1 – SOAP vs Restful ?

1. SOAP daha az tercih edilir
   REST daha fazla tercih edilir.

2. SOAP bir protokoldür
   REST bir mimarı yapıdır.

3. SOAP business logicleri expose etmek için service interfaceleri kullanır
   REST ise URI ları kullanır

4. SOAP' da dosyalar daha fazla yer kaplayacağından daha fazla bant genişliğine ihtiyaç duyulur.
   REST'de daha az bant genişliğine ihtiyaç duyulur

5. SOAP sadece XML data formatıyla çalışır
   REST birçok data formayıtla çalışır HTML,XML,JSON vb.

6. JAX-WS API ' ı SOAP web service leri içindir
   JAX-RS API' ı RESTFULL web service leri içindir.

7. SOAP kendi security 'sini tanımlar.
   REST temel aktarım alarak securityleri inherit eder.

2-Difference between acceptance test and functional test ?

Acceptance Tests : son kullanıcı senaryolarının testlerini ifade etmektedir. Özünde yine entegrasyon testleri olarak kabul edilebilse de, bir kullanıcı hikayesinin test edilmesini ifade etmektedir. Örneğin kod yardımıyla yine tarayıcı açılarak bir kullanıcı hikayesi, örneğin yeni bir kullanıcı oluşturularak, kullanıcıya yetkiler atanması işlemi test edilir. Test senaryoları İngilizce'ya yakın bir dilde kodlanır ve kabul testi yazmak için geliştirilmiş kendine özgü araçları vardır.

Kabul testlerindeki amaç; yazılımın sunulacağı kullanıcı grubunun yazılımı çalışıyor olarak kabul edebilmesi için gerekli olan işlemlerin gerçekleştirilebilip gerçekleştirilemeyeceğinin kontrolüdür. Sistemin neresinde hatalar olduğunun tespitinden ziyade, "Sistem kendisinden beklenilen işlemleri yapamıyor." demek için kullanılmaktadır.

Fonksiyonel testler: belirli bir veri grubunun sonuçlarını, bir sonuç kümesiyle karşılaştırarak kontrol eder. Fonksiyonel testler ara sonuçlar ya da yan etkiler ile ilgilenmezler. Sadece sonuç odaklı çalışırlar. X işlemi yapıldıktan sonra, sonucun ne olduğu ile ilgilenmezler. [5]

Yukarıdaki paragraf biraz karışık gelebilir. Bu nedenle biraz örneklemek yerinde olacaktır. Örneğin bir oturum açma işleminin testini yazdığımızı düşünelim. Oturum

aç butonuna tıkladığımızda, sadece bizi anasayfaya yönlendirmesini test ediyorsak ve girdiğimiz kullanıcı bilgilerinin doğru olup olmadığının önemli olmasını istemiyorsak bu bir fonksiyonel testtir. Eğer bu bilgilerin de doğru olup olmadığının kontrolünü istiyorsak, o halde bu bir entegrasyon testidir.

Birbirine çok yakın olarak gözüken bu iki test türünün temel farkı; birinde (fonskyionel test) yan etkiler (veritabanında kayıt oluşup oluşmaması gibi) önemsizdir. Diğerinde (acceptance test) yazılımdan beklenen işlerin yapılıp yapılmadığı. Çünkü sistemin bir bütün olarak çalışıp çalışmadığı kontrol edilmektedir. Genellikle bir resme tıkladığınız zaman, resmin bir modal üzerinde açılması, profil linkine tıkladığınızda sayfanın değişmesi gibi testler fonsksiyonel testler olarak isimlendirilirler.

Eğer amacınız yazılımdan beklenen işlemlerin yapılıp yapılamadığını kontrol etmek ise; acceptance test. (Yeni bir kullanıcı kaydedilmesinin arayüz üzerinden testi)

Eğer amacınız ortamdan ve yan etkilerden bağımsız olararak sadece bir fonksiyonun test edilmesi ise; functional test

3-What is Mocking?

Mocking(mocklama), popüler yazılım metodolojisi olan TDD ve özelde birim testlerinin (unit test), test ettikleri sistemi izole etmede kullandığı yöntemlerden biridir. Bu yöntemler, geniş anlamıyla test dublörleri (test double) olarak tanımlanabilir. Test dublörleri, test edilen sistemin bağımlı olduğu diğer birimlerin yerini tutar. Bu izolasyona birim testlerinde ihtiyaç duyulmasının temelde iki sebebi vardır:

Birim testleri, genelde test ettikleri sistemin kendisi ile ilgili varsayımları doğrulamak için yazılır.

