Soru 1-) MVC (Model-View-Controller), bir yazılım mimarisi ve tasarım desenidir. Temel olarak, bir uygulamanın farklı kısımlarını mantıksal olarak bölmeye ve organize etmeye yardımcı olur.

Model (Model): Veri ve iş mantığının bulunduğu kısımdır. Veritabanı işlemleri, veri işleme mantığı ve veri kaynakları bu bölümde yer alır.

View (Görünüm): Kullanıcı arayüzünün (UI) görsel temsili burada bulunur. Kullanıcıya sunulan verilerin gösterildiği, kullanıcıyla etkileşimde bulunulan kısım burasıdır.

Controller (Denetleyici): Kullanıcının etkileşimde bulunduğu ve işlemlerin gerçekleştirildiği kısımdır. Kullanıcıdan gelen istekleri alır, Model'i günceller ve sonuçları View'e ileterek kullanıcıya sunar.

MVC'nin kullanılmasının bazı avantajları şunlardır:

A) Modülerlik ve Kolay Bakım: MVC, kodun parçalara bölünmesini sağlar. Bu sayede her bir kısım bağımsız olarak geliştirilebilir ve bakımı kolaylaşır.
B) Test Edilebilirlik: Her bir bileşen ayrı ayrı test edilebilir. Model, View ve Controller'ın bağımsız olması test süreçlerini kolaylaştırır.

Java'da MVC, genellikle şu şekilde uygulanır:

A) Model: Java'da, Model veri işlemlerini ve iş mantığını temsil eder. Veritabanı işlemleri ve veri işleme kodları burada bulunabilir. Java'da bu genellikle sınıflar ve veri işleme mantığıyla temsil edilir.

B) View: Java'da View, kullanıcı arayüzünü temsil eder. HTML, JSP (JavaServer Pages), JavaFX gibi teknolojiler kullanılarak görsel arayüz oluşturulabilir.

C) Controller: Java'da Controller, kullanıcı isteklerini karşılayan ve iş mantığını yürüten kısımdır. Servletler veya Spring MVC gibi framework'ler kullanılarak HTTP isteklerini alabilir, Model'i güncelleyebilir ve uygun View'e yönlendirebilir.

Spring MVC gibi popüler Java framework'leri, MVC yapısını uygulamak için geniş imkanlar sunar. Bu framework'ler, Controller'ı yönetmek, istekleri işlemek ve View ile Model arasındaki iletişimi kolaylaştırmak için yardımcı olabilir.

OOP: Nesne Yönelimli Programlama , katmanlı bir yapıyı destekleyen bir yazılım geliştirme paradigmasıdır.

OOP Katmanları: 1) Entity Layer (Varlık) : Bu katman, veri modelini ve uygulama içindeki nesnelerin temsiliyetini içerir.

2) Business Logic Layer (İş Mantığı) : Bu katman, uygulamanın ana mantığını içerir.

3) Data Access Layer (Veri Erişim) : Bu katman, veritabanına veya dış kaynaklara erişimi sağlar.

4) User Interface Layer (Kulanıcı ara yüzü) : Bu katman, kullanıcıyla etkileşimde bulunulan katmandır

Soru 2-) Farklı programlama dilleri veya platformlar arasında iletişim kurmak için genellikle birkaç farklı yol bulunur:

a) API Kullanımı: RESTful API, SOAP, GraphQL gibi iletişim protokolleri kullanılabilir.
b) Veri Serileştirme: JSON veya XML gibi evrensel veri formatlarını kullanarak dönüştürebilirz.
c) Ortak Veri Tabanı Kulanımı
d) Ortak Veri Formatları: Her iki platform da bu formata uygun olarak veri alışverişi yapabilir.

Örn; RESTful API kullanarak Java'da yazılmış bir platform ile C# kullanılan bir platform arasında iletişim kurmak

Java'da RESTful API oluşturmak için genellikle Spring Framework'ü kullanırız. Öncelikle, Java'da bir RESTful servis oluşturulmalı.

