1.Soru

Endüstriyel dünyada işletmeler ve yazılım geliştiriciler açısından bir iş için harcanan zaman ve efor o işin cost-effective olmasının önünde yatan en büyük engellerdendir. İşte bu yüzdendir ki bir yazılım dilini en temel hali ile kullanmak yerine içerisinde önceden oluşturulmuş kütüphaneler sayesinde bütünlük ve stabilite sağlayan , yazılımcının işini kolaylaştıran araçları içerisinde barındıran frameworkler tercih edilir ve bu sayede üretkenlik artar ve aynı iş için harcanan süre azalır.Frameworkleri gruplayacak olursak :

1-ORM Frameworkleri, 2-Web Frameworkleri, 3-Testing Frameworkleri ve 4-Logging Frameworkleri diyebiliriz.

Orm Frameworkleri:

Nesne-tablo eşlemesi sağlarlar, yani Java nesnelerini veritabanı tablolarıyla ilişkilendirirler.

Veritabanı işlemlerini kolaylaştırır ve veritabanı bağımsızlığını sağlarlar.

Örnekler: Hibernate,JPA,MyBatis.

Web Frameworkleri :

Web uygulamaları geliştirmek için kullanılırlar.

İstek yönlendirme,şablon oluşturma,güvenlik sağlama gibi web geliştirme işlevlerini desteklerler.

Örnekler: Spring MVC ,Apache Struts, Play Framework.

Testing Frameworkleri:

Otomatik testler yazmak için kullanılırlar.

Birim testleri, entegrasyon testleri, kabul testleri gibi farklı test türlerini desteklerler.

Örnekler: JUnit, TestNG, Mockito.

Logging Frameworkleri:

Uygulama günlüklerini yönetmek için kullanılırlar.

Günlük mesajlarını farklı seviyelerde kaydetme, günlük rotasyonu gibi işlevleri desteklerler.

Örnekler: Log4j, SLF4J, Logback.

Java dünyasında yer alan bazı popüler frameworkler ve ele aldıkları çözümlere bakalım:

1-Spring Framework:

Dependency Injection,transaction management ve aspect-oriented programming(AspectJ JBoss) yada modülerizasyon da diyebileceğimiz birçok özelliği destekler.Bunlara ek olarak Web uygulamaları geliştirmek için Spring MVC'yi, Restful web servisleri oluşturmak için Spring Boot, veri erişimi için Spring Data ,güvenlik için Spring Security gibi çeşitli modülleri içerisinde barındırır.

Aşağıda bir Spring framework'ünde yer alabilecek bir kod örneğini gösterelim , ve turuncuyla belirttiğim anotasyonlar kodda boilerplate denilen tekrarlı yazımların önüne geçer.

```
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/library")
public class LibraryController {
    @DeleteMapping("/{bookId}")
    public String deleteBook(@PathVariable Long bookId) {
            return "Book with ID " + bookId + " has been deleted.";
    }
    // Sample method to handle POST requests
    @PostMapping("/add")
    public String addBook(@RequestBody Book book) {
            return "Book '" + book.getTitle() + "' has been added to the library.";
    }

}
```

Hibernate: Java için bir Object/Relational Mapping (ORM) framework'üdür. Veritabanı işlemlerini kolaylaştırır ve Java nesneleri ile veritabanı tabloları arasında bir köprü görevi görür. Aynı zamanda, veri erişimi ve manipülasyonunu daha basit ve platformdan bağımsız hale getirir.

```
public class Employee {
 @Id  @GeneratedValue(strategy=GenerationType.AUTO)
 private int id;
 private String firstName;
 private String lastName;
}
```

Apache Struts: Model-View-Controller (MVC) mimari desenini kullanarak web uygulamaları oluşturmak için kullanılan bir başka popüler Java framework'üdür. Struts, form verilerini işleme, veri doğrulama ve internalization gibi görevleri kolaylaştırır.