Test dublörleri, davranış ve kullanım şekillerine göre çeşitlenir. Bunlardan en çok kullanılanları dummy, fake, stub, spy ve mock'tur denebilir. Bu çeşitliliğe sebep olan genel faktörler, bu dublörlerin beklenen işi yapıp yapmadığı ve yaparken nasıl bir davranış gösterdiği ile ilgilidir.

NOT: 1. madde ile ilgili olarak; entegrasyonu testleri, izole halde çalışan bağımsız sistemlerin bir araya getirildiğinde ortaya çıkan durumlarını inceler. Birim testlerinin istenilen çalışma sürelerinin kısa olması da bir diğer faktör olarak sayılabilir.

Mock'ların genel kullanım şekli, yerine geçtiği bağımlılık üzerinde, test edilen sistemin yapması beklenen işlemlerin yapılıp yapılmadığını doğrulamak olarak tanımlanabilir. Mock nesnelerinin casus nesnelerinden (spy) farkı, her ikisi de üzerlerinde yapılan işlemleri takip ederken, mocklar bu işlemi testlerin doğrulama (assert) kısmına da entegre ederler.

Mocklama işlemi genelde kütüphaneler yardımıyla, test metodlarının içinde, veya tekrar eden test ayarlarının yapıldığı bir özelleştirme metodunu, satır arası kodlar ile yapılır. Yani çoğu zaman

mocklanan tipden devralan bir tip yazılmaz. Mock kütüphaneleri genelde bu işi, dilin reflection kütüphanesinden faydalanarak, çalışma zamanında, ayarlanan kurulumu sağlayacak vekil tipler üreterek sağlarlar. Bunun getirdiği bir avantaj, bütün ön koşulları yerine getiren bir dublör birimi yazmadan, her test metodu için yalnızca beklenen davranışı sağlayan satır içi ayarlamalar yapılmasına olanak sağlamasıdır.

Mock kütüphanelerinin diğer bir avantajı ise, genellikle kendi içlerinde doğrulama (assertion) mekanizmaları bulundurmaları ve mocklanan nesne üzerindeki beklentilerin karşılanıp karşılanmamasına göre, çağırıldıkları testin başarı durumunu etkilemeleridir. Mock kütüphanelerinin yaptığı bu işi, elle yazılan mock tiplerinde geliştiricinin kendisi yapması gerekebilir.

Mocklama işlemi yapılırken genelde bağımlı olan birimle ilgili yapılması beklenen çağırmalarla ilgili ayarların yapılmasının yanı sıra, bazen test edilen sistemin doğru çalışması için, bu çağırmaların geriye değerler döndürmesi gerekmektedir. Bu durumları çözmek için, geliştirilen veya ayarlanan mock tipinin, test edilen tipi rahatsız etmeyecek bir değer döndürmesi gerekebilir. Mock kütüphaneleri böyle durumlarda yapılan ayarlamalarda geriye değer döndürmeye de olanak sağlar.

Dezavantajlar

Mock nesnelerini kullanmanın çeşitli avantajları olduğu gibi, dezavantajları da olabiliyor.

Bir dezavantaj test ortamında mock kullanılacak bir sistemde neredeyse her şeyin birer arayüz (interface) üzerine inşa edilmesi gerekebiliyor. Bu da kimi zaman over-engineering olarak nitelendirilen probleme yol açabiliyor. Arayüz gerekmeyen yerlerde sınıfların kullanılması da sadece mocklamanın yapılabilmesi için normal şartlarda son ve kesin halinde olan metodların (örneğin C# için), virtual olarak işaretlenmesini gerektirebiliyor. Sınıflardan devralarak yazılan mocklar için bir başka sorun ise constructor'ların her durumda çağrılması, örneğin veritabanına constructor'ında bağlantı sağlayan bir sınıfın üzerine inşa edilen bir mock nesnesi de kullanmayacağı halde yine bu bağlantıyı kurmak zorunda kalabiliyor.

Mock ile ilgili bir başka dezavantaj, test ortamını kimi zaman karmaşıklaştırması olabilir. Bağımlı olunan sınıf yirmi satırlık bir kod barındırabilirken, test ortamının gerekliliklerini sağlamak için yüzlerce satır kod ile bahsedilen sınıfı taklit eden bir mock objesi yazmak gerekebiliyor. Bu karmaşıklığı çözmek için, "kendini tekrar etme" prensibini kullanmak gerekebiliyor. Bunun için test metodlarında parametreler ve bu parametreleri çözümleyen "fixture" kütüphanelerini kullanılabilir.

