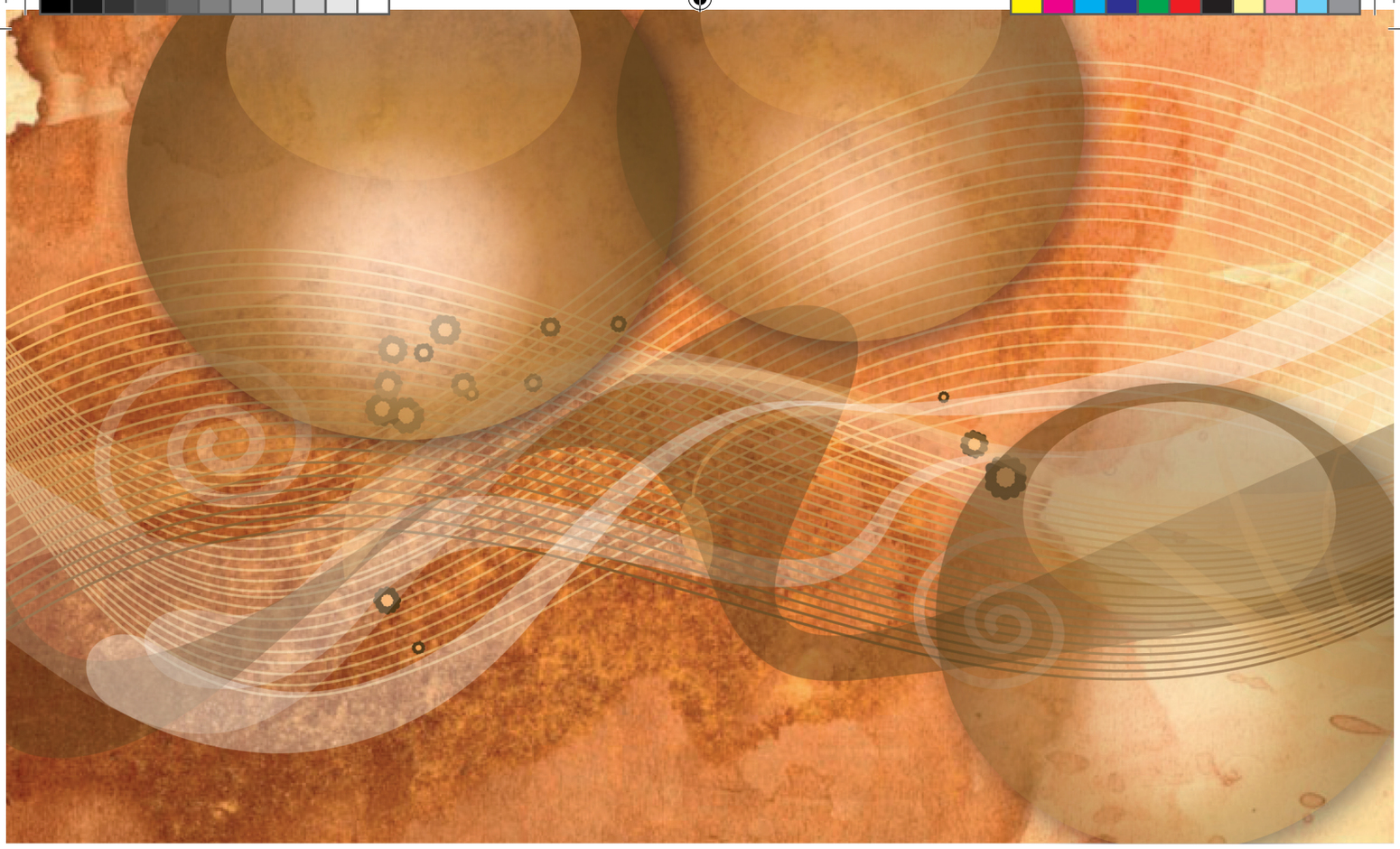
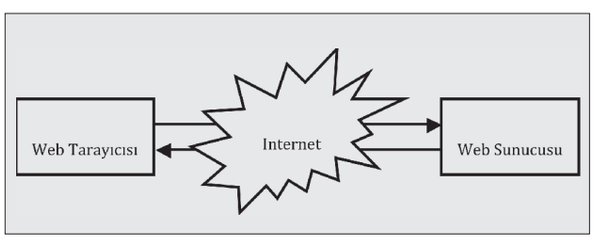


JAX-RS 1.1 ile REST

SOAP tabanlı web servislerin yazılım ve kullanım güçlüklerinden kaynaklanan sebeplerden dolayı REST tabanlı web servis mimarileri son zamanlarda popüler hale gelmiştir. REST mimarileri, kullanımı ve geliştirilmesi kolay web servis modüllerinin oluşturulmasını mümkün kılmaktadır. JAX-RS 1.0 ile Java dünyasına giren REST mimarileri, JAX-RS 1.1 olarak Java EE 6'nın bir parçası haline gelmiştir. Bu yazımda sizlere Java dünyasında JAX-RS 1.1 ile REST mimarilerinin nasıl uygulandığını aktarmak istiyorum.