Apache Camel: Entegrasyon desenlerini uygulamak için bir framework sağlar. Farklı sistemler arasında mesajlaşmayı kolaylaştırır ve entegrasyon işlemlerini basitleştirir. Örneğin inboxtan outboxa bir etkileşim şu şekildedir.

```
from("file:data/inbox").to("file:data/outbox");
```

JSF (JavaServer Faces): Sunucu tarafında kullanıcı arayüzleri oluşturmak için kullanılan bir Java web framework'üdür. Component tabanlı mimariyi destekler ve web uygulamalarının UI kısmının kolay bir şekilde geliştirilmesine olanak tanır. Aşağıda bir buton oluşturumu ve auto navigation control ile sayfa değişimini içeren kodu görelim.

```
<h:form>
  <h3>Using JSF outcome</h3>
  <h:commandButton action = "page2" value = "Page2" />
</h:form>
@ManagedBean(name = "navigationController", eager = true)
@RequestScoped

public class NavigationController implements Serializable {
  public String moveToPage1() {
    return "page1";
  }
}
```

2.Soru

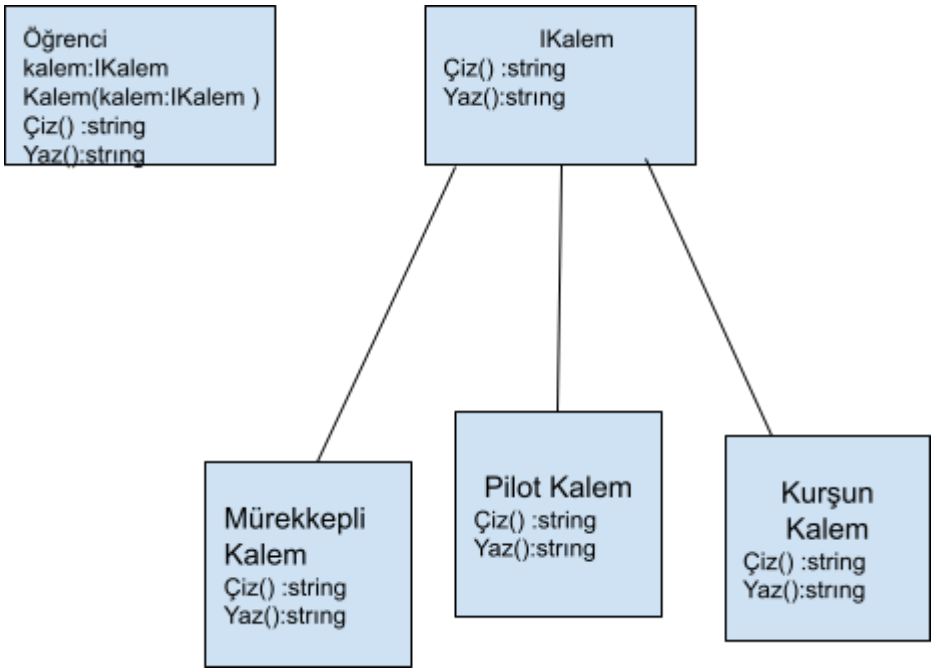
SOA nedir ?

Türkçeye hizmet odaklı mimari olarak çevrilen bu mimari bir yazılım geliştirme modelidir.Bu model sayesinde uygulama oluşturabilmek için tekrardan kullanılabilen ve farklı platformlar ile iletişim kurabilen servislerden faydalanılır.Her servisin kendine ait bir görevi ve servise ait bir input ve outputları içeren bir interface yer alır. Soa gevşek bağlılık (Loose coupling) adı verilen yöntemi kullanarak servislerin iletişimini yani veri iletimini yahut aktivitelerin koordinasyonunu sağlar.Loose coupling ise kısaca dependencyleri azaltmak için kullanılan bir yöntemdir.Örnekle açıklayacak olursak:

Bir öğrenci ve bir kalem arasında korelasyon kurabiliriz.

Öğrenci classımızda yazmak ve çizmek fonksiyonlarının olduğunu düşünelim.

Kalem classımızda ise yaz ve çiz fonksiyonlarının olduğunu farzedersek öğrenci classımızdaki yazmak ve çizmek fonksiyonları için kalem classına ihtiyaç duyarız ve bu bağlılık neticesinde Öğrenci sınıfını başka bir uygulamaya taşıyamayız ve dahası birbirlerine bağlı bu iki class proje için ilerde sıkıntı oluşturacaktır onun yerine



