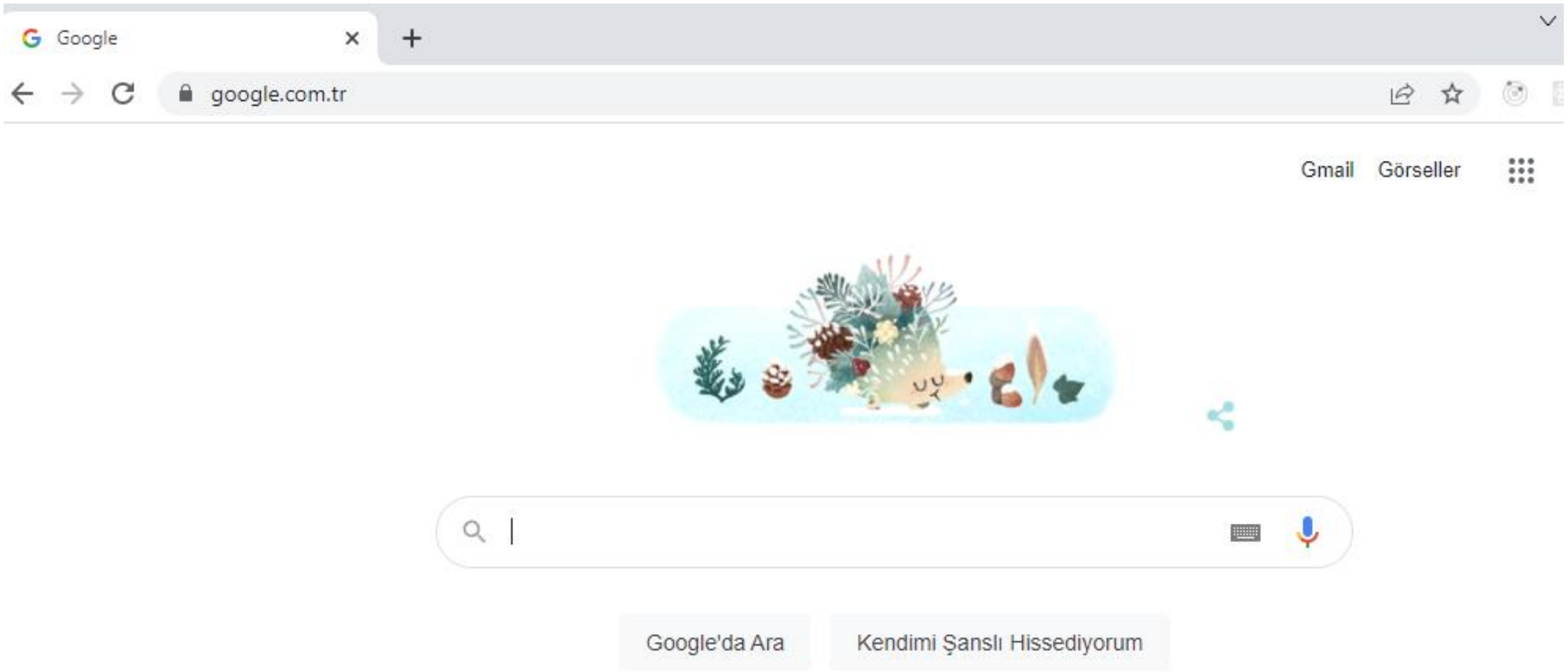
Web Programlama I

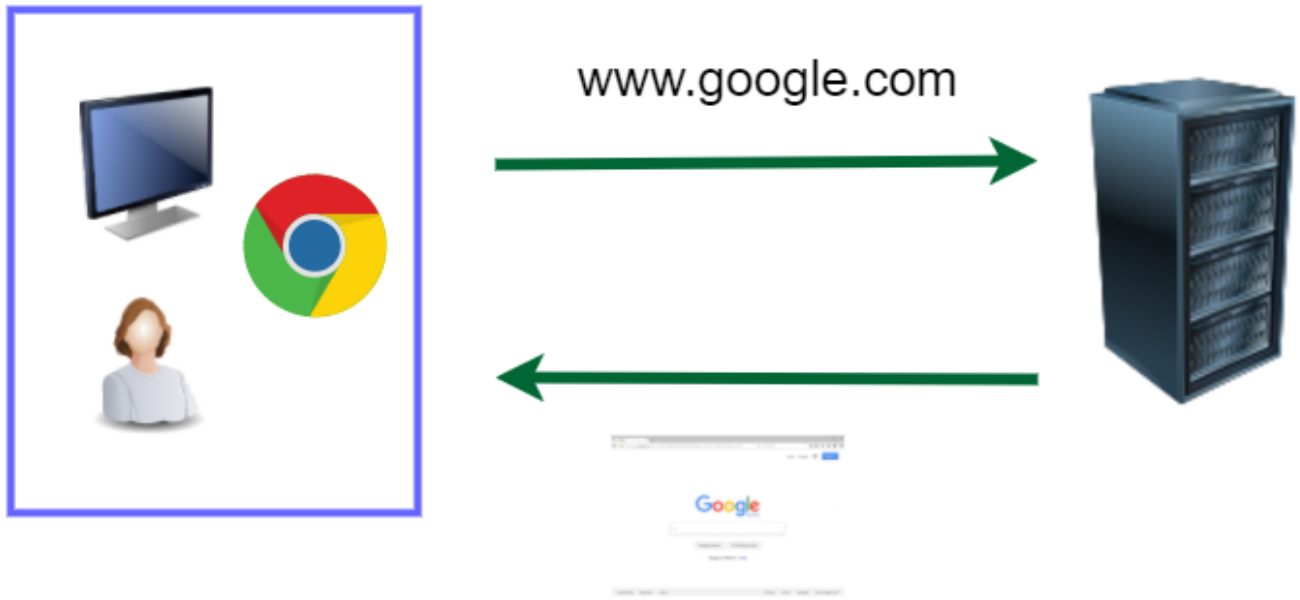
Ders 07: JSF ve Docker

Web Uygulamaları


Google
google.com.tr
Gmail
Görseller
Google'da Ara
Kendimi Şanslı Hissediyorum

World Wide Web (WWW)

- Tim Berners-Lee tarafından 1989 yılında İsviçre, CERN'de icat edilmiştir
- WWW farklı makinalarda dağıtılan doküman/kaynaklara karşılık gelir
- Kaynak/doküman'lar Uniform Resource Locator (URL) ile tanımlanır
- Kaynaklar internet üzerinden erişilebilir/indirilebilir
- Bir web sayfası ise HTML formatında yazılmış bir kaynaktır
- *Tarayıcılar* ise HTML sayfalarını indirip görselleştiren uygulamalardır

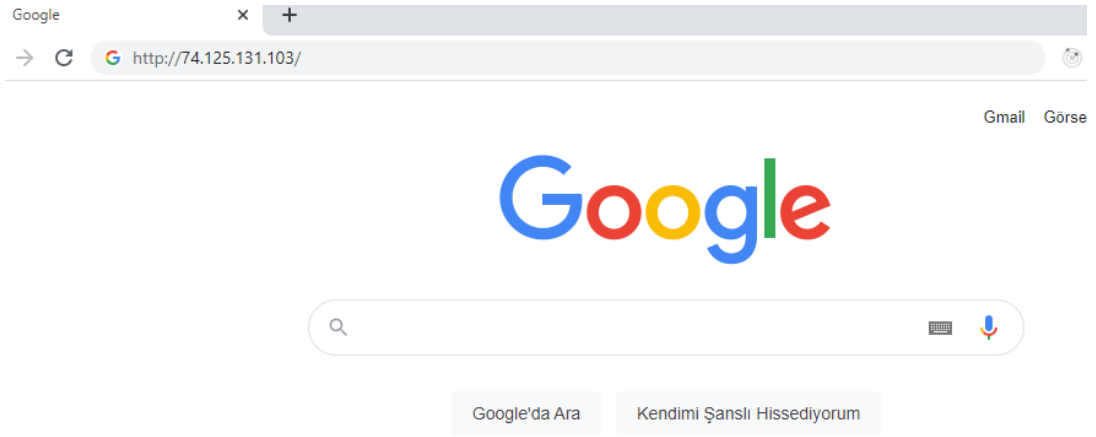


Bir HTTP isteği gönderilir ve gelen HTML sayfası tarayıcı tarafından görselleştirilir