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

@RestController
public class SampleController {

    @GetMapping("/hello")
    public String hello() {
        return "Merhaba, bu Java'dan bir RESTful servis!";
    }
}
```

Yukarıdaki kod, "/hello" endpoint'ine gelen GET isteğine cevap olarak "Merhaba, bu Java'dan bir RESTful servis!" metnini döndüren basit bir Spring Controller'ı temsil ediyor.

C# tarafında Java'ya istek yapacak istemci kodu aşağıda ki gibidir:

```csharp
using System;
using System.Net.Http;
using System.Threading.Tasks;

class Program
{
    static async Task Main(string[] args)
    {
        using var client = new HttpClient();

        try
        {
            HttpResponseMessage response = await client.GetAsync("http://java_service_url/hello");

            response.EnsureSuccessStatusCode(); // İstek başarılıysa devam et

            string responseBody = await response.Content.ReadAsStringAsync();

            Console.WriteLine("Java RESTful servisinden gelen yanıt: " + responseBody);
        }
        catch (HttpRequestException e)
        {
            Console.WriteLine("Hata oluştu: " + e.Message);
        }
    }
}
```

Bu C# kodu, belirtilen URL'ye bir GET isteği gönderir ve dönen cevabı konsola yazdırır.

Soru 3-) Web sayfasını yenilemeden güncel bilgiyi anlık olarak ekrana yansıtmak için JavaScript kullanabiliriz. Bu, web sayfasının dinamik olarak güncellenmesini sağlar. Bunları yapmak için birkaç yöntem var :

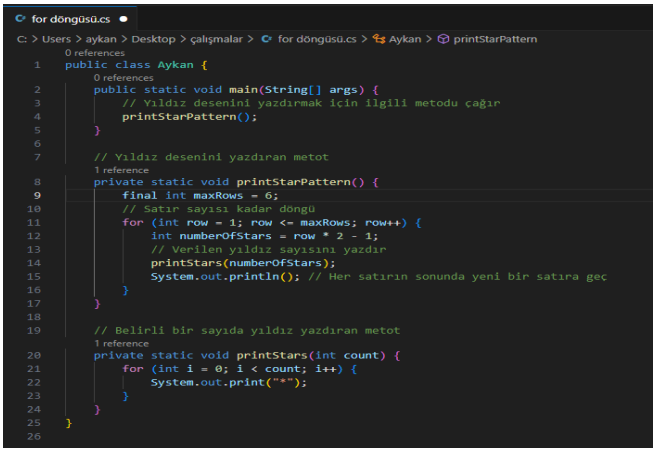
a) AJAX : Javascript'in AJAX özelliğiyle, belirli aralıklarla veya belir bir olay gerçekleştiğinde sunucudan veri alınabilir. Örneğin ; XMLHttpRequest veya modern tarayıcılar için fetch API'sini kullanarak sunucudan veri alabilir ve bu veriyi kullanarak sayfadaki belirli bir alanı güncelleyebilirsiniz.
b) Web Sockets Kullanımı : Sürekli bir bağlantı üzerinden sunucu ve istemci arasında veri alış verişi yapmamızı sağlar
c) Server-Sent Events: Sunucu , istemciye düzenli aralıklarla veya olaylar gerçekleştirdiğinde veri gönderir. İstemci bu olayları dinleyerek gelen verileri kullanarak sayfayı günceller.

Soru 4-)Kod:

```
public class Aykan {
    public static void main(String[] args) {
        // Yıldız desenini yazdırmak için ilgili metodu çağır
        printStarPattern();
    }

    // Yıldız desenini yazdıran metot
    private static void printStarPattern() {
        final int maxRows = 6;
        // Satır sayısı kadar döngü
        for (int row = 1; row <= maxRows; row++) {
            int numberOfStars = row * 2 - 1;
            // Verilen yıldız sayısını yazdır
            printStars(numberOfStars);
            System.out.println(); // Her satırın sonunda yeni bir satıra geç
        }
    }