Özcan Acar

REST (Representational State Transfer) 2000 yılında Roy Thomas Fielding'in [1] hazırladığı doktora tezinde tanımladığı bir yazılım mimarisi tarzıdır. REST ile HTTP (Hypertext Transfer Protocol) protokolünü kullanan ve kullanımı ve yayılımı diğer web servis teknolojilerine (SOAP, WSDL) oranla daha kolay olan web servisleri oluşturmak mümkündür. REST, HTTP protokolüne bağımlı olmamakla beraber, en çok HTTP protokolü ile beraber kullanılmaktadır. SOAP gibi diğer web service teknolojileri HTTP protokolünü sadece verilerin transferi için kullanırken, REST tamamen HTTP protokolünün sunduğu metotlara odaklanarak, HTTP tabanlı web servislerin oluşturulmasını mümkün kılmaktadır. REST mimarisi ile oluşturulan web servislerine RESTful web servisleri ismi verilmektedir. REST mimarisi anlayabilmek için HTTP protokolünü yakından tanımak gerekmektedir.



Resim 1 HTTP istek / cevap (request/response) modeli

HTTP - Hypertext Transfer Protocol

Internet'in bugünkü haliyle popüler olmasını sağlayan HTTP protokolüdür. Web sayfalarında yer alan içeriğe erişim HTTP protokolü ile sağlanır. Resim 1'de görüldüğü gibi web tarayıcımızı açıp bir web adresine bağlanmak istediğimizde, web tarayıcısı, web sayfasının bulunduğu web sunucusuna HTTP protokolü ile bağlanır.

HTTP Metodları

Web tarayıcısı, web sunucusunda bulu-

nan bir web sayfasına erişmek için HTTP protokolünde bulunan GET metodunu kullanılır. Örneğin http://www.javadergisi.com/index.html web sayfasına erişmek istediğimizde, kullandığımız web tarayıcısı web sunucusuna GET index.html komutunu gönderir. Bunun yanısıra HTTP bünyesinde POST, PUT, DELETE, HEAD, TRACE, OPTIONS ve CONNECT metotları bulunmaktadır.

GET

Bir web sunucusunda bulunan kaynagı (resource) edinmek için GET komutu kullanılır. Bu kaynak bir HTML web sayfası, bir JPEG resmi, bir PDF dosyası ya da bir program olabilir. GET komutu işlem esnasında kaynağı değiştirmez, sadece kaynağın içeriğini edinmek için kullanılır.

POST

REST mimarisinde POST metodu yeni bir kaynağın oluşturulması (create) için kullanılır. Örneğin POST metodunu kullanarak sanal bir kütüphane sistemine yeni bir kitap ekleyebiliriz.

PUT

REST mimarisinde PUT mevcut bir kaynağı değiştirmek (update) için kullanılır. Örneğin PUT metodunu kullanarak sanal bir kitaphane sisteminde bulunan bir kitabın sahip olduğu verileri değiştirebiliriz.

DELETE

REST mimarisinde DELETE mevcut bir kaynağı silmek için kullanılır. Örneğin DELETE metodunu kullanarak sanal bir kitaphane sisteminde bulunan bir kitabı sistemden silebiliriz.

Diğer HTTP Metotları

Daha öncede bahsettiğim gibi GET, POST, PUT ve DELETE haricinde HTTP metotları mevcuttur. Pek sık kullanılmayan bu metotlar şunlardır:

- HEAD: GET gibi çalışmakla beraber, kaynağın içeriğini edinmek için kullanılmaz. Bir kaynağın varlığını kontrol etmek için kullanılır.
- TRACE: Kullanıcı tarafından gönderilen verileri birebir geri göndermek (echo) için kullanılır.
- OPTIONS: Web sunucusunun ya da erişilmek istenen kaynağın ihtiva ettiği iletişim seçeneklerini tespit etmek için kullanılır.
- CONNECT: Bir proxy üzerinden tünelleme (tunnel) işlemleri için kullanılır. Genelde HTTP protokolünün başka bir protokole tünellenmesi işleminde kullanılır.