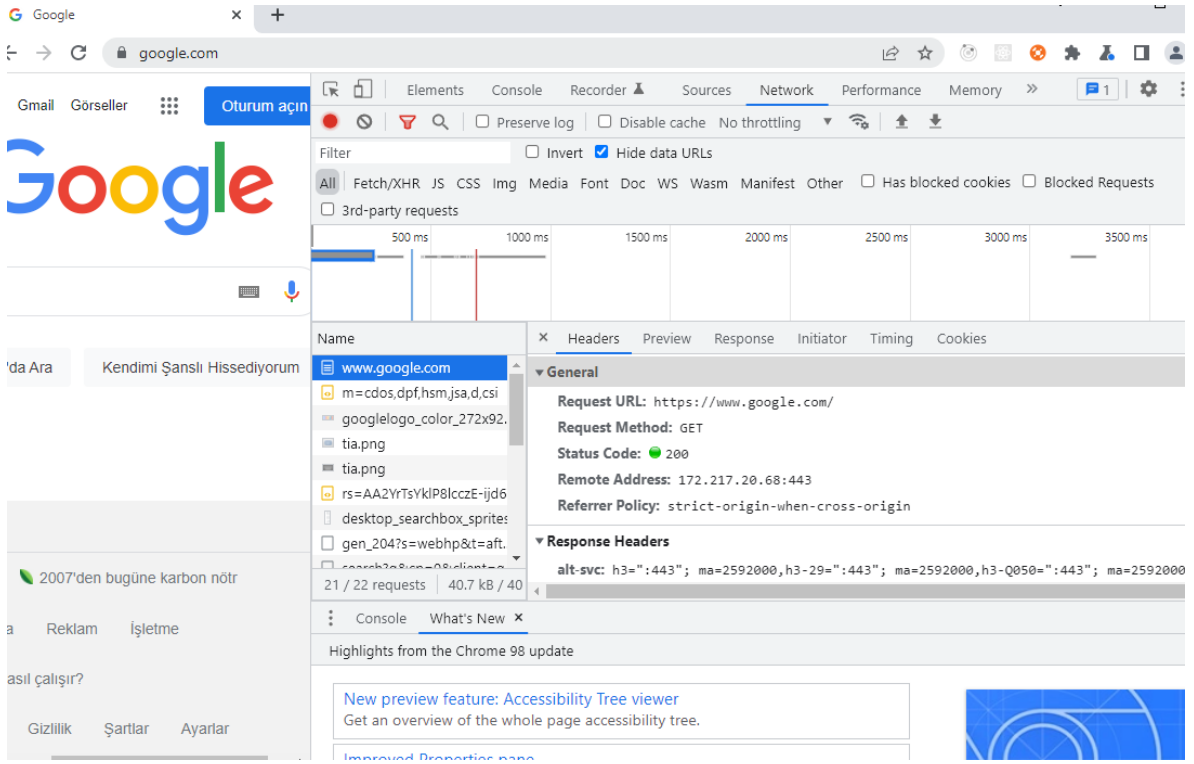
Domain Name System (DNS)

- Tarayıcıya “*www.google.com*” yazdığında bir makinaya nasıl bağlanırız?
- İnternet bağlantıları IP tabanlı adreslerden oluşur
  - 32 bit sayı, 0-255 aralığında oktetlerden oluşur
    - 74.125.131.103 -> www.google.com
  - IPv4 en çok kullanılan versiyonu. IPv6 ise oldukça yavaş bir şekilde yerine geçmektedir
- “*www.google.com*” adresinin 74.125.131.103 IP adresine çevrilmesi için bir DNS sunucusu ile iletişime geçmesi gerekir
- İşletim Sistemi tarafından gerçekleştirilen keşif: bilinen adresler veya internet servis sağlayıcı tarafından alınan sunucular aracılığı ile gerçekleştirilir

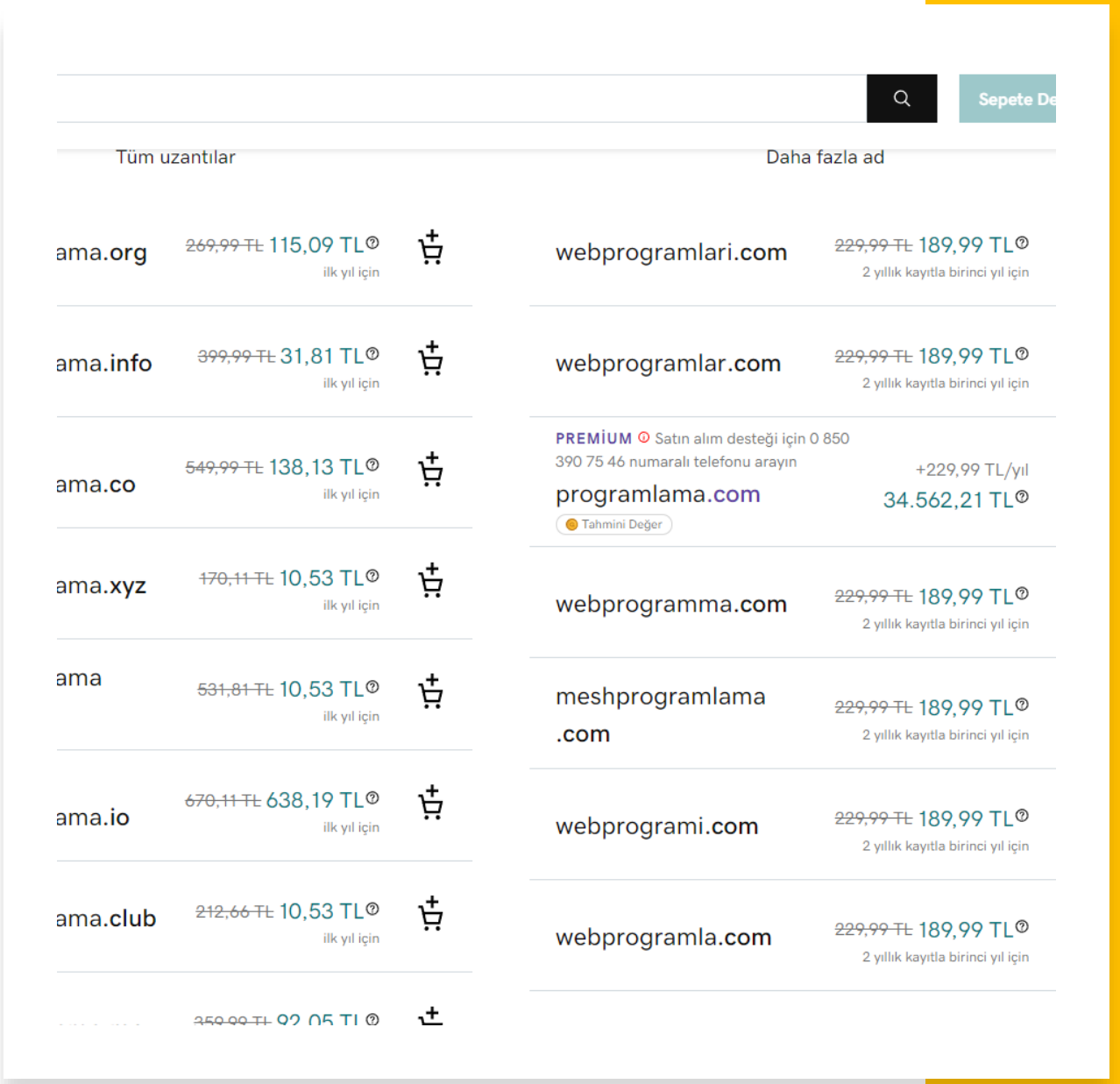
- Eğer IP adresini biliyorsanız doğrudan yazabilirsiniz
- Aynı isim farklı IP adreslerine işaret edebilir (ör: farklı sunucular)



OLDUKÇA ÖNEMLİ: Chrome -> Diğer Araçlar-> Geliştirici Araçları



- Seçtiğiniz bir IP adresi ile alan adı kaydetmek çok da pahalı değildir:
- Ör: *webprogramlama.org* 115 TL



Port’lar

- Bir IP adresi uzak TCP bağlantısı kurmak için yeterli değildir.
- Aynı zamanda 0-65535 (2^16 -1) aralığında “port” adı verilen yapılara da ihtiyaç bulunmaktadır.
- Uzak bir sunucuda bulunan uygulama aynı zamanda hangi portu dinleyeceği bilgisine de ihtiyaç duymaktadır
- Aynı makinada farklı portları dinleyen birden fazla farklı uygulama çalıştırabilirsiniz
- 0-1023 aralığı rezerve edilmiş portlardır ve bilinen uygulamalar kullanmaktadır

Bilinmesi Gereken Portlar

- 0: Dinamik olarak ayrılır (sonraki slayta bak)
- 22: SSH bağlantısı için
  - Uzak sunucuya terminal aracılığı ile bağlanmak istediğinizde oldukça sık kullanılır
- 80: HTTP bağlantısı için
  - Şifrelemesiz olarak bir sayfayı açmak isterseniz
- 443: HTTPS için, güvenli/şifreli HTTP
  - Bu günlerde çok daha yaygın kullanılmaktadır
- 8080: rezerve edilmemiş port
  - “Genellikle” yerel makinada HTTP sunucu kullanan araçlar tarafından kullanılır

Geçici/Dinamik portlar

- “Çoğu zaman” 49152–65535 aralığıdır
  - Ancak işletim sistemlerine göre çeşitlilik gösterebilir
- Kısa süreli iletişim için kullanılır
  - Uzak sunucu ile bağlantı kurulduğunda, port lokal olarak cevabı okumak için açılır
- Geliştirme sırasında port 0 kullanılırken,
  - Boşta, kullanılmayan bir dinamik port alın
  - Ör, debug/test için yerel bir sunucu başlatmak istiyorsanız, aynı portu kullanan diğer uygulamalarla çakışmalardan kaçınmalısınız.
  - Testleri paralel koştururken farklı sunucu örneklerine ihtiyacınız olacaktır ve burada oldukça önemlidir

Varsayılanlar

- Hiçbir şey belirtilmez ise varsayılan protokol ve portlar kullanılır
  - Varsayılan protokol HTTP'dir ve varsayılan kaynak da kök dizin «/»'dır
- *www.google.com* yazıldığında eşdeğer: ***http://****www.google.com****:80/***
- http**s**://www.google.com yazıldığında eşdeğer: http**s**://www.google.com**:443/**
- Not: 3xx HTTP yönlendirmesi gerçekleşebilir bu yüzden talep ettiğiniz sayfa ile size sunulan sayfa aynı olmayabilir