    // Belirli bir sayıda yıldız yazdıran metot
    private static void printStars(int count) {
        for (int i = 0; i < count; i++) {
            System.out.print("*");
        }
    }
}
```



Soru 5-)

a-) SSH ile Sunucuya Erişim: Komut istemcisini açarız ve şu kodu yazarız :

ssh kullanici_adi@ip_adresi –p port_numarasi

Burada kullanici_adi yerine sunucuya erişim için gerekli olan kullanıcıyı yazarız ve ip_adresinin sunucunun IP adresini , port numarasına ise SSH bağlantı noktamızı gireriz. Örnek;

ssh username@246.897.513.0 -p 21

Ardından şifreyi girdikten sonra sunucuya erişmiş oluruz.

b-) Dosya Transferi:

Sunucuya Dosya Atma (SCP) , komut istemcisini açarız ve aşağıdaki komutu gireriz:

scp /local/dizin/dosya kullanici_adi@ip_adresi:/uzak/dizin

Bu komut yerel sunucunuzdaki dosyanızı sunucuda bulunan diğer konuma kopyalar.

Sunucudan Dosya Alma (SCP) : scp kullanici_adi@ip_adresi:/uzak/dizin/dosya /local/dizin

Scp ek olarak, SFTP ile de aktarım yapılabilir. SFTP interaktif bir dosya transfer protokolüdür. Komut istemcisine sftp kullanici_adi@ip_adresi komutunu girerek sunucuya bağlanabilirsin. Ardından put komutuyla dosyaları sunucuya, get komutuyla da sunucudan yerel makinene aktarabilirsin.

Soru 6-) Adım 1: Proje Oluşturma

Maven kullanarak boş bir Spring projesi oluşturdu.

Bağımlılıklar arasında Spring Web ve Spring Data JPA gibi gerekli bağımlılıkları ekledim.

Adım 2: Veritabanı Yapısının Oluşturulması

PostgreSQL veritabanı oluşturdum.

Şirketler ve çalışanlar gibi iki tablo oluşturdum.

Bu tablolar arasında ilişki kurdum (Örneğin, çalışanlar tablosunda şirketlere referans veren sütun var).

Adım 3: Spring ile Veritabanı Bağlantısı

application.properties dosyasında veritabanı bağlantı bilgilerini ayarladım.

Spring Data JPA kullanarak entity sınıflarını (Şirket ve Çalışan) ve ilişkilerini tanımladım.

Repository sınıfları oluşturun ve bu sınıflar aracılığıyla veritabanı işlemlerini gerçekleştirecek metodları tanımladım.

Adım 4: Servis Katmanının Oluşturulması

Şirketler ve çalışanlar için ekleme, silme, güncelleme, listeleme gibi servis metodlarını oluşturdum.

Bu servis metodları, repository sınıflarını kullanarak veritabanı işlemlerini gerçekleştirdim.

İşlemler sonucunda dönecek olan işlem detaylarını veya isteğin başarılı olduğu/olmadığına dair bilgileri response olarak döndürdüm.

Adım 5: Controller Katmanının Oluşturulması

RESTful API endpoint'lerini oluşturacak controller sınıflarını hazırladım.

Her bir HTTP isteği için gerekli servis metodlarını çağırın ve istemcilere uygun response'ları oluşturdum.

Adım 6: Postman ile Test Etme

Postman veya cURL gibi araçlar kullanarak oluşturduğunuz API endpoint'lerini test edin.

Ekleme, silme, güncelleme, listeleme gibi işlemleri gerçekleştirin ve cevapları kontrol ettim.

Adım 7: Dökümantasyon

Projenin nasıl ayağa kaldırılacağı, servislerin nasıl kullanılacağına dair bir dökümantasyon hazırladım.

MVC yapısını ve veritabanı işlemlerini anlatan detaylı açıklamalar ekledim.

Soru 7-) Solr sorgularında genellikle istekler HTTP GET üzerinden yapılır ve bir sorgu forına uygun olmalı. 2020 Ocak ayından sonraki verileri getirmek için aşağıdaki gibi sorgu yazılabilir:

http://example/select?q=updateAt:[2020-01-01 T00:00:00Z TO *]

Bu sorgu, Solr'a updatedAt alanı için 2020 Ocak ayından sonraki tüm verileri getirmesini söyler. Burada q parametresi, sorgunun kendisi için kullanılır. updatedAt:[2020-01-01T00:00:00Z TO *] kısmı, updatedAt alanının belirli bir tarih aralığını belirtir. 2020-01-01T00:00:00Z tarihinden itibaren * (yıldız) işaretiyle belirtilen sona kadar (yani şu anki zamana kadar) olan verileri getirecektir.

Github link

https://github.com/aykansaridogan/enoca_backend_challange

AYKAN SARIDOĞAN

04/12/2023