1. SOAP vs Restful ?

REST'in uygulanması daha kolay, daha az bant genişliği ve kaynak gerektirirken **SOAP**'ın uygulanması çok kolay değildir, daha fazla bant genişliği ve kaynak gerektirir.

REST HTML,JSON,XML gibi veri formatlarına izin verirken, **SOAP** yalnızca XML veri formatına izin verir.

REST daha iyi performansa ve ölçeklenebilirliğe sahiptir. **REST** okumaları önbelleğe alınabilir, **SOAP** tabanlı okumalar önbelleğe alınamaz.

REST bir mimari yaklaşımken **SOAP** bir protokoldür. Bu yüzden **SOAP, REST**'i kullanamaz fakat **REST, SOAP** web servislerini ve SOAP ve HTTP gibi protokolleri kullanabilir.

REST, kaynakları ortaya çıkarmak için (genellikle) URI ve (GET, PUT, POST, DELETE) gibi yöntemler kullanırken **SOAP**, iş mantığını ortaya çıkarmak için hizmet arayüzlerini kullanır

Günümüzde **REST** yeni projelerle beraber çok daha fazla tercih edilirken, **SOAP** daha az tercih edilmekte ve genel olarak şuan eski ve büyük yapıda olan, değiştirilmesi güç projelerde kullanılmaya devam etmektedir.

2. Difference between acceptance test and functional test

Fonksiyonel testleri Yazılım Mühendisleri ya da QA mühendisleri tarafından yapılırken Kabul testleri müşteri tarafından yapılmaktadır.

Fonksiyonel testler kabul testinden önce gerçekleşir.

Fonksiyonel testlerde amacımız belirli bir işlevin doğruluğunu test etmek amaçlandığı için yan etkiler önemsizken kabul testlerinde bütüne bakıldığından yan etkiler de göz önünde bulundurulur

3. - What is Mocking ?

Bu kavramı açıklamadan önce unit test kavramını bir cümle ile açıklamak gerekir.
Unit Test, bir yazılımın en küçük test edilebilir bölümlerinin, tek tek ve bağımsız olarak doğru çalışması için incelendiği bir yazılım geliştirme sürecidir.

Mock kavramı, test edeceğimiz senaryolardaki bağımlılıkların kullanılmadan bu bağımlılıkları simüle edebilen sahte nesnelere verilen addır. Örnek vermek gerekirse,
Veritabanı işlemi yapan bir nesne yerine mock bir nesne kullanabilir ve bu işlemden beklediğimiz cevabı, mock nesnesi ile koda verebiliriz. Bu sayede bağımlıklardan kaynaklı oluşabilecek hatalardan sıyrılarak asıl senaryomuza odaklanabiliriz. Bunun yanı sıra gerçek işlemlerin yaratacağı yükten kurtularak yazacağımız çok sayıda testlerin daha hızlı çalışmasını sağlayabiliriz.

4 - What is a reasonable code coverage % for unit tests (and why) ?

%70-80 kod kapsamı, çoğu kapsam ölçütüne sahip çoğu projenin sistem testi için makul bir hedeftir. Yüksek test edilebilirlik için özel olarak düzenlenen veya yüksek başarısızlık maliyetleri olan projeler için daha yüksek bir hedef kullanın. Birim testi için minimum kod kapsamı, sistem testinden %10-20 daha yüksek olabilir.

Çünkü %100 kapsam yanıltıcı bir ölçümdür. Bunun sebebi tüm satırlara bir kez basarak % 100 elde edilebilir. Ancak yine de bu satırların mantıksal yolunu test etmeyi kaçırabiliriz.

Fakat daha kişisel bir yorumda bulunmam gerekirse bir birim testinin makul bir kod kapsamı yüzdesi yoktur. Bu, programın kapsamına, karmaşıklığına ve kodu yazan yazılımcının programlama ve test etme konusunda deneyimine bağlıdır.

5. HTTP/POST vs HTTP/PUT ?

POST kaynağa veri göndermek için kullanılır. **PUT** ise aynı kaynağa aynı adres ile erişilir ve eğer içerik var ise gelen veriler ile değiştirilir, eğer içerik yok ise yeni içerik yaratılır.

Kısaca **PUT** veri güncellemek için kullanılır.

PUT /questions/{question-id} şeklinde kullanılırken **POST** /questions şeklinde kullanılır

PUT pratikte genellikle güncelleme işlemleri için kullanılırken, **POST** ise create işlemleri için kullanılır.

PUT ile Aynı isteği birden çok kez gönderirseniz sonuç aynı kalır. **POST** ile aynı isteği birden fazla kez gönderirseniz, farklı sonuçlar alırsınız.