Yerel Ağlar

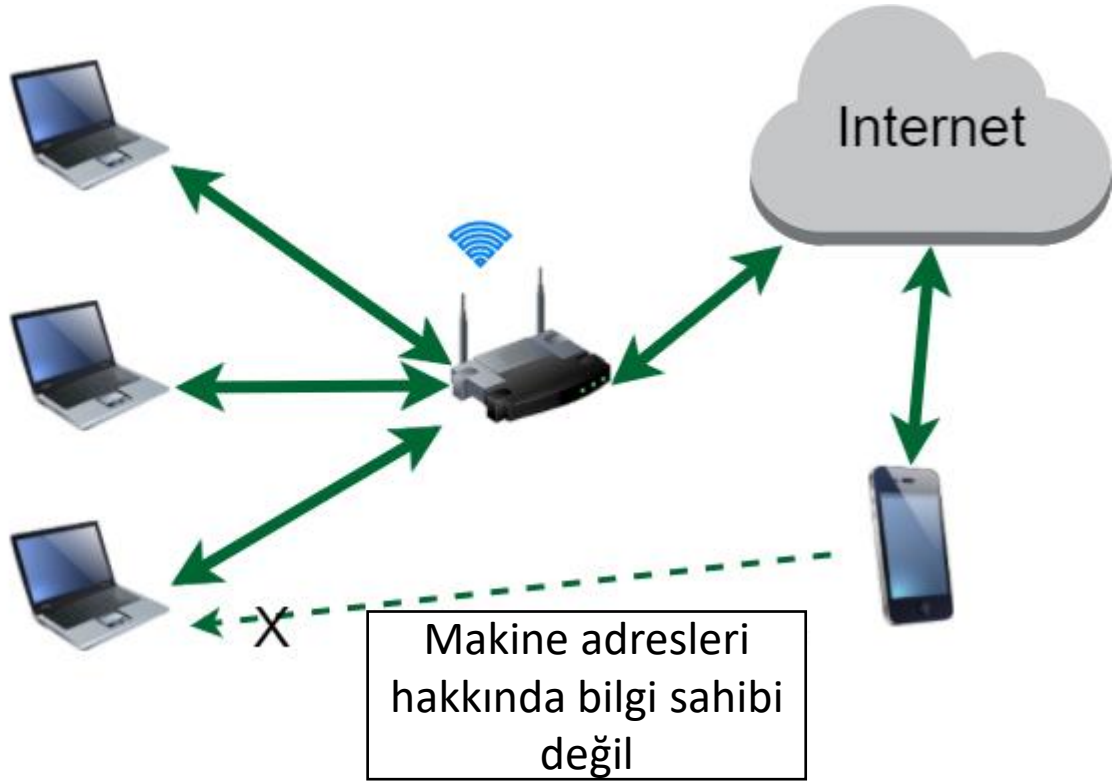
Router internetten erişilebilir IP adresine sahiptir

Router'a bağlı makinaların yerel IP adresleri bulunur ve dışarıdan erişilemez

10.x.x.x
172.[16-31].x.x
192.168.x.x

Router'da özel ayarlar veya doğrudan aynı router'a WiFi ile bağlı olmadığı sürece, bu tür makinelere bağlanmak için mobil cihaz kullanılamaz

Router güvenlik sebebiyle aynı ağ içerisinde makinadan-makinaya bağlantıya izin vermiyor olabilir



HTTP Mesajı

- Tarayıcıya göstermesi için “*http://www.google.com:80*” adresini yazdığınızda, tarayıcı uzak sunucu ile TCP bağlantısı kurar
- Tarayıcı bu TCP bağlantısı üzerinden byte’lardan oluşan bir stream yayını yapar
  - byte[] dizilerinden oluştuğu düşünülebilir
- HTTP (Hypertext Transfer Protocol) ise bu byte[] dizilerinin gönderilmesi/alınmasında kullanılan protokoldür

HTTP Protokolü (Özet)

Ayrıntılarına Web Programlama II dersinde gireceğiz

3 temel kısımdan oluşur

- Yapmak istediğiniz eylemi belirten ilk satır, Ör: bir kaynağı GET ile çekmek
- Ekstra bilgi sağlayan headers bilgisi
  - Ör cevap hangi türde olmalı: JSON? Plain Text? XML?
  - Hangi dilde olmalı? Türkçe? İngilizce?
- (Opsiyonel) Body: her şey olabilir.
  - Request: genellikle kullanıcı verisi sağlanır, ör: kullanıcı adı/şifre bilgileri
  - Response: Çekilen veri, ör HTML sayfası

En çok kullanılan versiyon 1.1'dir, bu versiyonda veri okunabilir string ifadelerle temsil edilir

- HTTP 2.0'da veri binary formatta gönderilir ancak aynı komutlar/header'lara sahiptir

HTTP Sunucu Uygulaması

Bir TCP port (ör, 80 veya 443) açıp bu portlardan gelecek istekleri dinleyen programdır

Her bir istekte bir HTTP cevabı döner

“Kaynak” diskte var olan bir dosya (ör., html sayfası, JPEG resim dosyası vs.), veya anında *oluşturulan* (ör., veritabanındaki içeriğe bağlı olarak) bir şey olabilir

“Eskiden”, bir “web uygulaması” yalnızca HTTP üzerinden erişilebilir statik dosyalardan oluşurdu

Eski Zamanlarda Sunucular

/where/installed/

/index.html
/figs/cat.jpeg
/figs/dog.jpeg

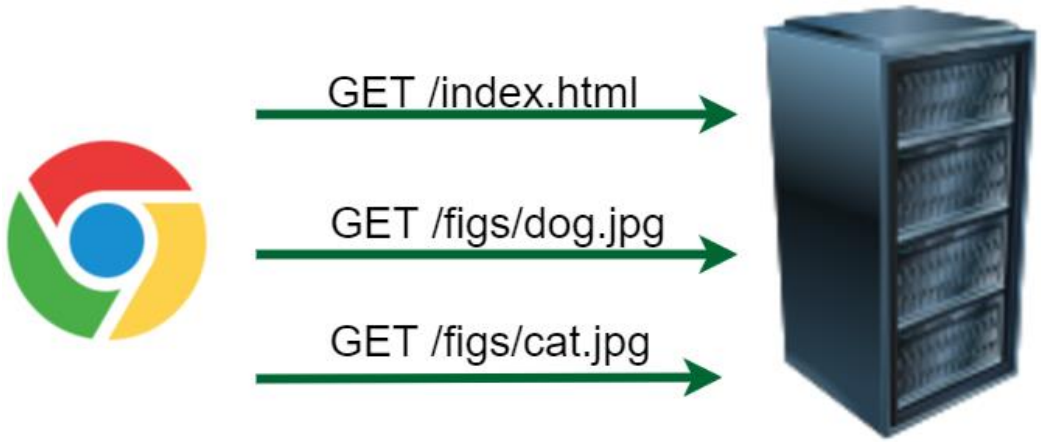
80 portunda çalışan uygulama, “*/where/installed*” klasöründeki dosyaları sunardı

Eğer tarayıcı “*www.foo.org/index.html*”, adresine istekte bulunursa sunucu “*/where/installed*” içerisinde “*index.html*” dosyasını HTTP response body’si içerisinde gönderirdi

```
<HTML>
<HEAD>
 <TITLE>Example</TITLE>
</HEAD>
 <BODY BGCOLOR="gray">
  <h2> A Cat and a Dog </h2>
  <DIV>
    <IMG SRC="figs/cat.jpg" style="width:256px">
    <IMG SRC="figs/dog.jpg" style="width:256px">
  </DIV>
 </BODY>
</HTML>
```

Diğer dosyalara bağlantı

- Bir önceki slayttaki sayfaı görüntülemek için tarayıcı 2 GET isteği daha yapacaktır
- Bu 2 bağlantı farklı TCP bağlantıları ile paralel olarak gerçekleştirilebilir

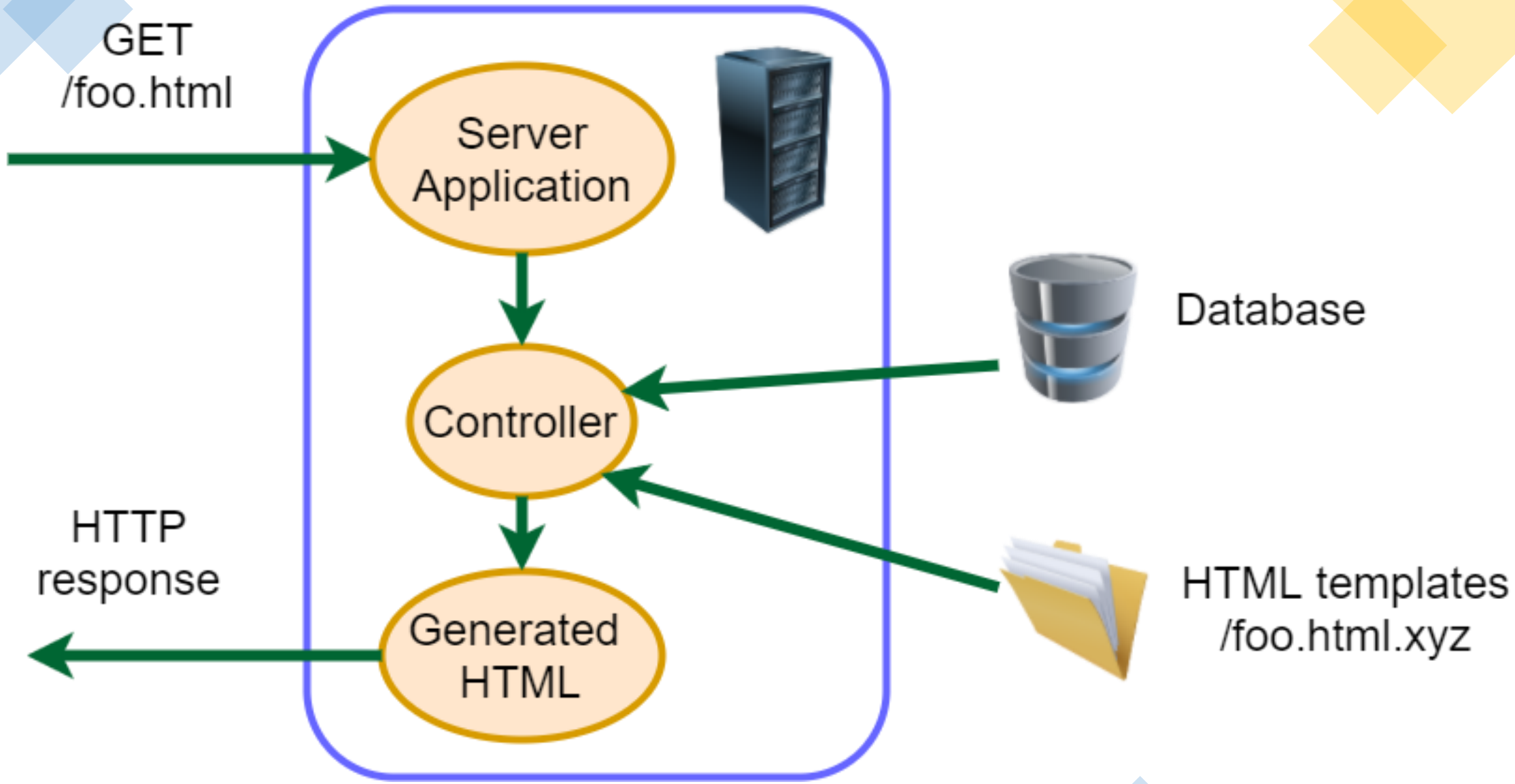


Devam...