REST Mimarisi

Her yazılım mimarisi türünün kendine has öğeleri vardır. REST mimarisi de kendine has mimarik öğeler tanımlamaktadır. REST mimarisini kullanabilmek için bu öğeleri yakından tanıyalım.

Kaynak (Resource)

REST mimarilerinde oluşturulan web servislere kaynak ismi verilir. Kaynaklar REST mimarilerinde merkezi bir rol oynarlar. Değişik HTTP metodları kulllanılarak bu kaynaklar üzerinde işlem yapılır.

Kaynak İdentifikatörleri (URI - Unified Resource Identifier)

HTTP protokülü üzerinden bir kaynağa erişimi sağlamak için kaynak identifikatörleri kullanılır. Kaynak identifikatörleri ile kaynakları doğrudan adreslemek mümkünleşir.

Kaynak Representasyonu (Resource Representation)

Kaynak representasyonu, kaynak ile olan iletişimin hangi veri formatında olması gerektiğine işaret eder. Örneğin bir kaynaktan edinilen veri XML, PDF ya da text formatında olabilir. Bir kaynak ile iletişim, o kaynağın sahip olduğu kaynak representasyonu aracılığıyla gerçekleşir.

JAX-RS 1.0 Spesifikasyonu

Prensipte REST tabanlı yazılım sistemlerini JDK 1.1 ile tanıştığımız Servlet teknolojisi ile oluşturmak mümkündür. Sonuç olarak REST tabanlı bir mimari için ihtiyaç duyduğumuz öğeler HTTP'ye dayalı web standartları ve URI mekanizmalarıdır. Servlet teknolojisini ve Apache Tomcat gibi bir aplikasyon serverini kullanarak çok kolayca REST tabanlı bir mimari oluşturabiliriz.

REST mimarisine olan ilginin artmasıyla Sun firması subat 2007'de JSR 311 [2] (Java Specification Requests) ile REST mimarisini Java bünyesinde standardize etmek için adım attı. Bu çalışmaların sonunda ekim 2008'de JAX-RS 1.0 (Java API for RESTful Web Services) isminde yeni bir

```
package server;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;

@Path(„book“)
public class BookResource
{
   @GET
   @Produces(„application/xml“)
   @Path(„{id}“)
   public String getBook(@PathParam(„id“) String id)
   {
            return „<?xml version=\“1.0\“?>“ + „<id> „+id +
„</id>“;
   }
}
```

Kod 1

```
@POST
@Consumes(„application/xml“)
@Produces(„text/plain“)
public String createBook(Book book)
{
            System.out.println(book.getYazar());
            return „yeni kitap no: 100000“;
}
```

Kod 2

```
@PUT
@Path(„{id}“)
@Produces(„application/xml“)
public Book updateBook(@PathParam(„id“) String
id)
{
            return new Book(„Orhan Pamuk“,
„Test“);
}
```

Kod 3

spesifikasyon oluştu. Kısa bir zaman sonra bu spesifikasyonu implemente eden Jersey [5], JBoss RESTEasy [3] ve CXF [6] gibi açık kaynaklı frameworkler ile tanıştık.

JAX-RS 1.1 Spesifikasyonu

Java dünyasında hemen hemen hiç bir spesifikasyon ilk haliyle kalmaz ve zaman içinde değişikliğe uğrar. Bu JAX-RS spesifikasyonu için de geçerli olan bir durumdur. Aralık 2009'da kullanıma sunulan Java EE 6 platformu bünyesinde JAX-RS 1.1 spesifikasyonu da yer almaktadır. Böylece JAX-RS Java EE'nin (Enterprise Edition) bir parçası haline gelmiş ve kurumsal projelerde kullanım için standardize edilmiştir. JAX-RS 1.1'in beraberinde getirdiği yenilikler JCP'in (Java Community Process) JAX-RS changelog [4] bölümünde yer almaktadır. Daha öncede belirttiğim gibi 1.1 ile gelen en büyük değişiklik, JAX-RS'in Java EE 6'nın bir parçası haline gelmiş olmasıdır. Bunun yanısıra Stateless Session EJB'leri REST kaynağı (resource) olarak kullanmak mümkündür. Ayrıca persistence manager, EJB, data source gibi kaynakları REST kaynaklarına anotasyonlar yardımı ile enjekte etmek (dependency injection) mümkündür.