6. What are the Safe and Unsafe methods of HTTP ?

Öncelikle kullandığımız HTTP metodu sunucunun durumunda bir değişiklik gerçekleştirmiyorsa yani bilgi almak gibi salt okunur bir işlem gerçekleştiriyorsa güvenli HTTP metodu diyebiliriz. Bunlar GET, HEAD ve OPTIONS metotlarıdır.

Eğer sunucu ya da uygulamamın durumunu değiştiriyorsak, örnek vermek gerekirse bir web uygulamasında bir kullanıcının profil bilgilerini güncellemek istiyorsak güvenli olmayan HTTP metotları kullanılır. Bunlar POST, PUT ve DELETE metotlarıdır.

7. - How does HTTP Basic Authentication work ?

İstemci ile sunucu arasında gerçekleşen bir doğrulamadır.

Öncelikle istemci temel kimlik doğrulama ile korunan bilgisayara erişimi isteği gönderir.

Sunucu, istemciye kullanıcı adı ve parola isteyen bir kutucuk döndürür ve geri yanıt bekler.

İstemci tarafında girilen kullanıcı adı ve parola sunucuya gönderilir.

Sunucuya gelen bu kullanıcı adı parola bilgisi sorgulanır eğer doğruysa istenen kaynak istemciye döndürülür.

8 - Define RestTemplate in Spring ?

Biz bir servis yazdık ama başka bir servise soru soracağız yani istek atacağız. O zaman RestTemplate kullanarak o metodu nasıl çağıracağımızı, bunlardaki geri dönüş verilerini nasıl alacağımızı ve işleyeceğimizi tanımlayarak request atmamıza yarıyor.

RestTemplate uygulanması oldukça basit ve sade bir yapıya sahiptir. Servis adresini, hangi HTTP yöntemi ile operasyonun gerçekleştirileceğini ve dönüş türünün ne olacağını veriyoruz.

```
ResponseEntity<Foo> response = restTemplate.exchange(uri,
HttpMethod.GET, null, new ParameterizedTypeReference<Foo>(){});

List<Foo> result = response.getBody();
```

9. What is idempotant and which HTTP methods are idempotant ?

Bir HTTP metodu, sunucuyu aynı durumda bırakırken aynı etkiyle arka arkaya bir veya birkaç kez aynı istekte bulunulabiliyorsa idempotant'tır.

Doğru bir şekilde uygulandığında GET, HEAD, PUT, ve DELETE yöntemleri idempotant'tır.

Buna örnek olarak DELETE metodunu verebiliriz. Örneğin, DELETE ile sunucundan bir kaynağı silmek için bir istek attık ve sunucuda bu işlenerek kaynak silindi. Daha sonrasında istemciden aynı DELETE isteği tekrarlanırsa kaynak silinmiş olacağından 404 döndürür.

10. What is DNS Spoofing ? How to prevent ?

DNS zehirlenmesi, Domain Ad Sistemi sunucusunun ön bellek veritabanına veri eklenmesi ya da oradaki verilerin değiştirilmesiyle oluşur. Bu durum oluşurken; yanlış IP adreslerinin dönmesi veya trafiğin başka bir bilgisayara yönlendirilmesi gerçekleşir. Genellikle, trafiğin; saldırıyı yapanın bilgisayarına yönlendirilmesine neden olur.

İnternete halka açık Wi-Fi'den erişen herhangi bir kullanıcı, DNS sahtekarlığına karşı savunmasızdır. DNS sahtekarlığından korunmak için internet sağlayıcıları DNSSEC (DNS güvenliği) kullanabilir. Bir etki alanı sahibi DNS girişlerini ayarladığında, DNSSEC, çözümleyicilerin DNS aramalarını gerçek olarak kabul etmeden önce ihtiyaç duyduğu girişlere bir kriptografik imza ekler.

Standart DNS şifrelenmez ve değişikliklerin ve çözümlenen aramaların meşru sunuculardan ve kullanıcılardan gelmesini sağlamak için programlanmamıştır. DNSSEC, güncellemeleri doğrulayan ve DNS sahteciliğinin engellenmesini sağlayan işleme bir imza bileşeni ekler. DNS sızdırma, herhangi bir genel Wi-Fi üzerinden kullanıcı veri gizliliğini ihlal etmekle tehdit ettiğinden, DNSSEC son zamanlarda daha popüler hale geldi.

HTTPS göstergesi her zaman tarayıcı adres çubuğunda olmalıdır. Bu, sitenin geçerli olduğunu bilmenizi sağlar. HTTPS göstergesinin görünümü değişkense, bir saldırının başladığını gösterebilir.

11. What is content negotiation ?

Content Negotiation konusu dilimize çevirmek istersek istemci ile sunucu arasındaki içerik anlaşması denilebilir. HTTP protokolünün inşa edildiği şekile bakıldığında kullanıcının ne tür veri göndereceği ve sunucunun ne tür veri kabul edeceğinin özelleştirilmesidir.