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

%100 . Her şeyin test edildiğinden emin olmak istediğiniz için bunu seçebilirsiniz. Bu size test kalitesi hakkında herhangi bir fikir vermez, ancak bazı kalite testlerinin her ifadeye (veya şubeye vb.) dokunduğunu söyler. Yine, bu güven derecesine geri döner: Kapsamınız %100'ün altındaysa , kodunuzun bazı alt kümelerinin test edilmediği ortadadır.

Bazıları bunun önemsiz olduğunu vurgular ve sadece kodunuzun önemli olan kısımlarının test eedilmesi gerekliliğini savunur. Ayrıca, kodunuzun yalnızca önemli olan kısımlarını da korumanız gerektiğini savunur. Test edilmemiş kodlar da kaldırılarak kod kapsamı iyileştirilebilir.

%80-20 kuralının, kodunuzun çoğunun test edildiğini göstermektir.

Pratikte %80'in altındaki sayıları görmek pek mümkün değildir.Bu standartların rolü, doğruluk konusundaki güveni artırmaktır ve %80'in altındaki rakamlar özellikle güven verici değildir.

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.

Sunucu durumunu değiştirmezlerse HTTP yöntemleri güvenli kabul edilir. Bu nedenle güvenli yöntemler yalnızca salt okunur işlemler için kullanılabilir. HTTP RFC, güvenli olmak için aşağıdaki yöntemleri tanımlar: GET, HEAD, OPTIONS ve TRACE.

Pratikte, herhangi bir sunucu durumunu değiştirmeyecek şekilde güvenli yöntemleri uygulamak çoğu zaman mümkün değildir.

Örneğin, bir GET isteği, günlük veya denetim mesajları oluşturabilir, istatistik değerlerini güncelleyebilir veya sunucuda bir önbellek yenilemesini tetikleyebilir.

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

Sunucu durumunu değiştirmezlerse HTTP yöntemleri güvenli kabul edilir. Bu nedenle güvenli yöntemler yalnızca salt okunur işlemler için kullanılabilir. HTTP RFC, güvenli olmak için aşağıdaki yöntemleri tanımlar: GET, HEAD, OPTIONS ve TRACE.

Güvenli olmayan metotlar ise sunucudaki bazı kaynakları değiştirebilirler : PUT,DELETE,POST,PATCH

7- How does HTTP Basic Authentication work ?

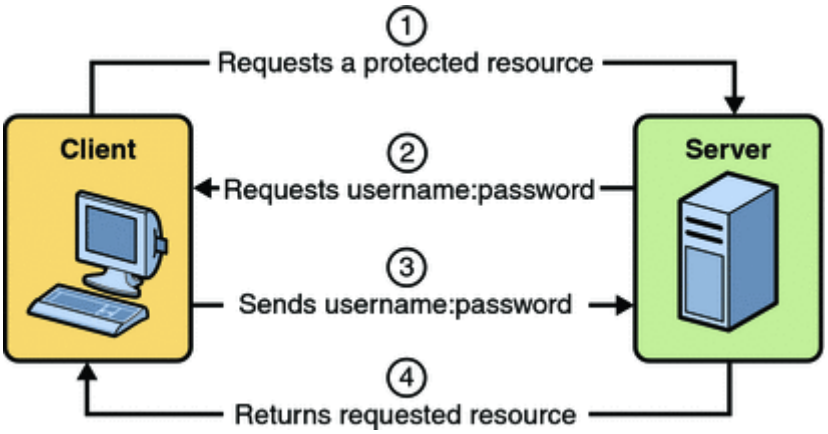
HTTP Temel Kimlik Doğrulaması, sunucunun web istemcisinden bir kullanıcı adı ve parola istemesini ve bunları yetkili kullanıcılardan oluşan bir veritabanıyla karşılaştırarak kullanıcı adı ve parolanın geçerli olduğunu doğrulamasını gerektirir. Temel kimlik doğrulama bildirildiğinde, aşağıdaki eylemler gerçekleşir:

1. Bir istemci, korunan bir kaynağa erişim ister.

2. Web sunucusu, kullanıcı adı ve parola isteyen bir iletişim kutusu döndürür.

3. İstemci, kullanıcı adını ve parolayı sunucuya gönderir.

4. Sunucu, belirtilen alanda kullanıcının kimliğini doğrular ve başarılı olursa istenen kaynağı döndürür.



8- Define RestTemplate in Spring ?

