|  |
| --- |
|  |
| **İSTANBUL ÜNİVERSİTESİ - CERRAHPAŞA** |
| **MÜHENDİSLİK FAKÜLTESİ** |
| muh_logo_300dpi |

İstanbul Üniversitesi - Cerrahpaşa

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

STAJ DEFTERİ

Fotoğraf

İSTANBUL ÜNİVERSİTESİ - CERRAHPAŞA

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

**Staj Yapan Öğrencinin**

|  |  |  |  |
| --- | --- | --- | --- |
| Adı | Anıl | Soyadı | Ertürk |
| Öğrenci No. | 1358130030 | Bölümü | Bilgisayar Mühendisliği |
| Eğitim-Öğretim yılı | 2017-2018 | Telefon No. | 5073184550 |
| e-posta | anilerturk1@gmail.com |
| İkametgâh adresi | Küçükyalı Mh., Eski Bağdat Cd., Ferah Sk., Önder sitesi B-blok Daire 11, Maltepe-İstanbul | | |

**Staj Yapılan İşyeri Yeri ve Staj Tarihi**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Adı | ERTÜRKLER DAY. TÜK. MAL. TEL. GIDA SAN. ve TİC. LTD. ŞTİ. | | | | |
| Adresi | Cevizli Mah. Hızır Reis Sk. Dragos Park Plaza No:10 K:12 D:26 Maltepe/İstanbul | | | | |
| Üretim/Hizmet Alanı | Telekominikasyon / Bilişim | | | | |
| İmza/ Mühür/Kaşe |  | | | | |
| Telefon No. | 216 366 15 54 | | Faks No. | 216 366 15 56 | |
| e-posta | [info@erturkler.org](mailto:info@erturkler.org) | | Web adresi | [www.erturktelekom.com](http://www.erturktelekom.com) | |
| Staja başlama tarihi | 13/08/2018 | Bitiş tarihi | 17/09/2018 | Süresi (gün) | 20 |
| Staj Türü | Staj - 2 | | | | |





|  |
| --- |
| **KURUM/BİRİM TANITIMI**  (Bu sayfada staj yapılan yer hakkında genel bilgiler verilecektir.)  Ankesörlü telefon kartı üretimi ve dağılımı ile işe başladı şirket. Ankesörlü telefon kullanımı azalıp, mobil telefon kullanımı arttıkça telefonlardan da kullanılabilen iletişim kartlarının üretimi ve dağılımına geçiş yapıldı. Sivil büfeler, askeri birlikler, cezaevleri gibi marketlere satış yapıldı.  Ankesörlü telefondan mobil telefona ve iletişim kartından iletişim hattına geçiş yapılırken askeri birliklerdede akıllı olmayan telefonlara izin verilmeye başlanıldığında şirket kendi telefonunu dizayn etti. Bilgisayar mühendisi C dili ile telefonu askeriyenin istediği şekilde programladı. Bunun yanında iletişim kartından iletişim hatlarına geçiş yapıldı. Türkcell ile birlikte çalışılıp askercell hatlarının şirketin ürettiği telefonla birlikte askeri birliklerde ve diğer marketlerde satışına başlandı. Şirket hatları hem sattı hemde türkcell ile anlaşıp hatların kayıp ve aktivasyonunuda kendisi yapmaya başladı.  BTK dan gerekli certifikalar alındı ve şirket iletişim trafiğini kendi serverlarından geçirmeye başladı.  Bunlar dışında PHP/Java dilinde uzman programcılarda şirkette çalışıyor.  Bilişim dışında inşaat ve askeri malzeme satışı gibi alanlardada faaliyet gösteriyor şirket. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **GİRİŞ/ÖZET**  (Bu sayfada staj dönemi boyunca yapılan işlerin ve gözlemlerin genel bir özeti bulunacaktır.)  Benim stajım Java server side programlama üzerineydi. Java platformunun PHP karşılığı olan Servlet teknolojisi ile programlar yazıldı.  Servlet teknolojisi web aplikasyonları yazmamıza izin veriyor statik web sayfaları yerine.  Servlet teknolojisi HTTP gibi internet protokolleri ile gelen isteklere (request) cevap (respond) verebilme özelliğine sahip. GET requesti veriyi url içinde (http header) gönderiyor, POST requesti veriyi http body içinde gönderiyor.  Servlet API’ının sunduğu javax.servlet ve javax.servlet.http paketlerindeki sınıflar kullanıldı.  GenericServlet ve HttpServlet sınıflarının bize sağladığı iskelet kullanılarak web aplikasyonları yapıldı. | | | | | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | | | | | Tarih: 13/08/2018 | | Sayfa No: 1 | | | |
| Yapılan iş: Oryantasyon | | | | | | | | | | | | | | | | | | | | |
| Core Java, Web programlama, Web hosting, PHP-Servlet gibi server side programlama teknolojileri, soket programlama, web servisleri konularında bilgi verildi.  Staj Servlet API’ı üzerine olduğu için özellikle bu konu üzerinde detaylı bilgi verildi. Staj boyunca kullanılacak teknolojilerin, bilgisayar teknolojilerinin nasıl çalıştığı hakkında konuşuldu. Core java aplikasyonu nedir, statik web aplikasyonu nedir, dinamik web aplikasyonu nedir anlatıldı. Servlet’ten önce kullanılan CGI (Common Gateway Interface) ile farkı, avantaj-dezavantajları anlatıldı. Servlet teknolojisi ile dinamik web sayfası yapabilmek için server’a konulması gereken application serverlar ve marketteki çeşitli örneklerinden bahsedildi.  İnternet sayfası nedir, statik internet sayfası nedir, dinamik internet sayfası nedir konuları hakkında konuşuldu. HTTP protokolü hakkında konuşuldu. HTTP request çeşitleri hakkında konuşuldu. Özellikle GET ve POST requestlerinin farkından bahsedildi. HTTP mesajının anatomısi (header ve body) gösterildi. HTTP protokolü ile farklı tip veriler gönderilebileceğinden bahsedildi MIME (Multipurpose internet Mail Extension)  Web server ve application server hakkında konuşuldu. Web server’ın statik veri istendiğinde gerekli cevabı verdiği, dinamik veri istendiğinde application server’a istek gönderdiğinden bahsedildi.  Servlet API’ının bize sundugu paketler ve bu paketlerdeki sınıflardan bahsedildi. Bu sınıflardaki metodlar tanıtıldı. Servletler sınıfı nasıl yüklenir, bu sınıftan objeler nasıl olusturulur, oluşturulan objeler nasıl başlatılır, kullanılır ve kullanımdan kaldırılır anlatıldı.  Servlet teknolojisinin özelliklerinden bahsedildi. welcome-file-list, load on startup, servlet request interface özelliklerinden bahsedildi. Request dispatcher ve send redirect özelliklerinden bahsedildi. servlet config, servlet context ve servlet attribute özelliklerinden bahsedildi. Cookie, hidden form, url rewriting ve httpsession ile session tracking özelliklerinden bahsedildi. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Bir iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | | Sayfa No: 2 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servlet’in Servlet arayüzündeki metodları tanımlanması gerekiyor. Bu metodlar init, service, destroy, getServletConfig ve getServletInfo. init metodu servlet objesini initialize etmek için gerekiyor. Bu metod bir servlet objesi için sadece bir kere çalıştırılır. service metodu, servlet objesine gelen her bir request için cevap oluşturmak için çalışır. Her bir request için çalışır yani birden çok kere çalıştırılır web container tarafından. destroy metoduda init metodu gibi sadece bir kere çalıştırılabilir bir servlet objesi için. sevlet objesini yok etmek için kullanılır.  Bir işi yapması için bir servlet sınıfı dizayn ederken servlet arayüzünü kendimiz implement edebiliriz. Eğer bir arayüz implement ediyorsak ve abstract biz sınıf değilsek, implement ettiğimiz arayüzün tüm metodlarını implement etmek zorundayız. 5 temel servlet metodunun ne yapacağını değiştirmek istiyorsak bu şekilde yapabiliriz.  **public** **class** ClassName **implements** Servlet  {  …  }  Ekran çıktılarında görebileceğimiz üzere io paketindeki PrintWrite sınıfını web sayfasına veri yazmak için ve IOExceptıon için kullanıyoruz. servlet paketini ise servlet arayüzünü implement etmek, ServletConfig sınıfı, ServletRequest ve ServletResponse sınıfları ve ServletException için kullanıyoruz.  Servlet çağırıldığında service metodundaki html kodu kullanıcıya gönderiliyor ve dolayısı ile ekrana “hello simple servlet”. Yazdırılıyor. Bu şekilde servlet kullanılabilir ama biz GenericServlet, HttpServlet gibi sınıflar kullanacağız ve bu sınıflar bizim için Servlet, ServletConfig ve Serializable arayüzlerinin implementation’unu yapacak. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Dört iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | | Sayfa No: 3 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servletler ile ilgili biraz daha bilgi vermemiz lazım.  Servlet kullandığımızda web.xml dosyasında servlet’leri belirtmemiz gerekiyor(deployment descriptor). Web container bu dosyayı kullanarak hangi servleti çağıracağına karar verir. Servlet tag’ı bir java sınıfını servlet olarak atar ve servlet e isim verir.  <servlet>  <servlet-name>ServletName</servlet-name>  <servlet-class>ServletClass</servlet-class>  </servlet>  Servlet mapping tag’ı bir servletin ismini ve o isme sahip servletin sahip olacağı url adresini belirtir. Bu adresi kullanarak servlete ulaşabiliriz.  http://hostname:portno/contextroot/urlpatternofservlet  <servlet-mapping>  <servlet-name>ServletName</servlet-name>  <url-pattern>/ServletURL</url-pattern>  </servlet-mapping>  Alttaki tanımlama ile foo adresi altında herhangi bir adrese ulaşmaya çalışan kullanıcıya aynı servletin servis vereceği belirtiliyor.  <servlet-mapping>  <servlet-name>ServletName</servlet-name>  <url-pattern>/foo/\*</url-pattern>  </servlet-mapping>  xml dosyasında farklı yaptığımız bir şey olmadığı sürece ekran çıktılarında göstermeyeceğim. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Dört iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | Sayfa No: 4 | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | | |
| Servlet arayüzünü kendimiz implement etmek yerine GenericServlet sınıfını extend edeceğiz. Bu sınıf Servlet, ServletConfig ve Serializable arayüzlerinin implementasyonunu sağlıyor.  Protokole bağımlı değildir. Her protokole cevap verebilir.  Daha önce bahsettiğimiz MIME’ı burada görüyoruz. Response’un veri tipi burada text/html olarak belirleniyor.  res.setContentType("text/html");  System.out console a veri gönderir.  PrintWriter out = res.getWriter();  daki out ise ise response’a veri ekler.  Response’a eklediğimiz veri ise aşağıda  out.print("<html><body>");  out.print("<b>hello generic servlet</b>");  out.print("</body></html>"); | | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Dört iş günü | | |  | | | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | Sayfa No: 5 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | | | |
| HttpServlet sınıfı GenericServlet sınıfını extend eder ve Serializabla arayüzünü implement eder. HTTP’ye özel metodlar sağlar. Bunlara örnek doGet, doPost, doHead, doTrace verilebilir.  javax.servlet.http paketi kullanılır.  public class HttpServlet2 extends HttpServlet  {  …  }  doGet metodu get requestlerine cevap verir.  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException  {  response.setContentType("text/html");//setting the content type  PrintWriter pw = response.getWriter();//get the stream to write the data  //writing html in the stream  pw.println("<html><body>");  pw.println("Welcome to http servlet 2");  pw.println("</body></html>");  pw.close();//closing the stream  }  Class loader servlet’i web server’a servlet için ilk request geldiğinde yükler. Web container servlet sınıfını yükledikten sonra objesini yaratır, init metodunu çağırır ve servlet’i initialize eder. Servlet objesine her request geldiğinde web container servlet’in service metodunu çağırır.  Destroy metodu manuel olarak çağırılabilir ama undeploy yapılırken tekrar çağırılacak dolayısı ile exception’a yol açacak. Eğer manuel olarak destroy metodu çağırılırsa bu exception’un halledilmesi gerekir. | | | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | | Stajyerin imzası | | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | |
| Dört iş günü | | | | |  | | | | | | |  | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | Tarih: 14-17/08/2018 | | | | | | | | | | | | | Sayfa No: 6 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | | | |
| Servlet arayüzü implementasyonunu kendimiz yapıyoruz yine ama bu sefer url olarak web sitesindeki servlet’in adresini girmiyoruz. Direk web sitesinin adresini giriyoruz. Bu bizi index.html’e götürüyor. Dosya belitrmediğimizde sitenin açma önceliği aşşağıdaki gibidir.   1. welcome-file-list in web.xml 2. index.html 3. index.htm 4. index.jsp   index.html dosyamızda bir link var. Linkimizin hypertext referansı başka bir siteye ait değil. Aynı sitedeki başka bir dosyayı gösteriyor. Ama alışkın olduğumuz gibi bir html dosyasını değil, bir servlet i gösteriyor. Servleti gösterirken web.xml dosyasında belirttiğimiz servlet url-pattern kullanılır. Buradada gözüktüğü gibi “hello” url pattern’ını kulandık.  <a href="hello">Invoke Simple Servlet 2</a>  Program akışı servlet’e geçtiğinde service metodu ile devam ediyor ve response’a eklenecek html kodları ekleniyor.  res.setContentType("text/html");    out.print("<html><body>");  out.print("<b>hello simple servlet 2</b>");  out.print("</body></html>"); | | | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | Stajyerin imzası | | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | |
| Dört iş günü | |  | | | | | | |  | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | | Sayfa No: 7 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| GenericServlet’i extend ediyoruz ama bu sefer url olarak web sitesindeki servlet’in adresini girmiyoruz. Direk web sitesinin adresini giriyoruz. Bu bizi index.html’e götürüyor.  Bir linkimiz var ve bu link generic servletimizi gösteriyor.  <a href="hello">Invoke Generic Servlet 2</a>  Ve servletimizdeki servis metodumuzda response’a eklediğimiz html kodu şu.  res.setContentType("text/html");    out.print("<html><body>");  out.print("<b>hello generic servlet 2</b>");  out.print("</body></html>");    Servlet’e url pattern ile erişebiliyoruz çünkü aplikasyon deploy edildiğinde (proje klasöründeki build dosyası) tüm web sayfaları ve servletler Web Pages klasörü altındadır. Bu build versiyonu server’a deploy edilen asıl aplikasyondur.  Hatta linkimize veri bile gizleyebiliriz.  <a href="ServletURL?operation=view">View My Guestbook</a> | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Dört iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 14-17/08/2018 | | | | | | Sayfa No: 8 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| HttpServlet i extend ederek kendi servletimizi oluşturuyoruz. Kendi servletimizde html dosyamızdaki formdan gelen veriyi alıp işliyoruz ve kullanıcıya dinamik bir cevap veriyoruz.  Html dosyamızdaki form için action attribute’unu kullanıyoruz. Bu attribute, butona basıldığında verinin nereye gönderileceğini gösterir.  <form action=”ServletURL”>  Formumuzda bir adet text field,bu text field’ı açıklayan bir text birde buton var.  Enter your name: <input type="text" name="name"> <br>  <input type="submit" value="login">  Bir servlet, kullanıcının http protokolü ile gönderdiği bir veriyi alırken getParameter metodunu kullanır ve istediği key-value ikilisinin key kısmını sağlar, value kısmını alır. Ve bizim servletimiz dinamik cevabı kullanıcıya iletir.  String name = request.getParameter("name");  pw.println("Welcome " + name);  Get request’i default request’tir.  Web projeleri transfer süresini azaltmak için WAR denilen dosyalar halinde kopyalanır server’a. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Dört iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 27-29/08/2018 | | | | | | Sayfa No: 9 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Bir kullanıcı dosya adı belirtmeden direk internet sitesinin adresine gitmek için server’a request yoladığında, server default olarak aşağıdaki sırayı takip eder ve bulduğu ilk dosyayı response olarak kullanıcıya gönderir.   1. welcome-file-list in web.xml 2. index.html 3. index.htm 4. index.jsp   welcome-file-list web.xml dosyamızda yaptığımız bir tanımdır. Dosya belirtmeden siteye ulaşmaya çalışan kullanıcılara gönderilecek dosya.adres belirtilir. Aşağıdaki örnekte index.html i hoşgeldin sayfası yaptık.  <welcome-file-list>  <welcome-file>index.html</welcome-file>  </welcome-file-list>  Aşağıdaki örnekte “ServletURL” url pattern’ına sahip servlet’i hoşgeldin sayfası yaptık.  <welcome-file-list>  <welcome-file>ServletURL</welcome-file>  </welcome-file-list>  Sıradaki örnekte bunun kullanımını göreceğiz. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Üç iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 27-29/08/2018 | | | | | Sayfa No: 10 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Load-on-startup tag’ı ise servletlerin deploy olacakları sırayı gösterir. Bazı servletlerin, server çalışmaya başladığında mümkün olduğunca hızlı deploy edilmesini isteyebiliriz. Bu özellik bu işimizi görmeye yarar. Ayrıca bazı servletlerin diğer servletlerden önce çalışmaya başlamasını isteyebiliriz. Deploy edilme sırasını web.xml dosyasından değiştirebiliriz. İlk sıra 1 değildir 0 dır. Eğer biz bu şekilde manuel olarak deploy etmez isek, ilk request geldiğinde otomatik olarak deploy edilir. Ama ilk istek yavaş çalışmış olur.  <servlet>  <servlet-name>NewServlet</servlet-name>  <servlet-class>NewServlet</servlet-class>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet>  <servlet-name>NewServlet2</servlet-name>  <servlet-class>NewServlet2</servlet-class>  <load-on-startup>2</load-on-startup>  </servlet>  Şimdi yapacağımız programda welcome-file-list kullanımıda içeriyor. Kullanıcı siteye ulaşmak istediğinde index.html dosyamız olduğu halde “Servlet” urlPattern’ına sahip servlet’e yönlendiriliyor kullanıcı. Bu servlet kullanıcıya cevap vermek için aşağıdaki kodu kullanıyor.  