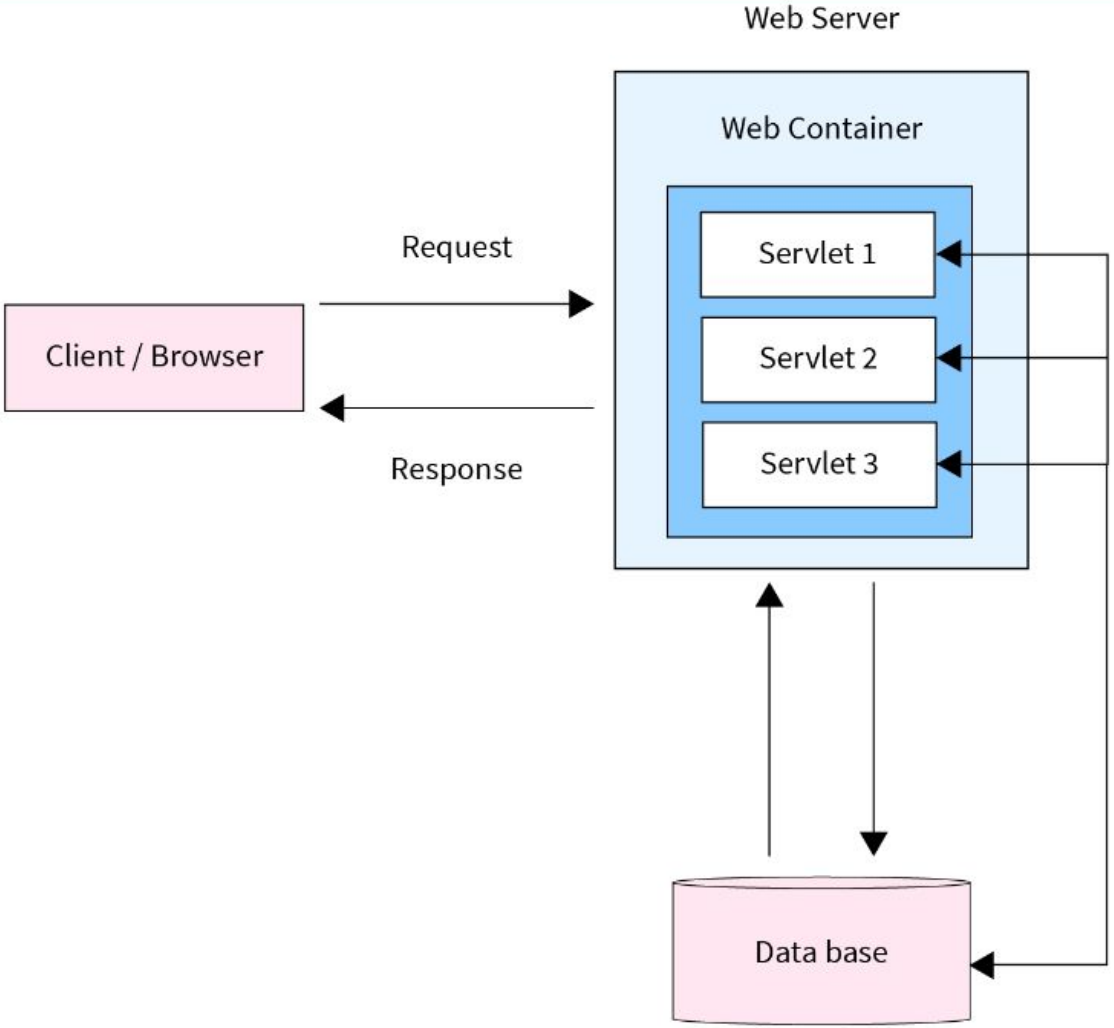
- TCP/IP mimarisinde İnternet üzerinden sunucu-istemci arasındaki veri iletimi için kullanılan protokol



Servlet

- Sunucuya gelen Http isteklerini yönetmek için kullanılan bir java sınıfıdır
- Web tabanlı ağ programlamada sunucu tarafında kullanılan bir java programıdır
- Dinamik web sayfaları oluşturmak için kullanılır
- Clienttan gelen istekleri alıp bu isteğe karşılık cevap olarak bir HTML sayfası oluşturup clienta iletir
- Java EE(Enterprise Edition) içinde bulunur.