- "*www.foo.org/index.html*" adresine istek atıldığında tarayıcı:
  - *www.foo.org* adresinin IP adresini çözümleyecektir
  - <IP adresi>:**80** adresi ile TCP bağlantısı kuracaktır
  - Şu komutla HTTP isteği atacaktır: "GET /index.html HTTP/1.1"
    - Burada "*index.html*" talep edilen kaynaktır
  - "*www.foo.org/figs/cat.jpeg*" adresine istek attığında ise komut "*GET /figs/cat.jpeg HTTP/1.1*" olacaktır
  - <IP address>:<port>'dan sonra gelen kısma "*path*" adı verilir ve sunucudaki kaynağı ifade eder
  - Not: İstemci tarayıcının dosyanın nerede saklandığına dair bilgisi bulunmaz, ör: "*/where/installed*" klasörü

Dinamik Sayfalar

- Statik, önceden tanımlı HTML sayfaları modern uygulamalar için yeterli değildir
- Veritabanındaki veriye veya dinamik bir veriye bağlı olarak HTML sayfası oluşturmak isteyebilirsiniz
  - Web forum
  - Alışveriş sepeti
  - Canlı chat
  - vs.
- HTML sayfalarının her bir istek sonucunda anlık oluşturulma ihtiyacı olabilir
- Tarayıcılar yalnızca bir HTML sayfası görürler, otomatik oluşturulsa bile bunu bilemez

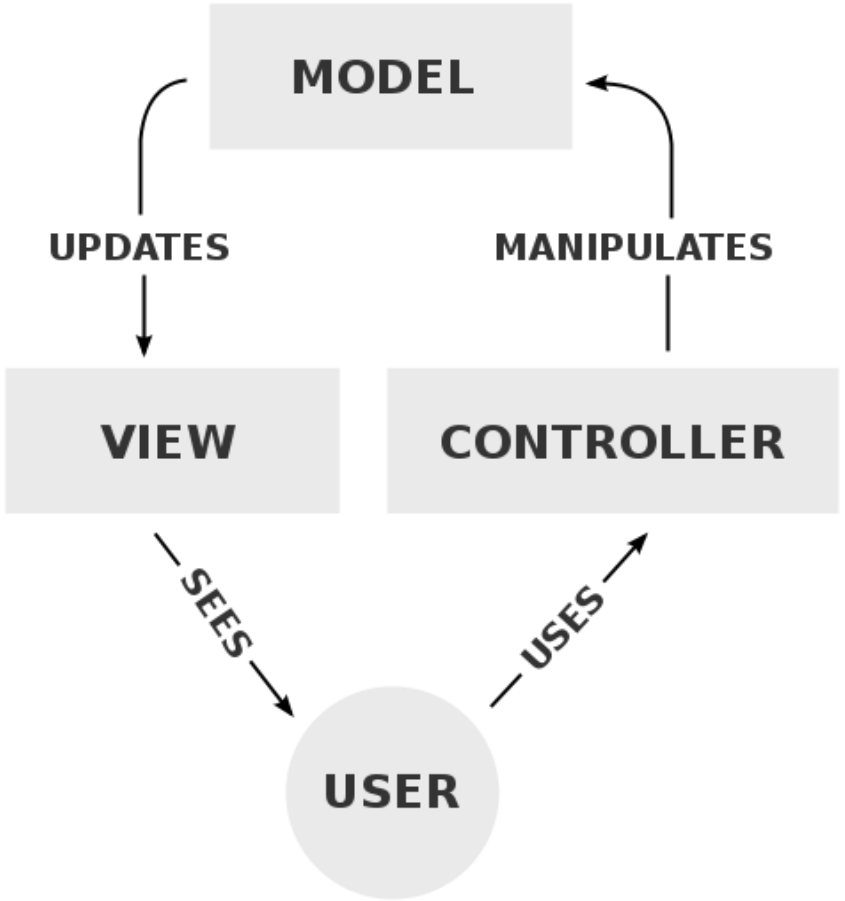

GET
/foo.html
Server
Application
Controller
Generated
HTML
HTTP
response
Database
HTML templates
/foo.html.xyz

Server-Side HTML Rendering

- Bütün HTML sayfası sunucu tarafında oluşturulur
- HTML taslakları
  - HTML ve verinin birleştirildiği ve dinamik kısmının belirlendiği dosyalardır
- Pek çok farklı teknoloji bulunmaktadır, hatta aynı dillerde bile birden fazla teknoloji bulunmaktadır
  - ör, Java (.xhtml) için JSF (JavaServer Faces)

MVC (Model–view–controller)

- GUI/Web uygulamaları için bir tasarım şablonudur
- Geliştirme ve bakımı kolaylaştırmak için açık bir ayrım sunmaktadır
- **Model**: uygulamanın iç durumu
- **Controller**: iş katmanında çalışan kod. Kullanıcı girdilerini alır ve durum güncellemesi yapar
- **View**: kullanıcının gördüğü yer, ör gösterilen HTML sayfaları



JavaScript ile Client-Side Rendering

- JavaScript'in (JS) Java ile hiçbir alakası yoktur
- Tarayıcıda çalışan *programlama dilidir*
- JS kodu web sayfası tarafından diğer kaynaklar gibi (resimler, css dosyaları vs.) referans gösterilebilir veya doğrudan HTML içerisine gömülebilir
- JS DOM (Document Object Model) manipülasyonu ile kullanıcı etkileşimlerine bağlı olarak (ör, Mouse ile tıklama) sayfada değişiklik gerçekleştirebilir)

JS: AJAX ve WebSocket

- DOM manipülasyonu için tarayıcıda JS çalıştırmak, AJAX ve WebSockets ile birçok fırsatın kapısını açar
- *AJAX* ile bütün HTML sayfası yerine ihtiyacınız olan veriyi (ör, JSON) çekebilirsiniz
- *WebSockets* ise sunucudan veri "*push*" etmeyi sağlar
  - Chat ve diğer gerçek zamanlı oyun gibi canlı uygulamalar için oldukça önemlidir

Frontend Development

- Front-end development daha karmaşık hale gelmeye başlamaktadır
  - Binlerce satır JS kodundan meydana gelebilir
- İyi bir GUI tasarımı özel bir yetenek gerektirir
- Büyük organizasyonlarda front-end ve back-end geliştirici takımlarının ayrılması oldukça olağandır
- Bu gibi durumlarda, «*taslak*» teknolojileri (JSF gibi) kullanımı en iyi seçenek olmayabilir
  - Front-end geliştiricileri back-end dilini bilmeyen HTML/CSS/JS uzmanları olabilir
  - Sayfalarınız backend çalıştırmaya bağlı olduğunda, GUI'yi prototiplemek daha zordur

Mevcut Çözümler

- Front-end ve back-end'i tamamen ayırma
- *Frontend*: yalnızca HTML/CSS/JavaScript gibi "*statik*" dosyalar
  - Taslak dili yok
- *Backend*: Yalnızca JSON formatında veri sağlar ve istemci tarafındaki JS HTML statik dosyalar üzerinde DOM manipülasyonu yaparak veriyi gösterir
  - Ör, client-side rendering
  - HTML sayfası yüklenirken veri de AJAX ile okunabilir
- JSON data nasıl sağlanır? **RESTful** Web Servisler

REST (Representational State Transfer)

- Protokol değil, mimaridir
- Web kaynaklarının nasıl yapılandırılıp erişileceğini tanımlamak için kullanılır
- HTTP protokolünü temel alır
- Sunucu yolu hiyerarşik olarak verileri temsil eden farklı URL isteklerine cevap verir, ör
  - /menus , bütün menüleri döner
  - /menus/today , günün menüsünü döner
  - /menus/today/drinks, günün menüsünün içeceklerini döner
- Veri farklı formatlarda dönebilir ancak çoğu zaman JSON (JavaScript Object Notation) formatındadır çünkü tarayıcı JS'i doğrudan JSON kullanabilir

JavaScript + REST

- Pek çok firma sunucu taraflı template kullanmak yerine bu ikiliyi kullanmaya başlamıştır
- Farklı RESTful servisler farklı dillerde yazılabilir ve tek bir sayfa olarak sunulabilir
- RESTful servisler ayrıca farklı GUI'ler (mobil uygulamalar gibi) ile de sunulabilir
- Frontend bir backend uygulaması olmadan test edilebilir/prototiplenebilir (yalnızca statik dosyalarla JSON yanıtları simüle edilebilir)

Bu Ders’te

- Bu derste, bu derste yalnızca JSF gibi template teknolojisi kullanarak server-side HTML rendering kısmını göreceğiz (Java Backend ile yakın ilişkili)
- Bu ders gelecek sene açılacak olan Web Programlama I dersi olacaktır.
- Gelecek sene açılacak olan Web Programlama II dersinde ise, REST ve AJAX ile WebSocket kullanarak client-side rendering göreceğiz. Burada React gibi bir kütüphane kullanacağız (backend ile hiçbir bağı bulunmayan) ve bu uygulamlara Single-Page-Applications adı verilmektedir
- Not 1: Mevcut piyasada, React gibi kütüphaneleri öğrenmek JSF öğrenmekten daha iyidir…
- Not 2: Oldukça basit sayfalarla JSF’i yüzeysel olarak göreceğiz.
- Not 3: Gelecek sene açılacak olan Web Uygulama Geliştirme dersi bu sene açılan Web Programalama I dersi ile aynı içeriğe sahip olacaktır.