JAX-RS 1.1 spesifikasyonunu implement eden ilk açık kaynaklı framework Jersey'dir [5]. Jersey ayrıca Sun'ın referans implementasyonudur. Referans implementasyonu, JAX-RS gibi kağıt üzerindeki bir spesifikasyonu çalışır bir program ya da framework haline getiren, spesifikasyonun nasıl kullanılması gerektiğini örnekleyen ilk implementasyondur.

REST Tarzı Yazılım

REST mimarisinde yer alan konseptleri bir web aplikasyona nasıl uygulayabileceğimizi bir örnek üzerinde yakından inceleyelim. Örneğin REST tabanlı bir kütüphane yönetim programında, bir kitabın sisteme eklenmesi, silinmesi, değiştirilmesi ve kitap bilgilerinin edinilmesi aşağıdaki şekilde gerçekleşecektir:

- GET metodu ile http://localhost/book/99999 adresinden 99999 nolu kitap hakkında bilgi edinebiliriz. Burada kaynak identifikatörü /book/99999'dur.
- Sisteme yeni bir kitap eklemek istedigimizde POST metodunu kullanarak XML, JSON ya da text formatında kitap verilerini http://localhost/book adresine gönderek, sisteme yeni bir kitap eklenmesini sağlayabiliriz. Kitap sisteme eklendikten sonra POST metodu 99999 rakamanı ya da başka bir rakamı kitap nosu olarak geri döndürecektir.
- DELETE metodu ile http://localhost/book/99999 adresinde yer alan bir kitabı sistemden silebiliriz.
- PUT metodunu kullanarak http://localhost/book/99999 adresinde yer alan bir kitabın verileri üzerinde değişiklik yapabiliriz.

HTTP metotlarını kullanarak Create/Oluştur (PUT), Read/Edin (GET), Update/Değiştir (PUT) ve Delete/Sil (DELETE) – (CRUD) operasyonlarını gerçekleştirebiliriz. Birçok yazılım sisteminde de zaten ön planda olan CRUD (Create/Read/Update/Delete) işlemleridir.

```
@XmlRootElement
public class Book
{
        private String yazar;
        private String isim;

        public Book(String pYazar, String pIsim) {
                this.yazar = pYazar;
                this.isim = pIsim;
        }

        public Book() {
        }

        public String getYazar() {
                return yazar;
        }
        public void setYazar(String yazar) {
                this.yazar = yazar;
        }
        public String getIsim() {
                return isim;
        }
        public void setIsim(String isim) {
                this.isim = isim;
        }
}
```

Kod 4

RESTful Web Servisleri için Java API'si

AX-RS 1.1 spesifikasyonunu yazılımda configuration by exception (istisnai durumlarda konfigürasyon) yöntemini kullanmaktadır. Birçok tanımlama Java sınıfları bünyesinde anotasyonlar kullanılarak, XML konfigürasyon dosyalarına ihtiyaç kalmadan yapılabilmektedir.
Başlıca JAX-RS anotasyonları şu şekildedir:

- @Path: Bu anotasyon ile bir REST kaynağının URI'si tanımlanır.
- @Get: Bu anotasyonla işaretli metotlar HTTP GET metoduna hizmet verirler.
- @Post: Bu anotasyonla işaretli metotlar HTTP POST metoduna hizmet verirler.
- @Put: Bu anotasyonla işaretli metotlar HTTP PUT metoduna hizmet verirler.
- @Delete: Bu anotasyonla işaretli metotlar HTTP DELETE metoduna hizmet verirler.
- @Produces: Bu anotasyon kullanıcıya gönderilecek olan verinin tipini belirler. Geri döndürülen verinin tipi XML, JSON ya da text formatında olabilir.
- @Consumes: Bu anotasyon, bu anotasyon ile işaretli metodun verileri hangi formatta kabul ettiğini tanımlar. Metodun kabul ettiği veriler XML, JSON ya da text formatında olabilir.

Kod 1'de yer alan basit bir POJO (Plain Old Java Object) Java sınıfıdır. JAX-RS anotasyonları kullanılarak bir REST kaynağı haline dönüştürülmüştür. @Path anotasyonu kullanılarak REST kaynağının /book URI'sinden erişimi sağlanmıştır. Sınıf bünyesinde bulunan getBook() metoduna erişim @Get anotasyonu ile sağlanmıştır. Eger HTTP GET metodu kullanılırsa, kullanıcı isteği @Get anotasyonu ile işaretli metoda delege edilir. @Produces anotasyonu ile kullanıcıya gönderilen verinin tarzı belirlenmektedir. Örneğin kullanıcı http://localhost/book/99999 şeklinde bir istekte (request) bulundu ise, kullanıcıya geri döndürülen değer XML formatında ve „<?xml version="1.0"?><id>99999</id>" şeklinde olacaktır. @Produces anotasyonu kullanılarak XML, JSON ya da text formatında veriler geri döndürülebilir. @Path("{id}") ve @PathParam("id") anotasyonları ile URI'ye eklenmiş olan parametrenin getBook() metoduna iletilmesi sağlanır. Örneğin /book/11111 URI'sinde yer alan 11111 parametresi @PathParam anotasyonu ile bir metod parametresi ha-

line dönüştürülür ve metot bünyesinde değişken olarak kullanılabilir.