Servlet


Web Server
Web Container
Servlet 1
Servlet 2
Servlet 3
Request
Client / Browser
Response
Data base

Servlet Container (Web Container)

- Sunucu tarafında, servlet'leri yönetmekten sorumlu olan bir Servlet Container bulunur. Servletlerin yaşam döngüsünü yönetir.
- İstemci sürecinden gelen istekler ilk olarak servlet container tarafından alınır ve ardından ilgili servlete yönlendirilir. Servlet, isteği aldıktan sonra, gerekli işlemleri gerçekleştirir (veri tabanı veya dosya sistemlerine erişmek vb.) ve sonuçları bir yanıt olarak istemciye gönderir.

Servlet Yaşam Döngüsü

- Bir servlet'in oluşturulmasından sonlandırılmasına kadar olan süreçte geçen adımlar:

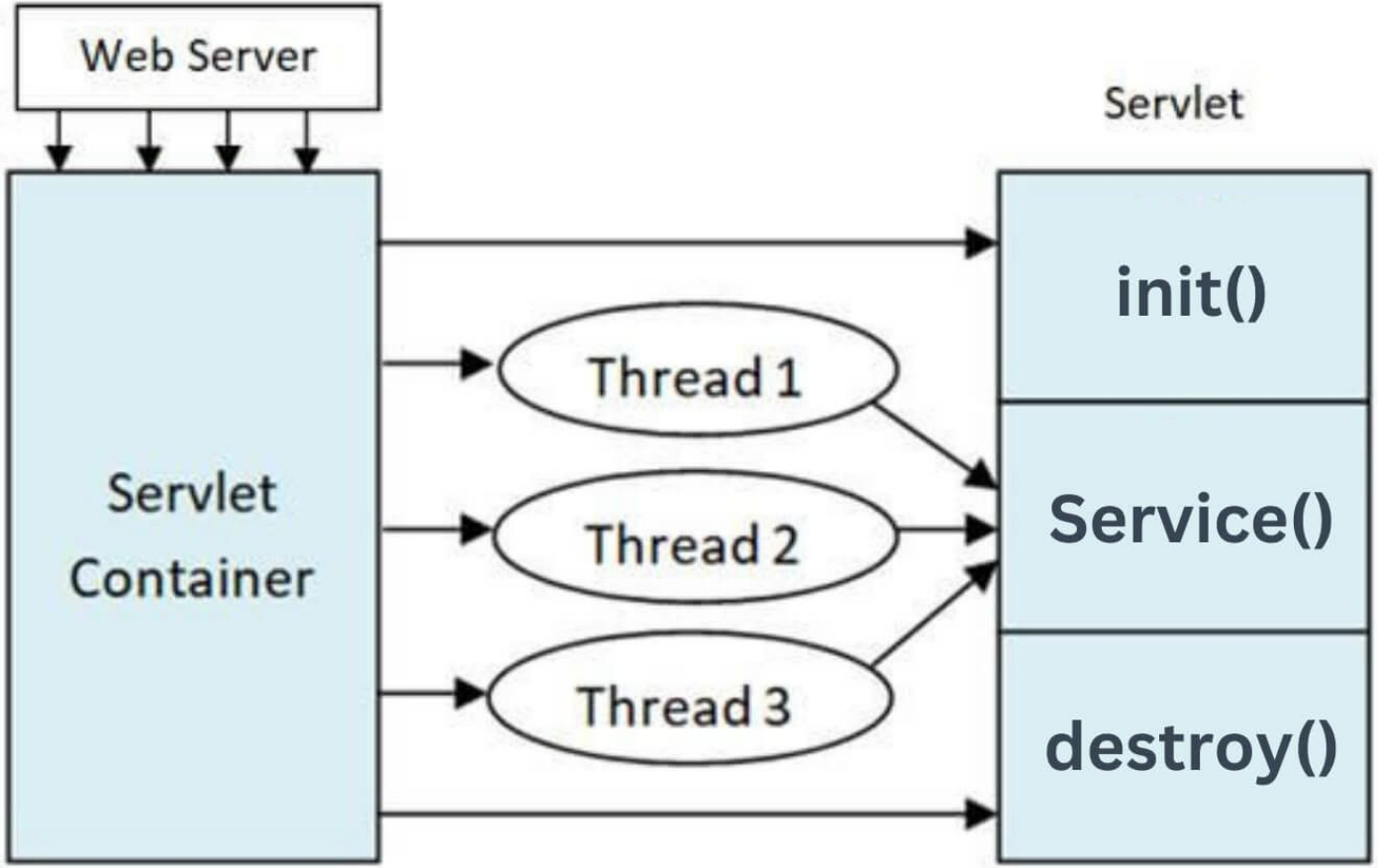
  1. **Yükleme** (Loading): Servlet, ilk kez talep edildiğinde veya sunucu başlatıldığında yüklenir ve servlet nesnesi oluşturulur.