RestTemplate, istemci tarafı HTTP erişimi için merkezi Spring sınıfıdır. Kavramsal olarak, JdbcTemplate, JmsTemplate ve Spring Framework ve diğer portföy projelerinde bulunan diğer çeşitli şablonlara çok benzer. Bu, örneğin, RestTemplate oluşturulduktan sonra iş parçacığı için güvenli olduğu ve işlemlerini özelleştirmek için geri aramaları kullanabileceğiniz anlamına gelir.

Bu yöntemlerin adları, hangi HTTP yöntemini çağırdıklarını açıkça belirtirken, adın ikinci kısmı neyin döndürüldüğünü gösterir. Örneğin, getForObject() bir GET gerçekleştirir, HTTP yanıtını seçtiğiniz bir nesne türüne dönüştürür ve o nesneyi döndürür. postForLocation, verilen nesneyi bir HTTP isteğine dönüştürerek bir POST yapacak ve yeni oluşturulan nesnenin bulunabileceği HTTP Konum başlığı yanıtını döndürecektir. Gördüğünüz gibi, bu yöntemler REST en iyi uygulamalarını zorlamaya çalışır.

HTTP	RestTemplate
DELETE	delete(String, String...)
GET	getForObject(String, Class, String...)
HEAD	headForHeaders(String, String...)
OPTIONS	optionsForAllow(String, String...)
POST	postForLocation(String, Object, String...)
PUT	put(String, Object, String...)

9-What is the difference between @Controller and @RestController ?

Controller, Spring içerisinde uzun zamandır bulunan bir annotation. Diğer yandan ise @RestController, Spring 4.0 ile framework bünyesinde katıldı. Temel amacı adından da anladığımız gibi REST endpoint'leri üretmeyi sağlamak. Aslında bunun bize sağladığı şey @Controller ve @ResponseBody notasyonlarını tek seferde vermesi. Bu sayede REST controller olmasını isteyeceğimiz her metoda @ResponseBody yazmaktan da kurtulmuş oluyoruz.

10-What is DNS Spoofing ? How to prevent ?

Saldırıdaki "spoofing" terimi, tehdit aktörünün, kullanıcının bildiği resmi web sitesine benzeyen kötü amaçlı bir site kullandığı anlamına gelir. DNS, İnternet iletişiminin kritik bir parçası olduğundan, zehirlenme girişleri, bir saldırgana hassas verileri toplamak için mükemmel bir kimlik avı senaryosu sunar. Tehdit aktörü parolaları, banka bilgilerini, kredi kartı numaralarını, iletişim bilgilerini ve coğrafi verileri toplayabilir.

Kullanıcı web sitesinin resmi olduğunu düşündüğü için saldırgan bir kimlik avı kampanyasını başarıyla yürütebilir. Sahte site, kullanıcının tanıyabileceği öğelere sahiptir ve ideal olarak, sitenin sahte olduğunu gösteren kırmızı bayraklar içermez. Sahte bir web sitesinde kasıtsız kırmızı bayraklar bulunabilir, ancak kullanıcılar bunları nadiren fark eder, bu da sahteciliği özel verileri çalmak için etkili bir yol haline getirir.

DNS spoofingden 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 şifreleme imzası ekler.

11-What is content negotiation

HTTP'nin bir parçası olarak tanımlanan ve bir belgenin farklı sürümlerinin (veya daha genel olarak bir kaynağın temsillerinin) aynı URI'de sunulmasını mümkün kılan mekanizmaları ifade eder, böylece kullanıcı aracıları hangi sürümün yeteneklerine en uygun olduğunu belirleyebilir. Bu mekanizmanın klasik bir kullanımı, bir görüntüyü GIF veya PNG formatında sunmaktır, böylece PNG görüntülerini görüntüleyemeyen bir tarayıcıya (örn. MS Internet Explorer 4) GIF sürümü sunulur.

Bir kaynak birkaç farklı gösterimde mevcut olabilir; örneğin, farklı dillerde veya farklı ortam türlerinde mevcut olabilir. En uygun seçeneği seçmenin bir yolu, kullanıcıya bir dizin sayfası vermek ve en uygun seçeneği seçmesine izin vermektir; ancak bazı seçim kriterlerine dayalı olarak seçimi otomatikleştirmek çoğu zaman mümkündür.

12-What is statelessness in RESTful Web Services ?

REST mimarisine göre, bir RESTful Web Hizmeti, sunucuda bir istemci durumu tutmamalıdır. Bu kısıtlamaya ' Statelessness' denir. Bağlamını sunucuya iletmek istemcinin sorumluluğundadır ve ardından sunucu, müşterinin daha sonraki isteğini işlemek için bu bağlamı saklayabilir. Örneğin, sunucu tarafından sürdürülen oturum, istemci tarafından geçirilen oturum tanımlayıcısı tarafından tanımlanır.