Kod 2'de yer alan createBook() metodu @Post anotasyonu ile işaretlenmiştir. Bu method HTTP POST metoduyla yapılan bir işleme cevap verir. JAX-RS anotasyonu olan @Consumes ile createBook() metodunun verileri XML formatında kabul ettigi ifade edilir. JAX-RS implementasyonu JAXB (Java for XML Binding) [7] yardımı ile XML ve Java sınıfları arasında transformasyon yapabilir. createBook() metodu örneğinde HTTP POST kullanılarak bir kitabı temsil eden XML verileri bu metoda iletilmekte ve JAX-RS tarafından bu XML verileri bir Book nesnesine dönüştürülerek, metot parametresi olarak kullanılmaktadır.

Kod 3'de yer alan updateBook() metodu @Put anotasyonu ile işaretlenmiştir. Bu metot HTTP PUT metoduyle yapılan bir işleme cevap verir. Bu örnekte @Produces anotasyonu ile kaynak representasyonunun XML formatında olması sağlanmıştır. Örnekte görüldüğü gibi verileri XML formatında kodlamak yerine, bir Book nesnesi oluşturulmakta ve geri döndürülmektedir. JAX-RS implementasyonu JAXB yardımıyla otomatik olarak bu Book nesnesinin XML karşılığı olan veri transformasyonunu gerçekleştirmektedir. JAXB ile bu transformasyonun yapılabilmesi için Book sınıfında bir JAXB anotasyonu olan @XmlRootElement'in kullanılması gerekmektedir. Book sınıfı kod 4'de yer almaktadır.

RESTful Web Servisleri için Kullanıcı (Client) API'si

JAX-RS 1.1 implementasyonu olan Jersey bünyesinde, bir REST kaynağı ile interaksiyonu sağlayacak bir kullanıcı API'si bulunmaktadır.

Bir REST kaynağı ile iletişimi sağlamak için ayrıca Apache HTTP Client [8] gibi bir framework kullanılabilir.

Sonuç

REST mimarileri web servis yazılımını daha kolay hale getirmektedir. JAX-RS 1.0 ile Java dünyasına giren REST mimarileri, JAX-RS 1.1 ile Java EE 6'nın bir parçası haline gelmiş ve standartlaşmıştır. Java dünyasında REST tabanlı mimariler oluşturmak bu girişimler sonunda çok kolaylaştırılmış ve sadeleştirilmiştir. Özellikle anotasyon güdümlü yazılım, istisnai durumlarda xml konfigürasyon dosyalarının kullanılmasını mümkün kılmaktadır. JAX-RS 1.1 ile REST mimarilerinin Java dünyasındaki popülerliği daha da artacaktır. ■

```
import java.net.URI;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.UriBuilder;
import server.Book;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;

public class RestClient {
        public static void main(String[] args) {
                ClientConfig config = new DefaultClientConfig();
                Client client = Client.create(config);
                WebResource service = client.resource(getBaseURI());
                System.out.println(service.path(„/book/99999“).accept(
                                MediaType.TEXT_PLAIN).get(String.class));
                Book book = service.path(„/book/9999“)
                                .accept(MediaType.APPLICATION_XML).put(Book.class);
                System.out.println(book.getYazar());
        }

        private static URI getBaseURI() {
                return UriBuilder.fromUri(„http://localhost:8084/javaee6“).build();
        }
}
```

Kod 5

Kaynaklar:
[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[2] http://jcp.org/en/jsr/detail?id=311
[3] http://www.jboss.org/resteasy
[4]http://jcp.org/aboutJava/communityprocess/maintenance/jsr311/311changelog.1.1.html
[5] https://jersey.dev.java.net/
[6] http://cxf.apache.org/
[7] https://jaxb.dev.java.net/
[8] http://hc.apache.org/httpclient-3.x/

Özcan Acar bilgisayar mühendisidir ve serbest danışman olarak çalışmaktadır. Yazar hakkında geniş bilgiyi http://www.ozcanacar.com adresinden edinebilirsiniz.