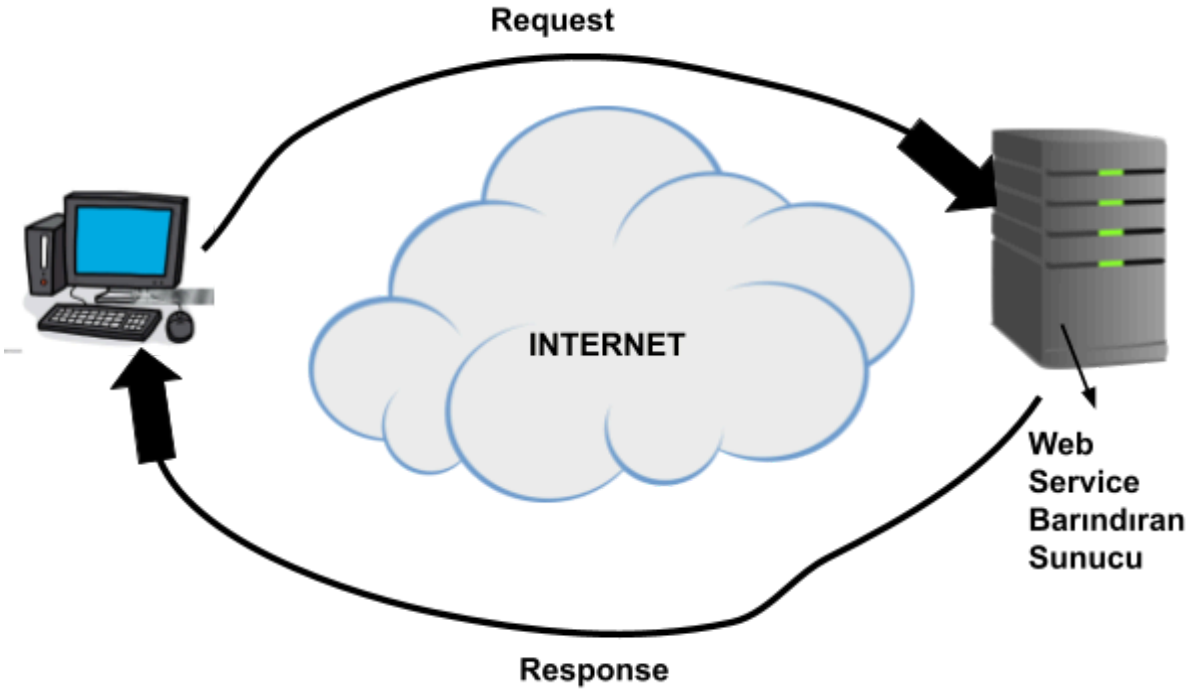
abstract class ya da interfaceler kullanarak Loose Couplingi sağlayabiliriz.Ayrıca SOA, standart ağ protokolleri (SOAP, JSON, ActiveMQ veya Apache Thrift gibi) kullanarak hizmetleri açığa çıkararak, geliştiricilerin entegrasyonu sıfırdan yapmasını engeller. Bunun yerine, enterprise service bus (ESB) olarak adlandırılan kalıpları kullanabilirler, bu kalıplar merkezi bir bileşen ile arka uç sistemleri arasındaki entegrasyonu gerçekleştirir ve ardından bunları hizmet arayüzleri olarak kullanılabilir hale getirir. Bu ayrıca geliştiricilerin mevcut işlevleri yeniden kullanmalarına izin verir, onları yeniden oluşturmak zorunda kalmazlar.

Webservice nedir?

Web service, client ve server uygulamaları arasında mesajların standart bir şekilde iletilmesi için kullanılan bir yöntemdir.Bir web servisi, belirli bir dizi görevi gerçekleştirmek için tasarlanmış bir yazılım bileşenidir. Bulut bilişiminde, bu hizmetler ağ üzerinden erişilebilir ve çağrılabilir. Kullanan istemcilere işlevsellik sağlarlar.Bir web servisi, farklı uygulamalar veya sistemler arasında veri alışverişine olanak tanıyan bir dizi açık protokol ve standarttır. Bu hizmetler, İnternet gibi bilgisayar ağları üzerinde veri alışverişi sağlarlar ve tek bir bilgisayar içinde işlem arası iletişim gibi çalışır. Çeşitli programlama dillerinde yazılmış ve farklı platformlarda çalışan yazılım programları tarafından kullanılabilirler. Web servisler XML, JSON, CSV vb. ortak bir biçim kullanarak farklı platformlar arasında ve uzak sistemler veri alışverişini sağlarlar.
Web servislerin en temel platformu XML ve HTTP'dir. Tipik tüm web servisleri tarafından kullanılan aşağıdaki bileşenler bulunmaktadır:
SOAP (Basit Nesne Erişim Protokolü)
SOAP, "Simple Object Access Protocol"ün kısaltmasıdır. Bağımsız bir iletişim protokolüdür. SOAP, SOAP Mesajları biçiminde XML verisi göndermek üzerine kurulmuştur. Her mesaja bir XML belgesi eklenir. Yalnızca XML belgesinin yapısı, içeriği değil, bir kalıbı izler. Web servisleri ve SOAP hakkında en iyi şey, her şeyin standart web protokolü olan HTTP üzerinden gönderilmesidir.
WSDL (Web Servisleri Tanım Dili)

