1.Hafta Ödevi: ilk iki soru.

1.

Java dünyasında birçok framework bulunmaktadır ve her biri farklı problemleri çözmek üzere tasarlanmıştır. İşte popüler Java framework'lerinden bazıları ve çözdükleri problemlere dair kısa açıklamalar ve örnekler:

- Spring Framework:

Problemi: Genel amaçlı bir framework olan Spring, Java uygulamalarının geliştirilmesi ve yönetilmesi için bir dizi araç ve kütüphane sunar. Dependency Injection (Bağımlılık Enjeksiyonu), Aspect-Oriented Programming (AOP), Transaction Management (İşlem Yönetimi) gibi çeşitli modüller içerir.
Örnek:

```
// Bir Spring bileşeni örneği
@Component
public class MyComponent {
    public void doSomething() {
        System.out.println("Spring Component çalışıyor.");
    }
}
// Kullanım
public class MyApp {
    @Autowired
    private MyComponent myComponent;

    public void run() {
        myComponent.doSomething();
    }
}
```

- Hibernate:

Problemi: Hibernate, Java nesnelerinin ilişkisel veritabanlarında depolanması ve yönetilmesi için bir ORM (Object-Relational Mapping) framework'üdür. Nesne yönelimli programlama ile ilişkisel veritabanı arasında köprü görevi görür.
Örnek:

```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "username")
    private String username;
    // Getters ve setters
```

```
}

// Kullanım
public class HibernateExample {
    public static void main(String[] args) {
        SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
        Session session = sessionFactory.openSession();
        Transaction transaction = session.beginTransaction();

        User user = new User();
        user.setUsername("example_user");
        session.save(user);

        transaction.commit();
        session.close();
    }
}
```

- Apache Struts:

Problemi: Web uygulamaları için bir MVC (Model-View-Controller) framework'ü olan Struts, web uygulamalarını organize etmek ve geliştirmek için kullanılır. JSP, Servlet ve JavaBean gibi teknolojilerle uyumlu çalışır.
Örnek

```
:// Action sınıfı
public class HelloWorldAction extends ActionSupport {
    private String message;

    public String execute() throws Exception {
        message = "Merhaba Dünya!";
        return SUCCESS;
    }

    public String getMessage() {
        return message;
    }
}

// struts.xml dosyası
<struts>
    <package name="default" namespace="/" extends="struts-default">
        <action name="hello" class="com.example.HelloWorldAction" method="execute">
            <result name="success">/hello.jsp</result>
        </action>
    </package>
</struts>
```

```
<!-- hello.jsp -->
<html>
<head><title>Hello World</title></head>
<body>
   <h1><s:property value="message"/></h1>
</body>
</html>
```

2.

SOA (Service-Oriented Architecture), web hizmetlerinin mimari yaklaşımını temsil eder. Bu yaklaşım, yazılım bileşenlerini bağımsız ve yeniden kullanılabilir hizmetlere bölmeyi ve bu hizmetleri birbiriyle etkileşime geçebilecekleri bir şekilde organize etmeyi amaçlar.

Web Servisleri, farklı uygulamalar arasında veri iletişimini sağlayan yazılım bileşenleridir. RESTful Servisler, web hizmetlerinin uygulandığı bir mimari tarzdır ve HTTP protokolünü kullanarak kaynaklara erişim sağlar.

HTTP yöntemleri, bir RESTful servisin sunabileceği temel işlemleri tanımlar. İşte bu kavramları örneklerle açıklayalım:

- GET: Bir kaynağın durumunu almak için kullanılır

Örneğin:

```
GET /customers/123
```

Bu istek, müşteri numarası 123 olan bir müşterinin bilgilerini getirmek için kullanılır.

- POST: Bir kaynağa yeni bir öğe eklemek için kullanılır.:

```
POST /customers
Body: { "name": "John", "email": "john@example.com" }
```

Bu istek, adı "John" olan ve e-posta adresi "john@example.com" olan yeni bir müşteri oluşturur.

- PUT: Bir kaynağın durumunu güncellemek için kullanılır.

Örneğin:

```
PUT /customers/123
Body: { "name": "John Doe", "email": "johndoe@example.com" }
```

Bu istek, müşteri numarası 123 olan müşterinin adını "John Doe" olarak ve e-posta adresini "johndoe@example.com" olarak günceller.

- DELETE: Bir kaynağı kaldırmak için kullanılır.

Örneğin:

```
DELETE /customers/123
```

Bu istek, müşteri numarası 123 olan müşteriyi siler.

Bu örnekler, SOA, web hizmetleri, RESTful hizmetler ve HTTP yöntemlerinin nasıl bir araya geldiğini göstermektedir. Bu kavramlar, uygulamalar arası iletişimde kullanılan temel prensipleri ve yöntemleri tanımlar.