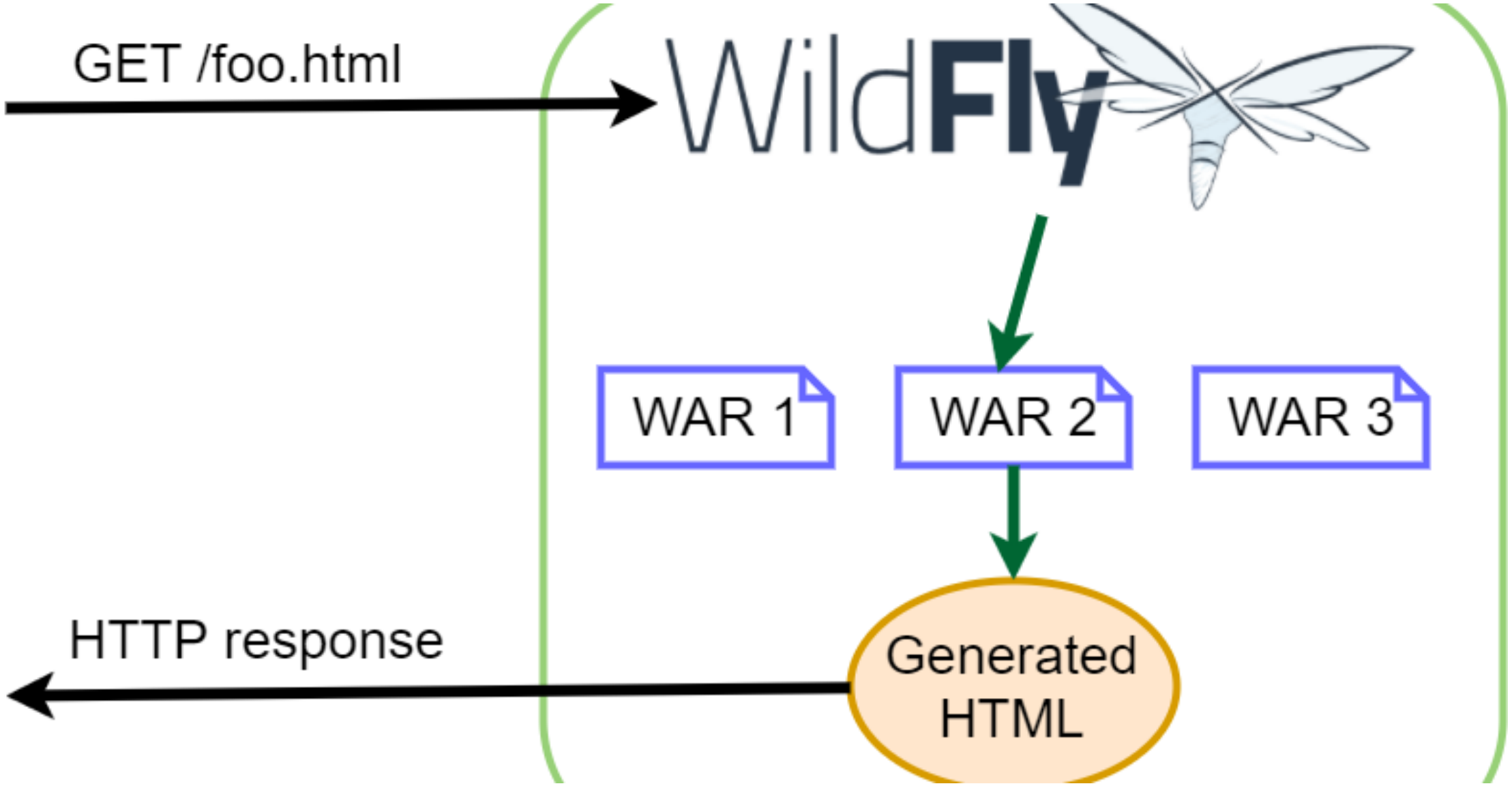
Peki neden JSF?

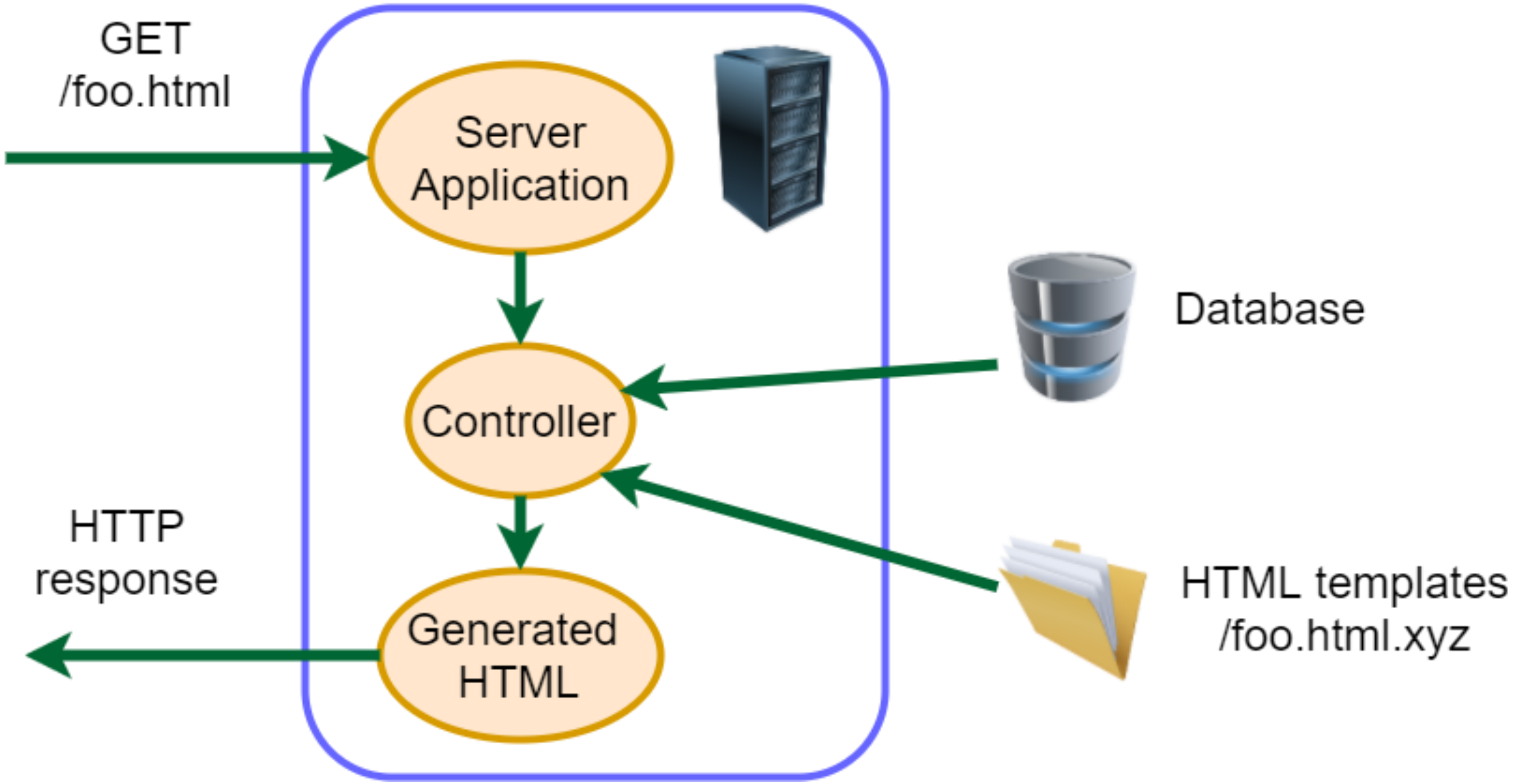
- Web Programlamanın temellerini öğrenmek önemlidir
  - JS olmadan tam web uygulamalarının nasıl geliştirilebileceğini öğrenmeniz gerekiyor
  - sadece en son trendleri incelemek size yarının yeni teknolojilerini anlamanız için derinlik ve fikir vermez.
- Server-side HTML rendering hakkında biraz tecrübe sahibi olmanız için
- Bir başka template teknolojisi de kullanılabilirdi ancak JSF seçim sebebi hem Spring hem de JEE'de kullanılabilir olması