Bir web servisi bulunamazsa, kullanılamaz. Web servisini çağıran istemcinin web servisinin konumunu bilmesi gerekir. İkinci olarak, istemci uygulamasının doğru web servisini çağırmak için web servisinin ne yaptığını anlaması gerekir. Bunu başarmak için WSDL veya Web Servisleri Tanım Dili kullanılır. WSDL dosyası, web servisinin ne yaptığını istemci uygulamasına açıklayan başka bir XML tabanlı dosyadır. İstemci uygulaması, web servisinin nerede bulunduğunu ve nasıl kullanılacağını anlayabilecek.

Web Servisi Nasıl Çalışır?



Diagram, bir web servisinin nasıl çalışacağının çok basitleştirilmiş bir versiyonunu göstermektedir. İstemci, gerçek web servisini barındıran bir sunucuya bir dizi web servisi çağrısı göndermek için istekler kullanır. Bu istekleri yapmak için uzak prosedür çağrıları kullanılır. İlgili web servisi barındıran yöntemlere yapılan çağrılar Uzak Prosedür Çağrıları (RPC) olarak bilinir. Örnek: Flipkart, Flipkart.com'da sunulan öğeler için fiyatlar gösteren bir web servisi sunar. Ön uç veya sunum katmanı .Net veya Java'da yazılabilir, ancak web servisi her iki programlama diliyle de iletişim kurulabilir.

İstemci ve sunucu arasında alışveriş edilen veri, web servisi tasarımının en önemli parçasıdır. XML (Genişletilebilir işaretleme dili), çeşitli programlama dilleri tarafından anlaşılan basit bir ara dil olarak hizmet eder. HTML'nin bir karşılığıdır. Bu nedenle, programlar birbirleriyle iletişim kurarken XML kullanırlar. Bu, farklı programlama dillerinde yazılmış uygulamaların birbirleriyle iletişim kurması için ortak bir platform oluşturur.

Uygulamalar arasında XML verisinin iletilmesi için, web servisleri SOAP'u (Basit Nesne Erişim Protokolü) kullanır. Veri, standart HTTP kullanılarak gönderilir. Bir SOAP mesajı, web servisinden uygulamaya gönderilen veridir. Bir SOAP mesajında bulunan tek şey bir XML belgesidir. Web servisini çağıran istemci uygulaması, içerik XML'de yazıldığı için herhangi bir programlama dilinde oluşturulabilir.

Restful Service nedir?
İlk olarak Roy Fielding tarafından 2000 yılında tanıtılan REST, o zamandan beri ölçeklenebilir, verimli ve bakımı kolay web tabanlı API'ların geliştirilmesinde temel bir yapı taşı haline gelmiştir.

REST, temelinde bir protokol veya standart değil, aksine basitlik, ölçeklenebilirlik ve durumsuzluğu vurgulayan bir mimari tarz olarak ön plana çıkar. Önceki sistemlerin aksine, REST, API geliştiricilerinin çeşitli yollarla uygulayabilecekleri esnek bir çerçeve sunar, bu da yeniliği ve adapte olabilirliği teşvik eder.

REST mimarisinin merkezinde, tasarımda tutarlılık ve uyum sağlayan kılavuz ilkeleri bulunur. REST'in altı kılavuz ilkesi, Uniform Interface, Client-Server ayrımı, Durumsuzluk, Önbelleğe Alma, Katmanlı Sistem ve isteğe bağlı Kod İndirme gibi, RESTful hizmetlerin inşa edildiği temel taşlarını oluşturur.

REST'teki önemli kavramlardan biri kaynakların, Uniform Resource Identifiers (URIs) aracılığıyla erişilebilen bilgi soyutlamaları olduğudur. Bu kaynaklar, GET, PUT, POST ve DELETE gibi iyi tanımlanmış işlemler kümesi kullanılarak harekete geçirilir ve istemcilerin etkileşimde bulunabileceği bir birim arayüzü sağlar.