  2. **Başlatma** (Initialization): Servlet nesnesi oluşturulduktan sonra ***init()*** metodu çağrılır. Bu yöntem, servlet'in başlatılmasında gerekli olan tüm başlangıç işlemlerini yapar. Bu sadece bir kez çağrılır. Aynı servlet'e birden fazla istek geldiğinde her biri için bir thread oluşturulur.

Servlet Yaşam Döngüsü

3. **İstek İşleme** (Request Handling): Her gelen istek için ***service()*** metodu tetiklenir. ***service()*** metodu, HTTP isteklerini (GET, POST, PUT, DELETE vb.) işlemek için uygun yöntemi (*doGet(), doPost(), doPut(), doDelete()*) çağırır.
4. **Sonlandırma** (Termination): Servlet'in yaşam döngüsü sona erdiğinde, ***destroy()*** metodu çağrılır. Bu yöntem, servlet'in kapatılması sırasında yapılması gereken temizlik işlemlerini içerir. Sunucu kapatıldığında veya servlet belli bir süre kullanılmadığında destroy() metodu çağrılır.

Servlet Yaşam Döngüsü



Servlet Programlama

- Servletin çalışabilmesi için öncelikle sunucuda servleti destekleyen bir web server programı kurulmalıdır (Apache Tomcat, Glassfish vb.)
- 2019 yılından itibaren Java EE sürümü Oracle tarafından Eclipse firmasına devredilip Java EE yerine Jakarta EE adını aldı. Aynı şekilde Java servlet sınıfı da yeni sürümlerde Jakarta servlet sınıfı olarak kullanıldı. Sınıf kütüphaneleri artık ***javax*** yerine ***jakarta*** ile başlamaktadır.
- Tomcat server v9 ve öncesi javax desteklerken son sürüm v10 jakartayı desteklemektedir.

Servlet Programlama

1. Tomcat v10 (zip veya Windows installer) indirilip kurulumu yapılır. *https://tomcat.apache.org/*

   Tomcatin kurulu olduğu klasördeki alt klasörler:

   bin: İkili dosyaları ve scriptleri içerir (örneğin, Windows için startup.bat ve shutdown.bat; Unix ve macOS için startup.sh ve shutdown.sh).

   conf: server.xml, web.xml ve context.xml gibi sistem geneli yapılandırma dosyalarını içerir.

   webapps: Yayınlanacak web uygulamalarını içerir. Dağıtım için WAR (Webapp Archive) dosyasını da buraya yerleştirebilirsiniz.

   lib: Tüm web uygulamaları tarafından erişilebilen Tomcat'in sistem geneli kütüphane JAR dosyalarını içerir. Ayrıca, MySQL JDBC Sürücüsü gibi harici JAR dosyasını da buraya yerleştirebilirsiniz.

   logs: Tomcat'in günlük dosyalarını içerir. Hata mesajları için bu dosyaları kontrol etmeniz gerekebilir.

   work: JSP'den Servlet'e dönüştürme işlemi için Tomcat'in çalışma dizini olarak kullanılır

Servlet Programlama

2. JAVA_HOME ortam değişkeni eklenip JDK'nın kurulu olduğu klasör yolu verilir (daha önce eklenmediyse). Komut satırından "set JAVA_HOME" komutu ile kontrol edilebilir.

3. Tomcat server ayarları: Tomcat dizinindeki conf klasöründe bulunan xml dosyaları güncellenir:
   a. server.xml: Web sunucunun çalışacağı port numarası değiştirmek için "Connector port="8080" " satırındaki 8080 portuna 1024-65.535 arası herhangi bir port numarası belirlenir
   b. web.xml : Varsayılan dosyalar bulunamadığında "404 Not Found" hatası vermeyip dizindeki dosyaları listelemesi için <servlet> tagı bölümünde <init-param> altındaki <param-value> değeri true yapılır.