JSF

JEE Container'ları

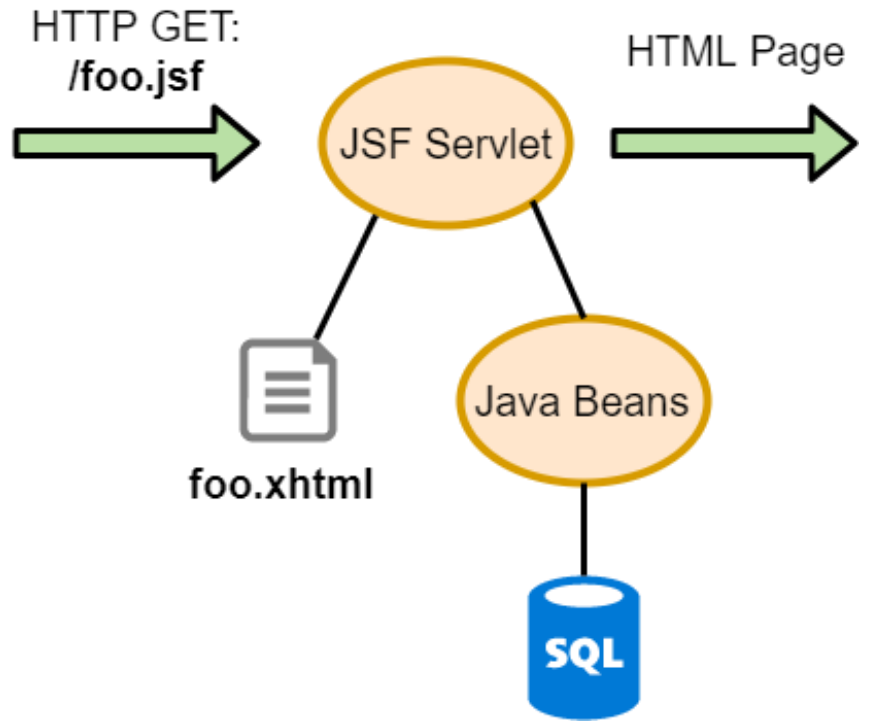
- Bir container JPA ve EJB gibi özellikleri yürütmenin yanında HTTP requestlerini de alabileceğiniz bir web server sağlar
- Bir *servlet* uygulaması, HTTP isteklerini işleyen ve şablonlara dayalı dinamik HTML sayfaları oluşturan kısımdır.
- Altında servlet kullanılan farklı template teknolojileri vardır
  - JSF, JEE'nin temellerinden biridir ve eski JSP'in yerine gelmiştir
- Container'a farklı WAR'lar kurulanabileceğinden ilk adım bir HTTP isteği geldiğinde hangi WAR'ın o isteği ele alacağını kontrol etmektir (URL path'ine bağlı olarak)


GET /foo.html
WildFly
WAR 1
WAR 2
WAR 3
Generated HTML
HTTP response


GET
/foo.html
Server
Application
Controller
Generated
HTML
HTTP
response
Database
HTML templates
/foo.html.xyz

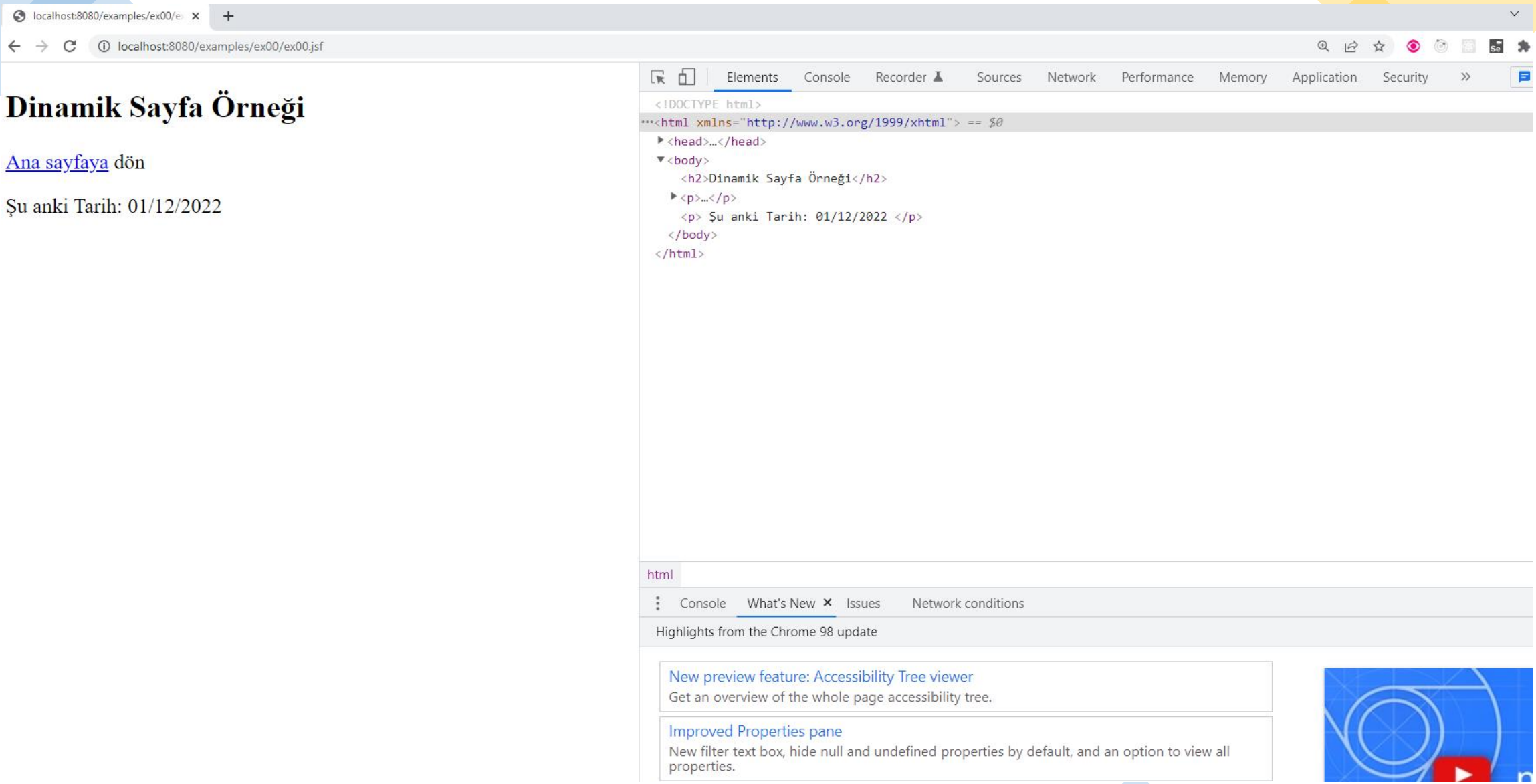
JSF: JavaServer Faces

- Template teknolojisi
- Template'ler *XHTML*: Extensible Hypertext Markup Language ile yazılır
- HTML'e benzeyen geçerli XML dosyalarıdır
- HTML etiketlerini (örneğin <p>, <a>) JSF Servlet tarafından işlenecek ve dönüştürülecek özel etiketlerle kullanım



JSF Template .XHTML Örneği

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html
        xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://xmlns.jcp.org/jsf/html"
        xmlns:f="http://xmlns.jcp.org/jsf/core"
>
<body>
<h2>Dinamik Sayfa Tarihi</h2>
<p>
<a href="../index.html">Ana sayfaya</a> dön
</p>
<p>
    Şu anki Tarih: <h:outputText value="#{session.lastAccessedTime}">
                        <f:convertDateTime pattern="MM/dd/yyyy" type="date"/>
                    </h:outputText>
</p>
</body>
</html>
```


localhost:8080/examples/ex00/e
localhost:8080/examples/ex00/ex00.jsf
Dinamik Sayfa Örneği
Ana sayfaya dön
Şu anki Tarih: 01/12/2022
Elements
Console
Recorder
Sources
Network
Performance
Memory
Application
Security
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"> == $0
<head>…</head>
<body>
<h2>Dinamik Sayfa Örneği</h2>
…
 Şu anki Tarih: 01/12/2022
</body>
</html>
html
Console
What's New
Issues
Network conditions
Highlights from the Chrome 98 update
New preview feature: Accessibility Tree viewer
Get an overview of the whole page accessibility tree.
Improved Properties pane
New filter text box, hide null and undefined properties by default, and an option to view all properties.

Tag’lar

- Tag’lara göre JSF *<tags>’ların* HTML’e nasıl dönüştürüleceğine karar verir
- JSF’in önceden tanımlı tag seti bulunmaktadır
- Bazı durumlarda doğrudan HTML’e dönüştürülür
- Diğer durumlarda HTML üretmesi için kod çalıştırılır
- **<h:outputText value="#{session.lastAccessedTime}">**
  - “#{}” içerisindeki kısımda Java kodu çalıştırılır
- **<f:convertDateTime pattern="MM/dd/yyyy" type="date"/>**
  - Java Date objesi nasıl yazdırılmalı ona karar verir

Namespace (ns)