Ek olarak, RESTful mimarisi kendiliğinden açıklayıcı mesajların ve durum etkileşimleri için hipermedya bağlantılarının kullanımını vurgular, böylece istemcilerin kaynakları otomatik olarak keşfedip etkileşime girmesine olanak tanır. Kaynakları temsilden ayırarak ve hipermedyadan yararlanarak, RESTful API'ler dağıtılmış sistemlerde esneklik ve adaptabilite sağlar.

RESTful web hizmetlerinin avantajları çok yönlüdür. Hız, çeşitli veri biçimleriyle uyumluluk, dil ve platform bağımsızlık ve çeşitli istemci-sunucu etkileşimlerini destekleme bunlardan sadece bazılarıdır. Ayrıca, RESTful mimarisi modern bilişim paradigmalarıyla, örneğin bulut tabanlı mimariler ve mobil bilişimle mükemmel bir uyum sağlar, bu da ölçeklenebilir ve dayanıklı web hizmetleri oluşturmak için ideal bir seçim yapar.

Sonuç olarak, RESTful mimarisinin evrimi ve yaygın kabulü, web hizmetlerinin peyzajını dönüştürdü, geliştiricilerin ölçeklenebilir, verimli ve birbirine bağlı API'lar oluşturmasını sağladı. Teknolojinin devam etmesiyle birlikte, REST modern yazılım geliştirme alanında temel bir prensip olarak kalmaya devam eder, yeniliği teşvik eder ve çeşitli platformlar ve sistemler arasında sorunsuz entegrasyonu kolaylaştırır.

Pazar günü (3.17.2024) yapılan ders sonrası yazdığım basit bir Rest Api örneği
Bu örnekte basitçe bir create işlevi gerçekleşmekte ve bir counter aracılığıyla sayısı tutulmakta.

```
@SpringBootApplication
@RestController
@RequestMapping("/products")
public class RestExampleApplication {

    private Map<Integer, Product> products = new HashMap<>();
    private int productIdCounter = 1;

    public static void main(String[] args) {
        SpringApplication.run(RestExampleApplication.class, args);
    }

    @PostMapping
    public Product createProduct(@RequestBody Product product) {
        product.setId(productIdCounter++);
        products.put(product.getId(), product);
        return product;
    }
}
```

HTTP methodlar nelerdir?

GET Method: Retrieving Data

GET yöntemi, sunucudan veri almak için kullanılır. İstemcilerin kaynakları değiştirmeden erişmesi gerektiğinde yaygın olarak kullanılır. Örneğin, bir e-ticaret API'sinin /products uç noktasına yapılan bir GET isteği, mevcut tüm ürünleri alırken, /products/123'e yapılan bir istek 123 numaralı ürünü getirir. Bir örnek düşünelim ve bu örnekte istemci sunucudan ürün listesini talep etsin:

GET /products HTTP/1.1

POST Method: Creating Resources

Aksine, POST yöntemi, sunucuda yeni kaynaklar oluşturmak için kullanılır. Bir e-ticaret veritabanına yeni bir ürün eklerken, POST isteği /products uç noktasına gönderilir. GET isteklerinin aksine, POST istekleri istek gövdesi içerir ve yeni kaynağın özelliklerini belirtir. İşte örnek bir POST isteği:

```
POST /products HTTP/1.1
Content-Type: application/json

{
  "name": "Spor Ayakkabıları",
  "color": "sari",
  "price": 900.00,
  "currency": "TL"
}
```

PUT Method: Updating Resources

PUT yöntemi, mevcut bir kaynağı güncel bir sürümle değiştirmek için kullanılır. PUT isteği gönderilerek, istemciler belirtilen kaynağı tamamen sağlanan veri ile değiştirebilirler. Eksik alanlar silinir ve yeni alanlar eklenir. Aşağıdaki PUT isteğini düşünelim:

PUT /products/123 HTTP/1.1
Content-Type: application/json

```
{
  "name": "Koşu Ayakkabıları",
  "color": "mavi",
  "price": 3000.00,
  "currency": "TL"
}
```

DELETE Method: Removing Resources

Son olarak, DELETE yöntemi sunucudan veri kaldırmak için kullanılır. Bir istemci bir DELETE isteği gönderdiğinde, belirtilen kaynağın silinmesini talep eder. Örneğin:

DELETE /products/123 HTTP/1.1