Servlet Programlama

4. Tomcat başlatmak için bin klasöründeki “startup.bat” dosyası komut satırından çalıştırılır: *“C:\WebProjects\apache-tomcat-10.1.24\bin>startup”* (mac için “catalina.sh”)

   Açılan konsol ekranında hatalar varsa gösterilecektir.(servlet kodundaki hata çıktıları da burda görünür)

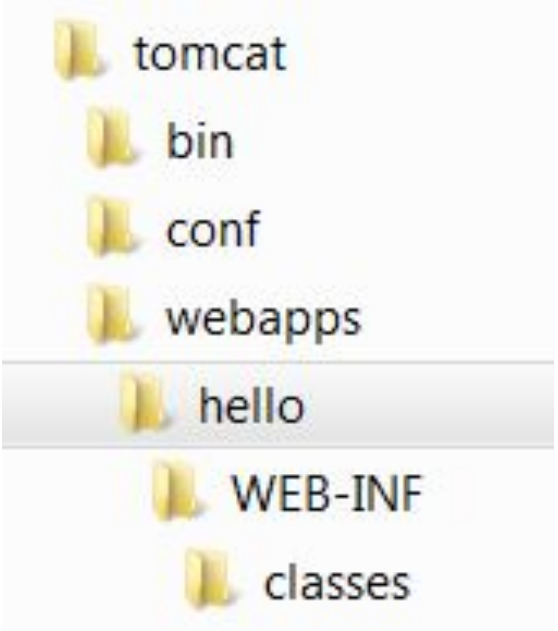
   Serveri kapatmak için ise “*shutdown*” komutu kullanılır.

5. Tarayıcıdan “http://localhost:8080/” adresine girilip Apache Tomcat anasayfası kontrol edilir.

   http://localhost:8080/examples/ adresinden örnekler incelenebilir)

Web Uygulaması Oluşturma

1. Tomcat dizinindeki webapps klasörü altında web uygulaması için klasör oluştur: “hello”
2. Oluşturulan klasör içinde “WEB-INF” klasörü ve WEB-INF içinde de "classes" klasörü oluştur.

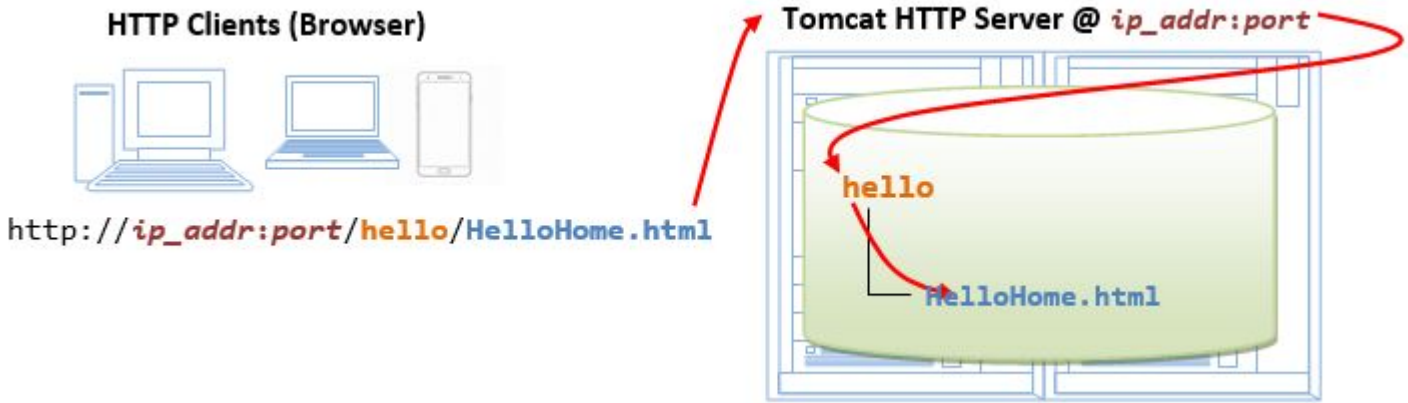


Web Uygulaması Oluşturma

3. “hello” klasörü altında html sayfalarını oluştur: "HelloHome.html"

```
<!DOCTYPE html>
<html>
     <body>
     <h1>Hello world</h1>
     </body>
</html>
```

4. http://localhost:8080/hello/HelloHome.html



Servlet Uygulaması Oluşturma

- Netbeans üzerinden Web Application seçildikten sonra Server kısmında Apache Tomcat dizini seçilip proje oluşturulur.
- Projeye New->Servlet sınıfı seçilip “HelloServlet” eklenir ve import kısmında “javax” olan kısımlar “jakarta” olarak değiştirilir
- Proje çalıştırılıp tarayıcıda açılır “http://localhost:8080/WebApplication/HelloServlet”
- Url’deki servlet adı yerine url-pattern kullanılarak başka bir adres vermek için @WebServlet anotasyonu:

  @WebServlet(urlPatterns = {"/hello"})