```
<html   xmlns="http://www.w3.org/1999/xhtml"
        xmlns:h="http://xmlns.jcp.org/jsf/html"
        xmlns:f="http://xmlns.jcp.org/jsf/core">
```

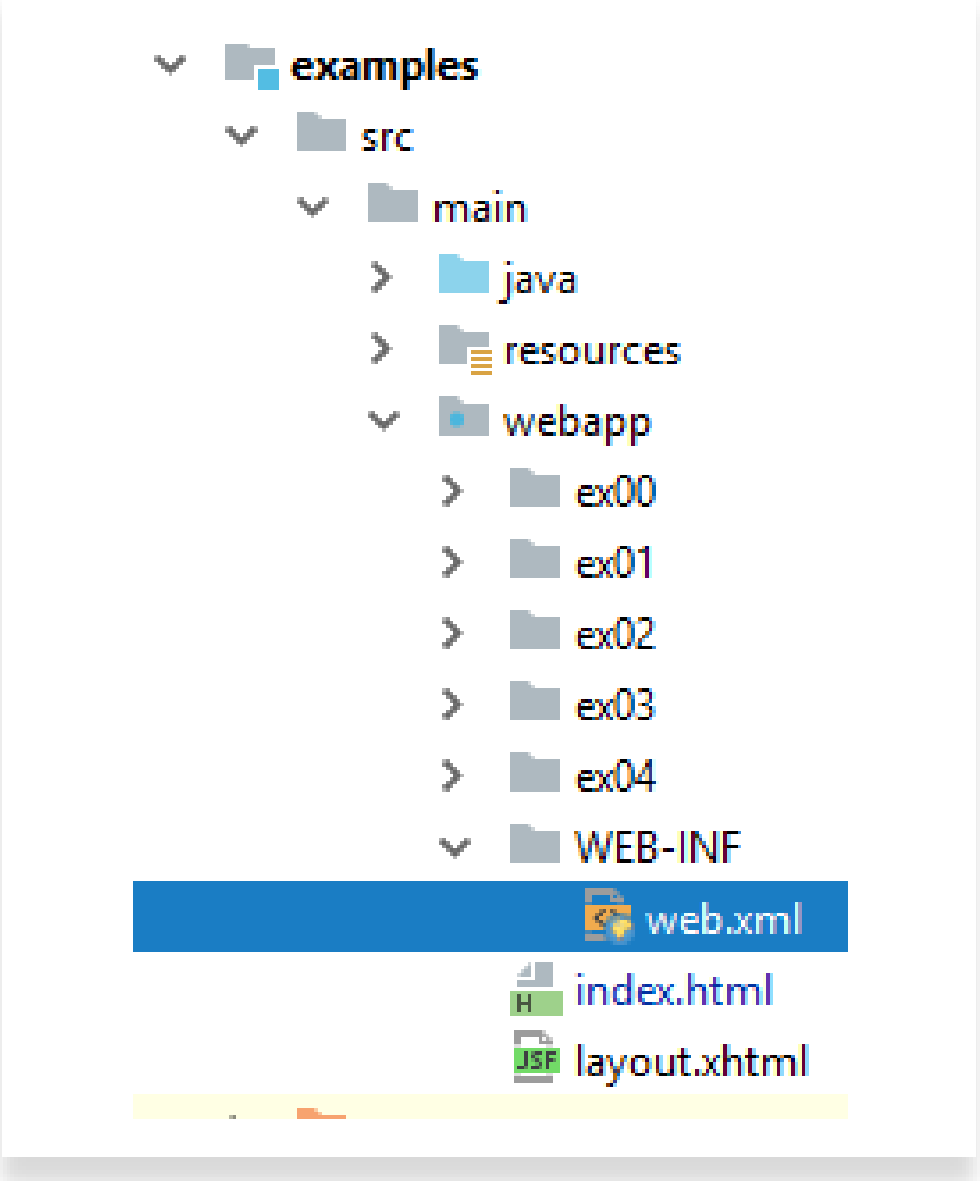
- Namespace'ler Java'daki "*import packages*" gibidir
- Kullanılan namespace'ler özel bir ad ile en başta belirtilir (genellikle çok fazla yazmaktan kaçınmak için h veya f gibi yalnızca bir harftir)
- *<a>* varsayılan namespace'e aittir
  - Bu örnekte "http://www.w3.org/1999/xhtml"
- *<h:outputText>* h olarak isimlendirilmiş namespace'e aittir
  - Bu örnekte "http://xmlns.jcp.org/jsf/html"

JSF Tag’ları

- JSF farklı namespace’lerle pek çok tag sağlar
- JSF ayrıca PrimeFaces ve OmniFaces gibi kütüphanelerle genişletilebilir taglara sahiptir
  - Ancak bunları bu ders kapsamında görmeyeceğiz…
- Mevcut tag/namespace’ler (Bütün listeye internetten erişebilirsiniz):
  - http://xmlns.jcp.org/jsf/html
  - http://xmlns.jcp.org/jsf/core
  - http://xmlns.jcp.org/jsf/facelets
  - http://xmlns.jcp.org/jsf/composite
  - http://xmlns.jcp.org/jsf/passthrough
  - http://xmlns.jcp.org/jsp/jstl/core
  - http://xmlns.jcp.org/jsp/jstl/functions

Maven Dosya Yapısı

- *.XHTML* template'leri ve diğer statik dosyalar (HTML/CSS/JS/images/vs.) derlenecek kaynak dosyalar değildir
- Bunlar "*src/main/webapp*" içerisine konur
- Bu klasör altındaki her şey HTTP ile erişilebilirdir ancak "*WEB-INF*",
- "*WEB-INF/web.xml*" ayar dosyasıdır



JSF Managed Beans

- XHTML içerisinde, **#{}** blokları içerisinde Java kodu çağırabiliriz
  - Sunucu tarafında HTML oluşturulduğunda "*değer*" okunabilir
  - HTTP POST isteği sunucuya geldiğinde bir "*eylem*" yürütülebilir
- JSF HTML GUI ve iş katmanı (ör, EJB) arasında köprü görevi gören özelleşmiş bean'ler kullanır
- Farklı türde JSF-Bean'ler bulunur ancak biz 2'sine ihtiyaç duyacağız:
  1. *@RequestScoped*, her bir HTTP isteğinde oluşturulur
  2. *@SessionScoped*, her bir kullanıcı oturumunda oluşturulur (cookie tabanlı)

Devam...

- XHTML'den referans alınabilmesi için bean'lerin @Named ile işaretlenmesi gerekir
  - daha sonra bean örneğini çağırmak için sınıfın adını ilk harfle küçük harfle kullanılabilir olacaktır, örneğin bir FooBar sınıfına fooBar adlı bir değişkenle çağırılabilir
- JSF-Bean'lerin injected EJB'leri olabilir
- JSF-Bean'lerin business logic (iş mantığı) bulunmamalıdır (iş mantığı yalnızca EJB'de olmalı)
- JSF-Bean'ler temel olarak form verileri ve sayfa yönlendirmelerini ele alır

Sayfa Yönlendirmeleri

- Basit sayfa yönlendirmeleri için *<a>* tagı kullanılabilir
- *<form>* tagı backend'de JSF-Bean'leri POST ile çalıştırdığında karmaşa ortaya çıkmaktadır
  - Bu "*action*" özelliği ile tanımlanır
  - ör, *<h:commandButton value="X" action="#{fooBar.someMethod}" />*
- Eğer JSF-Bean metodu void dönerse POST da mevcut HTML sayfasını döner
- Eğer String bir ifade dönerse, başka bir sayfa yönlendirme URL'i olabilir
  - ör, form'a bir veri submit edilirse yeni verileri içeren sayfa görüntülenmek istenebilir
  - ör, login sonrası (userId/password form ile gönderildiğinde), otomatik olarak anasayfaya yönlendirme

Forward ve Redirect

- JSF'te, sayfa yönlendirmeleri için 2 temel yöntem bulunmaktadır
- **Forward**: POST isteğinin body payload'ı olarak bağlantılı sayfanın render edilmiş HTML'ini döndür
  - Varsayılan davranıştır
  - Problem: Tarayıcıdaki URL değişmez
- **Redirect**: 302 yönlendirmesi döner location header'ı içerisinde path bulunur
  - Problem: Tarayıcı 2 istek atmak zorunda kalacaktır (ilki POST, sonra GET), ancak adres satırı doğru bir şekilde güncellenecektir
  - URL özelliği ile aktif edilmesi gerekir "**?faces-redirect=true**"
  - Çoğu zaman kullanacağınız budur

Docker

OS Image Deployment

- Bir uygulama geliştirirken yalnızca kodunuzu paketlemekle sınırlı kalmayın
  - Java, NodeJS, PHP, vs.
- Bütün ihtiyaç olan uygulamaları barındıran bir OS imaj dosyası oluşturun
  - ör JRE/.Net/Ruby/vs. gibi ihtiyaç duyduğunuz versiyonda uygulamalar
  - Ör WildFly gibi JEE container'ı
- Özellikle bir web uygulamasını sunucuya kuracak olduğunuzda oldukça kullanışlıdır
- Sunucuya bir OS imajı kurmayın, bunun yerine bir sanal makine oluşturun
- Ayrıca bir veritabanı kurmak yerine yalnızca veritabanı barındıran bir imaj dosyası çalıştırabilirsiniz
- Peki bütün bunlar nasıl otomatikleştirilir?

Docker

- Yazılım container'ları içerisinde deployment'ı otomatize eder
- Önceden tanımlanmış olanlara bağlı olarak bir OS imajı oluşturmanızı sağlar
- Ör, En son sürümleri barından frameworklere sahip bir Linux dağıtımı
    - NodeJS, PHP, JDK, vs.
- Docker Hub'ta var olan imajlara ait büyük *online* katalog bulunmaktadır
- Uygulaman ve ihtiyaç duyduğun 3. parti kütüphaneler OS imajının bir parçası olacaktır
- Docker kullanarak OS imajlarını deploy edebilir ve onları lokal/uzak sunucularda başlatabilirsiniz

Docker nasıl kullanılır?

- İlk olarak kurmanız gerekir
  - https://store.docker.com/
  - Not: Eğer Windows kullanıyorsanız 2020 yılı itibari ile Windows Home'a docker desteği gelmiştir. Daha öncesinde kullanılamıyordu
- Var olan imajları çalıştırmak için bir Shell terminal'e ihtiyacınız vardır (ör, GitBash)
- Kendi uygulamanı yazdığında ise bir ayar dosyası oluşturmalısın
  - *Dockerfile*: OS imajı nasıl oluşturulacak belirtmek için
  - *docker-compose.yml*: Çoklu imajları yönetmek için
- Daha sonra *docker* ve *docker-compose* komutlarını komut satırında kullanabilirsiniz

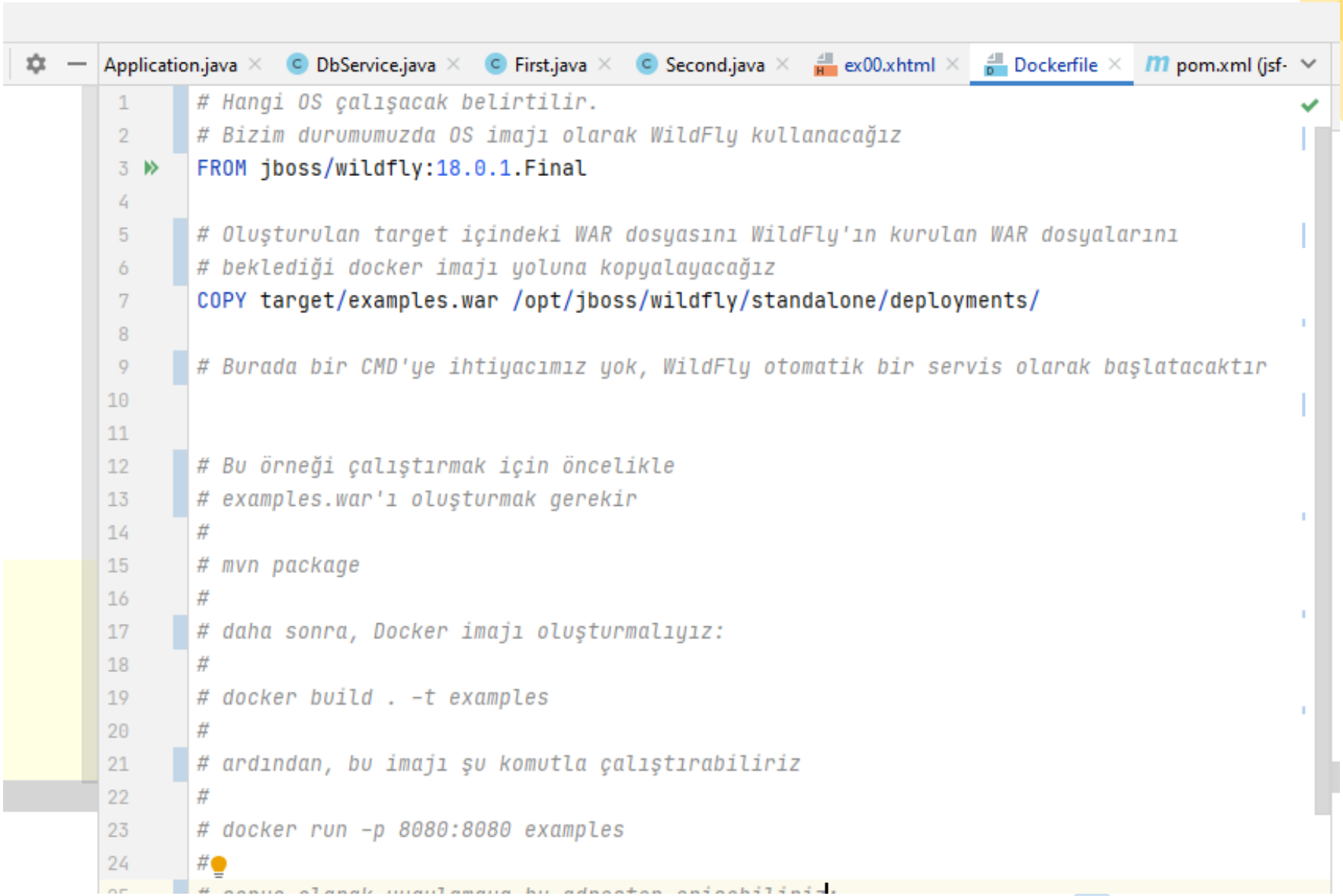
Docker Örnekleri

- https://docs.docker.com/get-started/
- https://hub.docker.com/r/docker/whalesay/
- ***docker run docker/whalesay cowsay boo***
  - Bu "*docker/whalesay*" imajını kuracaktır ve daha sonra girdi olarak «cowsay boo» ile çalıştıracaktır
  - İlk çalıştırmada, "*docker/whalesay*" imajı indirilecektir