out.println( request.getMethod()+ " " + request.getRequestURL() +  ( (request.getQueryString() == null)?"":("?" +request.getQueryString()) ) );  Kullanıcıdan gelen request’in metodu(GET), kullanıcıdan gelen request’in URL’si (<http://localhost:8080/ShowURL/>) ve eğer kullanıcıdan gelen request’in query string’i yoksa boş string, varsa soru işaretinden sonra query string kullanıcıya verilecek response’a eklenir. Java dilinin bize sağladığı tüm formatted output kabiliyetlerini kullandık. Hatta println metodu içine bir conditional operator bile koyduk. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Üç iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 27-29/08/2018 | | | | | Sayfa No: 11 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Sıradaki programımızda welcome-file-list kullanmadığımız için ve index.html’e sahip olduğumuz için direk siteye gittiğimizde index.html’e ulaşıyoruz. Buradaki linki kullanarak servlet’imize ulaşabiliyoruz. Servletimiz bizim HTTP requestimizdeki header’ları bize gösteriyor. İlk olarak ne yaptığımızı kullanıcıya bildiriyoruz.  out.println("<p>HTTP headers sent by your client:</p>");  Header’ların isimlerini e isimli bir enum objesinde tutuyoruz.  Enumeration e = request.getHeaderNames();  Sonrada bir while döngüsü içerisinde e enum objemizde daha fazla eleman olduüu sürece bu elemanın ismini yazdırıyoruz ve sağına bu isme sahip requestteki HTTP header’ının değerini yazdırıyoruz.  while (e.hasMoreElements())  {  String headerName = (String) e.nextElement();  String headerValue = request.getHeader(headerName);  out.print("<b>" + headerName + "</b>: ");  out.println(headerValue + "<br>");  } | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Üç iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 27-29/08/2018 | | | | | | Sayfa No:12 | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servlet collaboration dinamik internet sitesi yapmada önemli bir kabiliyettir. Java’nın servlet teknolojisi bize RequestDispatcher ve SendRedirect kabiliyetlerini verir.  RequestDispatcher ile forward veya include seçeneklerini kullanabliriz. forward seçeneği, kullanıcıya verilecek response’un sorumluluğunu tamamen başka bir kaynağa verir. Mesela eğer kullanıcının girdiği şifre doğru ise, kullanıcıya sadece “WelcomeServletURL” servletinin verdiği cevabı göstermek isteyebiliriz. include seçeneği ise sadece işaret ettiğimiz kaynağın cevabını vermez kullanıcıya, zaten kullanıcıya verilecek cevaba bu kaynağın vereceği cevabıda ekler. Mesela eğer kullanıcı yanlış şifreyi girerse “login.html” kaynağının vereceği cevabı kullanıcıya verilecek cevaba ekleyip birde “şifre yada kullanıcı adını yanlış girdiniz” gibi bir mesajı yine kullanıcıya verilecek cevaba ekleyebiliriz.  String p = request.getParameter("userPass");  if (p.equals("servlet"))  {  RequestDispatcher rd = request.getRequestDispatcher("WelcomeServletURL");  rd.forward(request, response);  }  else  {  RequestDispatcher rd = request.getRequestDispatcher("login.html");  rd.include(request, response);  out.print("Username or password error.");  }  Sıradaki internet sitemizde iki adet servletimi olacak. index.html bir forma sahip olacak ve bu form verisini Simple UrlPattern’ına sahip servlet’e gönderecek. Form kullanıcı adı ve şifresine sahip olacak. Simple servlet’i kullanıcıdan gelen requestteki “userPass” ismine sahip parametreye bakacak. Eğer bu parametrenin değeri “servlet” ise “WelcomeServlet” UrlPattern’ına sahip servlet’e forward edilecek request. WelcomeServlet servlet’i response’a “Welcome ” ekleyecek ve kullanıcı adı parametresinin değerini alıp bunuda response’a ekletecek. Eğer parametrenin değeri “servlet” değilse bu requeste index.html’in verdiği cevap eklenecek, bununda üstüne bir hata mesajı eklenecek. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Üç iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 31/08/2018 | | | | | Sayfa No: 13 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servlet collaboration SendRedirect yöntemiylede yapılabilir. sendRedirect SendRedirect’in RequestDispatcher’ın forward seçeneğinden farkları:  -forward sadece server tarafında çalışabilir. sendRedirect client tarafındada server tarafındada çalışabilir.  -forward ayni request ve response objelerini başka bir kaynağa gönderir. sendRedirect yeni bir request gönderir.    sendRedirect kullanmak için önce Servlet’imizi dizayn ediyoruz.  <servlet>  <servlet-name>Servlet</servlet-name>  <servlet-class>Servlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>Servlet</servlet-name>  <url-pattern>/Servlet</url-pattern>  </servlet-mapping>  Bir index.html dosyamız var ama index.html dosyamızın olup olmaması servleti etkilemiyor. Biz urlPattern’ı kullanarak welcome-file-list kullanmadan servlet’imize ulaşabiliyoruz.  Servletimizdeki doGet metodu GET requestlerine cevap veriyor.  response.sendRedirect("http://www.google.com");  Bu komut, kullanıcının adına bu adrese bir request yapıyor.  sendRedirect hem relative url hemde absolute url kullanabilir. Yukarıdaki örnek absolute url’e bir örnek. Sadece html yada servlet dosyasının adını vermek bir relative url’e örnek. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Bir iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 3-7/09/2018 | | | | | Sayfa No: 14 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Bir arama motoru yapmak için kullanıcıdan sorgulamak istediği string’i almamaız gerekiyor.  