Gerekli xml configürasyonlarını yaptıktan sonra istediğimiz formattaki dosya tiplerini @RequestMapping anatasyonununda produces anahtarını kullanarak aşağıdaki örnekteki gibi tanımlama yapabiliriz.

@RequestMapping(value = "/employeeData", method = RequestMethod.GET, produces={"application/xml", "application/json"})

Bu örnekte xml ve json biçimlerini desteklemek istediğimiz için bunları yukarıdaki gibi belirtiyoruz.

12. What is statelessness in RESTful Web Services ?

REST mimarisine göre, bir RESTful Web Hizmeti, sunucuda bir istemci durumu tutmamalıdır. Bu kısıtlamaya Stateless denir.

Server tarafında client ile ilgili bir context veya Session tutulmaz. Client tarafından yapılan her request Server'ın response verebilmesi için gerekli bilgiyi taşır, yani her türlü state client tarafında tutulur, ihtiyaç duyulursa request içerisinde server'a bildirilir.
Bu **Scalability** açısından da önemlidir, çünkü server'ın requestler arasında herhangi bir state'i saklamasını gereksiz kılar ve kaynak yönetimini kolaylaştırır. **Visibility** açısından önemlidir, çünkü request'in amacını anlamak için tek bir request'in içerdiği bilgiler yeterlidir.

Avantajları olarak, Web hizmetleri, her bir yöntem isteğini bağımsız olarak ele alabilir.

Durum bilgisi olmaması, tüm sunucu tarafı durum senkronizasyon mantığını kaldırarak REST API'lerini daha az karmaşık hale getirir.

Dezavantaj olarak, Client her request'de gerekli bilgileri eklemek zorundadır bu da network trafiğini arttırır.

13. What is CSRF attack? How to prevent ?

CSRF saldırısı, daha önce kimliği doğrulanmış başka bir web sitesi aracılığıyla bir web uygulamasına istek gönderen kötü amaçlı bir bağlantı içerir. Elde edilen kimlik bilgileriyle mağdur kimliğine bürünülür ve kötü amaçlı faliyetlerde kimlik doğrulama bilgisi atlanılmış olur. Örneğin, bankacılık sistemine giriş sayfası tarayıcıda açık bulunduğu bir durumda , mail adresine gelen tehlikeli bir bağlantı tıklanarak saldırgana kullanıcı bilgileri verilmiş olur. Saldırgan bu bilgilerle bankacılık sistemine girip para transferi gerçekleştirebilir.

Bu tür saldırılar genellikle bankacılık, sosyal medya ve ağ cihazları için kullanılan web arayüzlerine karşı gerçekleştirilir. Şu şekilde önlemler alınabilir;

Kullanıcıya her oturum için random ve benzersiz "token" bilgisi verilir.

Kullanıcıdan alınan önemli veriler GET yerine POST metodu ile alınmalıdır.

Bir web formunda captcha bilgisi doğru girilmediği sürece işlem gerçekleştirilemeyeceği için "CSRF" saldırısına karşı alınacak bir önlem niteliğindedir.

14. What are the core components of the HTTP request and HTTP response ?

HTTP Request'in 5 temel bileşeni vardır.

Verb: GET, POST, DELETE, PUT vb. gibi HTTP metotlarını belirlir.

URI: bir kaynağı açık bir biçimde tanımlayan standart forma sahip bir karakter dizgisidir.

HTTP Version: HTTP sürümünü belirtir. Örneğin HTTP v1.1 gibi

RequestHeader : Anahtar/değer çiftleri olarak HTTP İstek mesajı için meta verileri içerir. Örneğin, istemci (veya tarayıcı) türü, istemci tarafından desteklenen biçim, ileti gövdesi biçimi, önbellek ayarları gibi.

Request Body: İleti içeriği veya Kaynak gösterimi

HTTP Response 3 temel bileşene sahiptir.

Status Line: Aşağıda görüldüğü gibi HTTP Version, HTTP Response Code ve Status Code gösterilir.

```
HTTP/1.1 200 OK
```

Response Header: yanıt olarak döndürülen içerikle ilgili bilgileri, onu gönderen Sunucu hakkındaki verilerle birlikte içerir. Bu bilgi, Müşterinin/Tarayıcının yanıt verilerinin ne şekilde kullanılacağına karar vermesine yardımcı olur. Başka bir deyişle, başlıklar, kendisi hakkında daha fazla bilgi sağlamak için bir yanıtla birlikte gönderilen meta veriler olarak söylenebilir.

Body: Gövde verileri taşır ve json, html, image , vb. gibi başlıklarda buna göre belirtilen birçok formattan birinde olabilir .