```
Microsoft Windows [Version 10.0.19042.1415]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\Vita>docker run docker/whalesay cowsay boo
Unable to find image 'docker/whalesay:latest' locally
latest: Pulling from docker/whalesay
Image docker.io/docker/whalesay:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image for better future compatibility. More information
at https://docs.docker.com/registry/spec/deprecated-schema-v1/
e190868d63f8: Pull complete
909cd34c6fd7: Pull complete
0b9bfabab7c1: Pull complete
a3ed95caeb02: Pull complete
00bf65475aba: Pull complete
c57b6bcc83e3: Pull complete
8978f6879e2f: Pull complete
8eed3712d2cf: Pull complete
Digest: sha256:178598e51a26abbc958b8a2e48825c90bc22e641de3d31e18aaf55f3258ba93b
Status: Downloaded newer image for docker/whalesay:latest
 _____
< boo >
 -----
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \     \        __/
          \____\______/

C:\Users\Vita>
C:\Users\Vita>docker run docker/whalesay cowsay boo

 _____
```

Özelleştirilmiş İmajlar

- Geliştirdiğiniz uygulamaları çalıştırmak için mevcut imajları genişletin
  - *Yalnızca "Dockerfile"* oluşturmanız gerekir
- **FROM:** temel OS imajını belirtir
- **RUN**: programları yüklemek veya dosya/dizin oluşturmak gibi sanal işletim sisteminde komutlar yürütür
- **CMD**: uygulamanız için komut yazmaya yarar
- **ENV**: environment değişkeni tanımlamak için kullanılır
- **COPY**: hard-diskinizdeki X dosyasını alıp Docker image'indeki Y path'ine kopyalar
  - Docker imajı çalıştığında, görüntüyü uzak bir sunucuya dağıtsanız bile X'e Y yolundan erişilebilir
  - Not, ayrıca **ADD** seçeneği de vardır ancak kullanmamalısınız (ör, önerilmemektedir çünkü dosyaları eklerken çeşitli side-effect ile karşılaşabilirsiniz)
- **WORKDIR**: çalışılan klasörü belirtmek için kullanılır
  - Bu dosyaya «cd» yapıyor gibi düşünebilirsiniz, böylelikle bütün komutlar o klasör içerisinde çalıştırılır ve full path vermenize gerek kalmaz
- **#** yorum için kullanılır


Application.java
DbService.java
First.java
Second.java
ex00.xhtml
Dockerfile
pom.xml (jsf-
Hangi OS çalışacak belirtilir.
Bizim durumumuzda OS imajı olarak WildFly kullanacağız
FROM jboss/wildfly:18.0.1.Final
Oluşturulan target içindeki WAR dosyasını WildFly'ın kurulan WAR dosyalarını
beklediği docker imajı yoluna kopyalayacağız
COPY target/examples.war /opt/jboss/wildfly/standalone/deployments/
Burada bir CMD'ye ihtiyacımız yok, WildFly otomatik bir servis olarak başlatacaktır
Bu örneği çalıştırmak için öncelikle
examples.war'ı oluşturmak gerekir
#
mvn package
#
daha sonra, Docker imajı oluşturmalıyız:
#
docker build . -t examples
#
ardından, bu imajı şu komutla çalıştırabiliriz
#
docker run -p 8080:8080 examples
#

Docker Komutları

- ***docker build -t <name> .***
  - Mevcut . Klasöründeki Dockerfile'a göre <*name*> isimli bir imaj oluştur
- ***docker run <name>***
  - İsmi verilen imajı çalıştır
- **docker ps**
  - Çalışan imajları listele
- **docker stop <id>**
  - Verilen imajı durdur. Not: bir imaj birden fazla örnekte farklı id'lerle çalışıyor olabilir
- IntelliJ'de ayrıca "*Docker integration*" plugin'ini kurabilirsiniz

Network

Local host’ta bir uygulama çalıştırdığınızda genellikle 80 veya 8080 TCP portu açılır

Docker içerisindeki sunucu da benzer tipte bir port açar ancak host işletim sistemi tarafından bu erişilemez

Bu yüzden host’a guest port olarak belirtmemiz gerekir

Ör.: **docker run –p 80:8080 foo**

- 80 portundaki localhost’a istek attığımızda docker’da 8080 portuna yönlendirilir

CTRL-C

- Docker'ı terminalde çalıştırdığınızda (ör "*docker run*") CTRL-C ile durdurabilirsiniz
- Ancak Windows/GitBash'te arka planda çalışmaya devam edebilir
- Bu durumda "*docker ps*" ile kontrol edebilirsiniz
- "*docker stop <id>*" kullanarak imajı durdurabilirsiniz
- Eğer arkaplanda çalışan Docker imajınız varsa ve TCP portlarını kullanıyorsa bu problem yaratabilir

Bu Derste Docker

- Java oldukça taşınabilir durumdadır ve monolitik Java sunucuları için Docker çok da kritik öneme sahip değildir
- Ancak JEE container'larını başlatmayı oldukça kolaylaştırmaktadır
- Ayrıca veritabanı için de kullanacağız ve tarayıcı (ör: chrome) testleri (ör: selenium) için kullanacağız
- Mikro servisleri ele alacağımız bir sonraki derste Docker oldukça önemlidir
  - Burada pek çok uygulamayı yöneteceğiz

Git Repository Modülü

- *NOT: açıklamaların büyük bir çoğunluğu kod içerisinde yorum satırı olarak bulunmaktadır, burada slaytlarda bulunmamaktadır*
- **intro/jee/jsf/base**
- **intro/jee/jsf/examples**
- Ders 07 alıştırması (dokümantasyona bakınız)