İndex.html dosyamız hoşgeldin dosyası değil ama ismi index olduğu için server otomatik olarak bu dosyayı cevap olarak kullanıcıya sunuyor. index.html dosyamızda sorgu string’ini alacak bir form var. Bu form verisini GET metodu ile “Search” urlPattern’ına sahip servlet’e gönderiyor.  <form action="Search">  <input type="text" name="name">  <input type="submit" value="Search">  </form>  Servletimiz’e get metodu ile veri geliyor. Dolayısı ile get metoduna cevap verebilmesi için doGet metodunun doğru işi yaptığından emin olmamız gerekiyor.  doGet metodumuzda request ile gelen “name” isimli parametrenin değeri “name” isimli bir String tip değişkene(obje) atanıyor.  String name = request.getParameter("name");  Daha sonra sendRedirect kullanarak kullanıcının istediği sorgu sonucunu getiriyoruz. Bunu yapabilmek statik bir url’e istek göndermekle yapılamaz. Dolayısı ile Kullanıcının gönderileceği url’i dinamik olarak değiştiriyoruz. Kullanıcının sağladığı string’i Java’nın string concatenation özelliğini kullanarak “<https://www.google.com/search?q=>” stringinin sonuna ekliyoruz. Soru işareti, parametre adı ve eşittir işareti hazır. Tek eklenilen şey sorgu stringi. Bu eklenti yapıldıktan sonra sendRedirect çağırılıyor.  response.sendRedirect("https://www.google.com/search?q=" + name); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Beş iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 3-7/09/2018 | | | | | Sayfa No: 15 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Web container her servlet için bir ServletConfig objesi oluşturur. Bu obje web.xml dosyasından konfigürasyon bilgisi almak için kullanılabilir. Eğer konfigürasyon bilgisi web.xml dosyasından değiştirilirse servlet’e dokunmamıza gerek kalmaz. Dolayısı ile bilgiyi sisteme “hard code” etmemize gerek kalmaz ve web aplikasyonumuzun çeşitli içeriğini zamandan zamana değiştirmek daha kolay bir hale gelir.  Aşağıda gördüğümüz üzere servletimize isim ve bir java sınıfı atamak dışında şimdi birde Key-Value formatında parametre atadık. Parametremizin adı “driver” ve değeri “oracle.jdbc.driver.OracleDriver”.  <servlet>  <servlet-name>Servlet</servlet-name>  <servlet-class>Servlet</servlet-class>  <init-param>  <param-name>driver</param-name>  <param-value>oracle.jdbc.driver.OracleDriver</param-value>  </init-param>  </servlet>  index.html dosyamızda sadece Servletimize giden bir link var.  <a href="Servlet">See servlet config "driver"</a>  Servlet driver parametresini web.xml’den alır ve response’a ekler.  ServletConfig config = getServletConfig();  String driver = config.getInitParameter("driver");  out.print("Driver is: " + driver); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | | | İş yeri yetkilisinin imzası | | | | | | | | | | |
| Beş iş günü | | | |  | | | | | |  | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 3-7/09/2018 | | | | | Sayfa No: 16 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servlet config bilgilerini tek tek almak zorunda değiliz. Hepsini bir kerede alabiliz.  Öncelikle web.xml dosyamızda birden çok veri oluşturuyoruz.  <servlet>  <servlet-name>Servlet</servlet-name>  <servlet-class>Servlet</servlet-class>  <init-param>  <param-name>username</param-name>  <param-value>system</param-value>  </init-param>  <init-param>  <param-name>password</param-name>  <param-value>oracle</param-value>  </init-param>  </servlet>  index.html dosyamızdan link ile Servletimize gidiyoruz.  <a href="Servlet">See all servlet configs.</a>  Config objemizi oluşturup, tek bir isim verip değer istemek yerine config objemizi oluşturup tüm config isimlerini bir enum objesine atıyoruz.  ServletConfig config = getServletConfig();  Enumeration<String> e = config.getInitParameterNames();  web.xml dosyasındaki veri sayısı kadar (enumdaki veri sayısı kadar) döngü dönüyor, ger döngüde parametrenin ismi ve bu isme sahip parametrenin değeri response’a ekleniyor.  String str;  while (e.hasMoreElements()) {  str = e.nextElement();  out.print("<br>Name: " + str);  out.print(" value: " + config.getInitParameter(str));  } | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Beş iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 3-7/09/2018 | | | | | Sayfa No: 17 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| ServletContext ve ServletConfig arasında fark, serverda sadece bir ServletContext objesi var. Web aplikasyonundaki tüm servletler bu objesi kullanabilir. web.xml deki değerleri alabiliriz, varolan bir değeri değiştirebiliriz yada silebiliriz. Ayrıca aplikasyonlar arası veri iletişimindede kullanılabilir.  web.xml de servletin bir parçası olarak değil, projenin bir parçası olarak ServletContext’imizi oluşturuyoruz.  <servlet>  <servlet-name>Servlet</servlet-name>  <servlet-class>Servlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>Servlet</servlet-name>  <url-pattern>/Servlet</url-pattern>  </servlet-mapping>    <context-param>  <param-name>dname</param-name>  <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>  </context-param>  Servletimiz bir config değil bir context objesi kullanıyor context’lere erişmek için.  ServletContext context = getServletContext();  “dname” ismine sahip context parametresinin değerini “driverName” ismindeki String değişkenine(obje) atıyoruz ve response’a ekliyoruz.  String driverName = context.getInitParameter("dname");  pw.println("driver name: " + driverName); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Beş iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 3-7/09/2018 | | | | | Sayfa No: 18 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Birden çok context parametresinede sahip olabiliriz. Bu değerleri web.xml dosyamızda belirtmemiz gerekiyor.  <context-param>  <param-name>dname</param-name>  <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>  </context-param>  <context-param>  <param-name>username</param-name>  <param-value>system</param-value>  </context-param>  <context-param>  <param-name>password</param-name>  <param-value>oracle</param-value>  </context-param>  Servletimizde context objemizi yaratıyoruz.  ServletContext context = getServletContext();    Sadece bir context değerini sorgulamak yerine tüm contextlerin listesini alıyoruz ve bir enum objesine atıyoruz.  Enumeration<String> e = context.getInitParameterNames();  Tek tek tüm context’lerin isimlerini(enumdan) ve bu isme sahip contextlerin değerlerini(context objemizle) bulup response’a ekliyoruz.  String str = "";  while (e.hasMoreElements())  {  str = e.nextElement();  out.print("Name: " + str);  out.print(" value: " + context.getInitParameter(str) + "<br>");  } | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | |
| Beş iş günü | | | |  | | | |  | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 19 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Veriyi bir xml dosyasında tutmak yerine veriyi bir ServletContext objesinde tutabiliriz. Objeye veriyi xml’den vermek yerine direk kod üzerinden veriyoruz.  İnternet sitemizde iki servlet var. Birincisi veriyi giriyor, verinin girildiğini kullanıcıya belirtiyor ve ikinci servlet’e bir link sunuyor.  try {  ServletContext context = getServletContext();  context.setAttribute("company", "IBM");  out.println("Attrubute has been set.");  out.println("<a href='Servlet2'>visit</a>");  out.close();  }  catch (Exception e) {  out.println(e);  }  İkinci servletimiz context objesini yaratıyor. Company ismine sahip context’in değerini alıyor, ismini ve değerini response’a ekliyor.  try {  ServletContext context = getServletContext();  String n = (String) context.getAttribute("company");  out.println("Attribute name: company<br>");  out.println("Attribute value: " + n);  out.close();  }  catch (Exception e) {  out.println(e);  } | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | | |
| Beş iş günü | | | |  | |  | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 20 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Session tracking denilen sistem kullanıcı hakkında veri tutma için kullanılır. HTTP protokolü stateless olduğu için state tutma işi bize düşüyor. Bu yöntemlerden birisi cookie’dir. Cookie’ler browser kapatıldığında silinen tiptende olabilir (non-persistent) cookie, belirli bir süre sonra silinen tipte olabilir (persistent cookie). Cookie’lerin dezavantajı eğer kullanıcı tarayıcıdan cookie’leri kapattıysa işe yaramaz.  Cookie kullanan bir site yapalım şimdi. index.html dosyamız Servlet1’i hedef alan bir forma sahip. Post metodunu kullandığınada dikkat edelim. Kullanıcı adını girip butona basıyoruz.  <form action="Servlet1" method="post">  Name:<input type="text" name="userName"/><br/><br/>  <input type="submit" value="Servlet1"/>  </form>  Servlet1’de userName parametresinin değeri alınır ve response’a eklenir.  String n = request.getParameter("userName");  out.print("http request parameter \"userName\": " + n);  Yeni bir cookie objesi oluşturulur bu isme ve değere sahip. Bu cookie’de response’a eklenir. Ayrıca kullanıcıya bildirilir yapılanlar.  Cookie ck = new Cookie("userName", n);  response.addCookie(ck); // Add cookie to the response  out.print("<br/><br/>Created cookie with cookie name: \"" + ck.getName() +  "\" and cookie value: \"" + ck.getValue() + "\" and added cookie to reponse.");  Daha sonra servlet2’ye giden bir buton ekler response’a.  out.print("<br/><br/><form action='Servlet2' method='post'>");  out.print("<input type='submit' value='Servlet2'>");  out.print("</form>"); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | | |
| Beş iş günü | | | |  | |  | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 21 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Servlet2 kendisine gelen request ile birlikte gelen cookie’ler arasından “userName” ismine sahip cookie varmı yokmu ona bakar. request ile gelen tüm cookie’leri bir cookie dizisine koyar. Bu bu dizi üzerinde bir for döngüsü kullanarak ilerler. Eğer “userName” ismine(değerine değil) sahip bir cookie varsa bu cookie’nin adı ve değeri response’a eklenir.  out.print("Searching through cookies that were sent with request to find "  + "the cookie named \"userName\"<br/><br/>");  // Get cookies and print the one with the name "userName"  Cookie ck[] = request.getCookies();  for (int i = 0; i < ck.length; i++)  {  if ("userName".equals(ck[i].getName()))  {  out.print("Cookie name: " + ck[i].getName() +  "<br/>Cookie value: " + ck[i].getValue());  }  }  Daha sonra servlet3’e giden bir buton response’a eklenir.  // Creating a button link to Servlet3  out.print("<br/><br/><form action='Servlet3' method='post'>");  out.print("<input type='submit' value='Servlet3'>");  out.print("</form>");  Servlet3 userName isminde bir cookie yaratır ve değerini boş string olarak seçer, cookinin yaşınıda 5 saniye olarak seçer. Bu cookie’yi response’a ekler ve kullanıcıya bildirir.  response.setContentType("text/html");  PrintWriter out = response.getWriter();  // Deleting value of cookie "userName"  Cookie ck = new Cookie("userName", "");  out.print("Cookie's value has been deleted.");  ck.setMaxAge(5); // Changing the maximum age to 0 seconds  out.print("<br/>Cookie's max age has been set to 5 seconds.");  response.addCookie(ck); // Adding cookie in the response | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | | |
| Beş iş günü | | | |  | |  | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 22 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Cookie ile login yapabildiğimiz bir internet sitesi yapacağız.  Hiçbir html dosyasına gitmeden önce “Index” servletine gidiyoruz. Welcome-file-list’i ekran çıktısından görüyoruz. Index servletinde (ekran çıktısında gördüğümüz üzere) cookie listesi alınıyor. For döngüsüyle cookie’lerden birisinin adı “name” mi değilmi bakılıyor. Eğer cookie’lerden birisinin adı name ise index2.html’e(giriş yapmış kullanıcının görmesi gereken ekran) yönlendiriliyor kullanıcı ve “loggedIn” isimli boolean değişken true yapılıyor. Yani kullanıcı giriş yapmış anlamına geliyor bu. Döngüden çıkılıyor, loggedIn değişkeni false ise kullanıcı index.html’e yönlendiriliyor. Yani cookilerden birisinin ismi “name” değilse ilk dediğimiz yerine bu oluyor. Ayrıca birde try catch durumu var. Eğer hiçbir cookie yoksa kullanıcının bilgisayarında bizim sitemiz ile ilgili, hata vermek yerine giriş yapmadığını anlayıp index.html’e yönlendiriyoruz (giriş yapmamış kişinin görmesi gereken ekran).  index.html admin login linkine sahip. Bu bizi login.html’e götürüyor. login.html bir tabloya sahip. Kullanıcı adı ve şifre girilebilen bir form bu tabloda düzgünce duruyor. Buton bizi Login servletine götürüyor. Login servleti girdiğimiz şifreye bakıyor, eğer şifre “admin” ise response’a link2.html’i ekliyor. link2.html header dediğimiz ekranın üstündeki linklerden oluşan kısım. link2.html giriş yapmış kullanıcıların görmesi gereken header. Homepage (index2.html), Logout(servlet) ve Profile(servlet) linklerine sahip. link2.html’i response’a ekledikten sonra “name” isimli parametrenin değerini alıp “name” ismine sahip ve kullanıcının girdiği kullanıcı adı değerine sahip bir cookie yaratılıp response’a ekleniyor. Sonra giriş yapıldı yazısı response’a ekleniyor. Eğer kullanıcının girdiği şifre admin değilse link.html header olarak kullanılıyor. Giriş yapmamış kişinin görmesi gereken header. Homepage (index.html) ve login(login.html) linklerine sahip. link.html’i header olarak ekledikten sonra birde login.html’i ekliyor tekrar giriş denemesi yapılsın diye ve kullanıcı adı veya şifre hatası yazısını ekliyor response’a.  Logout servleti ilk olarak link.html’i response’a ekliyor. Çünkü giriş yapmamış bir kişide bu servlet’e gelse, giriş yapmış bir kişide bu servlete gelse aynı link.html header’ını görmeli. Daha sonra name ismine sahip bir cookie varmı bakılıyor. Varsa cookie siliniyor ve çıkış yapıldı yazısı yazılıyor. Yoksa giriş zaten yapılmadı yazılıyor. Hiçbir cookie yoksada zaten giriş yapılmadı yazılıyor.  Profile servleti ise yine cookiler üzerinden name ismine sahip cookie varmı bakıyor. Yoksa veya hiç cookie yoksa giriş yapılmadı diyor. Varsa profil ekranını gösteriyor. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | | |
| Beş iş günü | | | |  | |  | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 23 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Bu sitede hidden form field tekniği ile state tutacağız. Cookie’ye göre iyi yanı, browser ayarına bağlı olmaması.  index.html’deki form kullanıcı adı alıyor ve Servlet1’e yolluyor.  <form action="Servlet1">  Name:<input type="text" name="userName"/><br/>  <input type="submit" value="go"/>  </form>  Servlet1 userName parametresini alıyor ve hosgeldin userName diyor.  String n = request.getParameter("userName");  out.print("Welcome " + n);  Daha sonra bir form koyup servlet2 ye giden bir buton ekliyor. Ama bu formda gizli veri var.  //creating form that have invisible textfield  out.print("<form action='Servlet2'>");  out.print("<input type='hidden' name='uname' value='" + n + "'>");  out.print("<input type='submit' value='go'>");  out.print("</form>");  Servlet2 uname parametresini gizli gönderilmiş olsada alabiliyor ve n isimli string değişkenine atıyor. Daha sonrada ekrana “Hello n” yazılıyor.  //Getting the value from the hidden field  String n = request.getParameter("uname");  out.print("Hello " + n); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | | |
| Beş iş günü | | | |  | |  | | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 24 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Bu sitede URL rewriting tekniğiyle state tutacağız. Cookie’ye göre iyi yanı, browser ayarına bağlı olmaması.  index.html forma sahip ve formda userName textfield’a sahip. veri Servlet1’e gidiyor.  <form action="Servlet1">  Name:<input type="text" name="userName"/><br/>  <input type="submit" value="go"/>  </form>  Servlet1 userName parametresinin değerini alıyor ve n isimli string değişkenine atıyor ve ‘Welcome n” yazıyor ekrana.  String n = request.getParameter("userName");  out.print("Welcome " + n);  Son olarak bir link koyuyor ekrana. Linki kendisi gizli olarak editliyor. index.html’den aldığı string’i Servlet2’te giderken query string te kullanıyor.  //appending the username in the query string  out.print("<br><br><a href='Servlet2?uname=" + n + "'>visit</a>");  Servlet2 uname parametresinin değerini alıyor ve n isimli bir stringe atıyor ve ekrana hello n yazdırıyor.  //getting value from the query string  String n = request.getParameter("uname");  out.print("Hello " + n); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | |
| Beş iş günü | | | |  | | |  | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 10-14/09/2018 | | | | | Sayfa No: 25 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Bu sitede Session ile state tutacağız. Sessionlar tarayıcıyı kapadığında silinir. Server tarafında tutulurlar. iİnvalidate ederek silinebilirler. Bir kullanıcı başına sadece bir session olabilir cookiler aksine.  index.html dosyamızda formumuz var. userName isimli text field alınıyor ve Servlet1 servletine veriliyor get metoduyla.  <form action="Servlet1">  Name:<input type="text" name="userName"/><br/>  <input type="submit" value="go"/>  </form>  Servlet1 userName parametresinin değerini alıyor, n isimli bir string’e atıyor ve “welcome n” yazdırıyor.  String n = request.getParameter("userName");  out.print("Welcome " + n);  Bir session objesi yaratıyor ve uname ismine ve kullanıcıdan aldığımız değere sahip bir değişken ekliyor session objesine.  HttpSession session = request.getSession();  session.setAttribute("uname", n);  En son servlet2 ye bir link koyuyor.  out.print("<br><br><a href='Servlet2'>visit</a>");  Servlet2 request’in sessionu varmı bakıyor. Varsa session’un uname değişkeninindeğerini alıp n isimli bir string değişkenine atıyor ve “Hello n” yazdiriyor.  HttpSession session = request.getSession(false);  String n = (String) session.getAttribute("uname");  out.print("Hello " + n); | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | |
| Bir iş günü | | | |  | | |  | | | | | | | | | | | | | |
| İş Yerinde Çalışılan Birim: Yazılım | | | | | | | | | | | Tarih: 17/09/2018 | | | | | Sayfa No: 26 | | | | |
| Yapılan iş: Servlet ile web applikasyonu yazıldı. | | | | | | | | | | | | | | | | | | | | |
| Sessıon ile login yapabildiğimiz bir internet sitesi yapacağız.  Hiçbir html dosyasına gitmeden önce “Index” servletine gidiyoruz. Welcome-file-list’i ekran çıktısından görüyoruz. Index servletinde (ekran çıktısında gördüğümüz üzere) request’in session’u varmı bakılıyor. Session varsa index2.html’e(giriş yapmış kullanıcının görmesi gereken ekran) yönlendiriliyor kullanıcı. Eğer session yok ise kullanıcı index.html’e yönlendiriliyor.  index.html Admin Login linkine sahip. Bu bizi login.html’e götürüyor. login.html bir tabloya sahip. Kullanıcı adı ve şifre girilebilen bir form bu tabloda düzgünce duruyor. Buton bizi Login servletine götürüyor. Login servleti girdiğimiz şifreye bakıyor, eğer şifre “admin” ise response’a link2.html’i ekliyor. link2.html header dediğimiz ekranın üstündeki linklerden oluşan kısım. link2.html giriş yapmış kullanıcıların görmesi gereken header. Homepage (index2.html), Logout(servlet) ve Profile(servlet) linklerine sahip. link2.html’i response’a ekledikten sonra “name” isimli parametrenin değerini alıp “name” ismine sahip ve kullanıcının girdiği kullanıcı adı değerine sahip bir session değişkeni yaratılıyor. Sonra giriş yapıldı yazısı response’a ekleniyor. Eğer kullanıcının girdiği şifre admin değilse link.html header olarak kullanılıyor. Giriş yapmamış kişinin görmesi gereken header. Homepage (index.html) ve login(login.html) linklerine sahip. link.html’i header olarak ekledikten sonra birde login.html’i ekliyor tekrar giriş denemesi yapılsın diye ve kullanıcı adı veya şifre hatası yazısını ekliyor response’a.  Logout servleti ilk olarak link.html’i response’a ekliyor. Çünkü giriş yapmamış bir kişide bu servlet’e gelse, giriş yapmış bir kişide bu servlete gelse aynı link.html header’ını görmeli. Daha sonra request’in bir session’u varmı bakılıyor. Varsa session invalidate ediliyor ve çıkış yapıldı yazısı response’a ekleniyor. Session zaten yoksa giriş yapılmadı yazısı response’a ekleniyor.  Profile servleti ise yine request’in session’u varmı ona bakıyor. Varsa link2.html header’ını response’a ekliyor ve profil ekranını response’a ekliyor. Yoksa giriş yapılmadı yazısını response’a ekliyor. | | | | | | | | | | | | | | | | | | | | |
| Yapılan işin süresi | | | | Stajyerin imzası | | | İş yeri yetkilisinin imzası | | | | | | | | | | | | | |
| Bir iş günü | | | |  | | |  | | | | | | | | | | | | | |
| **STAJ KAZANIMLARI**  Java konusunda uzmanlaşmak üzere çalışan birisi olarak kasıtlı olarak birinci ve ikinci stajlarım Java üzerine oldu. Aldığım Java kitabından ve internetten çeşitli sitelerden çalışarak kendimi geliştirebildim. Okulda web programlama dersinde php ve asp.net gösterildiği için stajlar java teknolojileri öğrenme ve deneyim kazanma konusunda çok yardımcı oldu. Stajlarda gördüğüm teknolojiler kendime çizdiğim yolda düzeltmeler yapmamı sağladı.  İş yeri tecrübesi edindim, insanlar gördüm. Nasıl işlerde çalışmak istiyorum, yüksek lisansı hangi ülkede ve üzerine yapmak istiyorum, yüksek lisanstan sonra hangi ülkeye yerleşmek istiyorum sorularına cevapta bulmuş oldum. Hangi teknolojiler önemli, hangi teknolojiler kullanılıyor hangileri kullanılmıyor, hangi teknolojileri öğrenmeliyim sorularına cevap buldum.  Servlet konusunda kendimi çok geliştirdim. Dinamik web programla dünyasına girince internetin nasıl çalıştığı konusunda çok bilgi edindim. HTML konusundada pratik yapmış ve daha önce bilmediğim şeyleri öğrenmiş oldum. | | | | | | | | | | | | | | | | | | |

|  |
| --- |
| **KAYNAKLAR**  DEITEL, Paul, DEITEL, Harvey, *Java How to Program*, 07458, 0-13-380780-0, Pearson, USA, One Lake Street, Upper Saddle River New Jersey, 2015  HTTP POST request parameters,  <https://stackoverflow.com/questions/14551194/how-are-parameters-sent-in-an-http-post-request>  Query string, <https://en.wikipedia.org/wiki/Query_string>  Uniform Resource Name, <https://en.wikipedia.org/wiki/Uniform_Resource_Name>  MIME types, <https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types>  Difference between servlet containers and full java EE servers, <https://stackoverflow.com/questions/3821640/what-is-the-difference-between-tomcat-jboss-and-glassfish>,  <https://www.quora.com/Whats-the-difference-between-Glassfish-and-Apache-Tomcat-Are-they-widely-used> |
| **EKLER**  Yapılan işlerin ekleri, yapılan işlerin yazılı anlatımından hemen sonra konuldu. |