13-What is CSRF attack? How to prevent ?

Türkçe açılımı “Siteler Arası İstek Sahtekârlığı” şeklinde olan CSRF zafiyeti; web uygulamasını kullanmakta olan kullanıcıların istekleri dışında işlemler yürütülmesidir. Uygulamaya giden isteklerin hangi kaynaktan ve nasıl gönderildiği kontrol edilmeyen sistemlerde ortaya çıkan bu zafiyet, aslında uygulamayı kodlayan yazılımcıların gözünden kaçan bir ayrıntıdır diyebiliriz. Genelde CSRF veya XSRF şeklinde kısaltılan bu güvenlik açığı “**Session Riding**” olarak da bilinmektedir.

CSRF zafiyetine karşı hem sistem tarafından hem de kullanıcı tarafından alınabilecek önlemler vardır.

Sistem tarafında alınması gereken önlemler:

1. Kullanıcının sisteme gönderdiği önemli talepler POST metodu ile alınmalıdır.
2. Siteler Arası Talep Sahteciliğini (CSRF) önlemek için en popüler yöntem, kullanıcıya rastgele üretilmiş eşsiz bir “token” bilgisi vermektir. CSRF Token veya Synchronizer Token olarak adlandırılan bu yöntem şu şekilde çalışır:

1. Web sunucusu bir token oluşturur. (Bu token işlem yapıldıkça yeniden üretilir.)
2. Token, formda gizli bir bilgi olarak depolanır.
3. Kullanıcı POST işlemini gerçekleştirir.
4. Token bilgisi POST verilerine dahil edilir.
5. Web uygulaması, sistem tarafından oluşturulan tokeni, talepte gönderilen token ile karşılaştırır.
6. Eğer token verileri eşleşirse, isteğin gerçek kullanıcı tarafından gönderildiği anlaşılır ve istek onaylanır.
7. Eşleşme yoksa, istek reddedilir. Bu sayede kötü niyetli istekler engellenmiş olur.

Bu yöntem, Siteler Arası Talep Sahteciliği (CSRF) saldırılarına karşı koruma sağlar. Saldırganın başarılı bir saldırı yapabilmesi için hedef kullanıcının token bilgisini tahmin etmesi gerekir. Oluşturulan tokenin kalitesine bağlı olarak, tahmin edilmesi neredeyse imkansızdır. Ayrıca, her token belirli bir süre için geçerli olmalıdır ve kullanıcı oturumu kapattıktan sonra geçersiz hale getirilmelidir.

Token yönteminin düzgün bir şekilde uygulanabilmesi için, kriptografik olarak güvenli olması gerekir. Böylece; saldırganlar tarafından kolay tahmin edilemeyen, güvenilir bir araç olacaktır.

- Her ne kadar kesin çözüm olmasa da gönderilen isteklerin `**HTTP Referer**` başlıkları da kontrol edilmelidir.

Kullanıcı tarafında alınması gereken önlemler:

- Çerezleri ve site verilerini düzenli olarak temizlemek oldukça önemlidir.
- Bilmediğiniz bağlantılara tıklamamalı, kimden geldiğini anlamadığınız mailleri açarken dikkatli olmalısınız.

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

Bir HTTPRequest in beş ana bölümü vardır -

Verb - GET, POST, DELETE, PUT vb. gibi HTTP yöntemlerini belirtin.

URI - Sunucudaki kaynağı tanımlamak için Tekdüzen Kaynak Tanımlayıcısı (URI).

HTTP Sürümü – HTTP sürümünü belirtin, örneğin HTTP v1.1 .

Resnponse Header – 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 vb.

Response Body – Mesaj içeriği veya Kaynak gösteri

Bir HTTP Response dört ana bölüme sahiptir -

Status/Response Code− İstenen kaynak için Sunucu durumunu belirtin. Örneğin 404, kaynak bulunamadı ve 200, yanıtın tamam olduğu anlamına gelir.

HTTP Sürümü – HTTP sürümünü belirtin, örneğin HTTP v1.1 .

Response Header − Anahtar/değer çiftleri olarak HTTP Yanıt mesajı için meta verileri içerir. Örneğin içerik uzunluğu, içerik türü, yanıt tarihi, sunucu türü vb.

Response Body – Yanıt mesajı içeriği veya Kaynak gösterimi.