**HW#3**

**1** – SOAP vs Restful ?

**SOAP** , REST'ten önce tasarlanmış ve devreye girmiş bir protokoldür. SOAP'ı tasarlamanın arkasındaki ana fikir, farklı platformlar ve programlama dilleri üzerine kurulmuş programların kolay bir şekilde veri alışverişi yapabilmesini sağlamaktı. SOAP, Basit Nesne Erişim Protokolü anlamına gelir.

**REST** , medya bileşenleri, dosyalar ve hatta belirli bir donanım aygıtındaki nesneler gibi bileşenlerle çalışmak için özel olarak tasarlanmıştır. REST ilkelerine göre tanımlanmış herhangi bir web servisi RestFul web servisi olarak adlandırılabilir. Dinlendirici bir hizmet, gerekli bileşenlerle çalışmak için GET, POST, PUT ve DELETE'nin normal HTTP fiillerini kullanır. REST, Temsili Durum Transferi anlamına gelir.

**Farklar:**

- SOAP, Basit Nesne Erişim Protokolü anlamına gelirken, REST Temsili Durum Aktarımı anlamına gelir.
- SOAP bir protokoldür, REST ise bir mimari modeldir.
- SOAP, işlevselliğini istemci uygulamalarına göstermek için hizmet arabirimlerini kullanırken, REST, donanım aygıtındaki bileşenlere erişmek için Tekdüzen Hizmet bulucularını kullanır.
- SOAP, kullanımı için daha fazla bant genişliğine ihtiyaç duyarken, REST'in fazla bant genişliğine ihtiyacı yoktur.
- SOAP ve REST API karşılaştırıldığında, SOAP yalnızca XML biçimleriyle çalışırken, REST düz metin, XML, HTML ve JSON ile çalışır.
- SOAP, REST'i kullanamazken, REST, SOAP'ı kullanabilir.

---

**2** - Difference between acceptance test and functional test ?

Acceptance test - ürünü test edin, tasarladığınız veya oluşturduğunuz niteliklere sahip olduğunu doğrulayın (işlevler, hız, hatalar, tutarlılık vb.)

Functional test - ürünü kendi bağlamında test edin, bu insan etkileşimini (simülasyonunu) gerektirir, orijinal problem(ler) üzerinde istenen etkiye sahip olup olmadığını test edin.

---

**3** - What is Mocking ?

Mocking etmenin amacı kod parçalarını izole etmektir.

Mocking, izolasyon hedefine ulaşmaya yardımcı olan farklı türde teknikler olarak tanımlanır. Bu tekniklere test çiftleri denir ve en yaygın ikisi **Stubs** ve **Spies'tır** ,

---

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

Hedefiniz %100 kapsam ise (tüm özelliklerin %100 test edilmesi yerine), Kod Kapsamı yanıltıcı bir ölçümdür.

- Tüm satırlara bir kez basarak %100 elde edebilirsiniz. Ancak yine de, bu satırların çarpıldığı belirli bir sırayı (mantıksal yol) test etmeyi kaçırabilirsiniz.
- %100 elde edemediniz, ancak yine de tüm %80/sık kullanılan kod yollarınızı test ettiniz. Girdiğiniz her 'atma ExceptionTypeX' veya benzeri savunma programlama korumasını test eden testlere sahip olmak, 'olması gereken' değil 'olması güzel' bir şeydir.

Bu nedenle, kendinize veya geliştiricilerinize eksiksiz olmalarına ve kodlarındaki her yolu kapsayacağına güvenin. Pragmatik olun ve büyülü %100 kapsamın peşinden koşmayın. Kodunuzu TDD yaparsanız, bonus olarak %90'dan fazla kapsama almalısınız.

---

**5** – HTTP/POST vs HTTP/PUT ?

Bir HTTP'nin PUT isteğin gövdesini kabul etmesi ve ardından bunu URI tarafından tanımlanan kaynakta saklaması gerekir.

Bir HTTP POST daha geneldir. Sunucuda bir eylem başlatması gerekiyor. Bu eylem, istek gövdesini URI tarafından tanımlanan kaynakta depolamak olabilir veya farklı bir URI olabilir veya farklı bir eylem olabilir.

PUT, bir dosya yükleme **gibidir .** Bir URI'ye yerleştirme, tam olarak bu URI'yi etkiler. Bir URI'ye yapılan bir POST, herhangi bir etkiye sahip olabilir.

---

**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.

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.

HTTP yöntemine genel bakış

Aşağıdaki tablo, hangi HTTP yöntemlerinin güvenli ve önemsiz olduğunu özetlemektedir:

| **HTTP Method** | **Safe** | **Idempotent** |
|---|---|---|
| GET | Yes | Yes |
| HEAD | Yes | Yes |
| OPTIONS | Yes | Yes |
| TRACE | Yes | Yes |
| PUT | No | Yes |
| DELETE | No | Yes |
| POST | No | No |
| PATCH | No | No |

---

**7** - How does HTTP Basic Authentication work ?

HTTP temel kimlik doğrulama şeması, yalnızca web istemcisi ile sunucu arasındaki bağlantı güvenli olduğunda güvenli kabul edilebilir. Bağlantı güvenli değilse, şema, yetkisiz kullanıcıların bir sunucu için kimlik doğrulama bilgilerini keşfetmesini önlemek için yeterli güvenliği sağlamaz. Bir parolanın ele geçirilebileceğini düşünüyorsanız, kullanıcı kimliğini ve parolayı korumak için SSL şifrelemeli temel kimlik doğrulamasını kullanın.

Bir istemci, sunucunun kimlik doğrulama bilgisi beklediği bir istekte bulunursa, sunucu bir 401 durum kodu, bir kimlik doğrulama hatasını gösteren bir neden ifadesi ve bir WWW-Authenticate başlığı ile bir HTTP yanıtı gönderir. Çoğu web istemcisi, bu yanıtı son kullanıcıdan bir kullanıcı kimliği ve parola isteyerek gerçekleştirir.

HTTP temel kimlik doğrulaması için WWW-Authenticate üstbilgisinin biçimi şöyledir:

```
WWW-Authenticate: Basic realm="Our Site"
```

WWW-Authenticate başlığı, kullanıcı kimliğinin ve parolanın uygulanacağı kaynak kümesini tanımlayan bir bölge özniteliği içerir. Web istemcileri bu dizeyi son kullanıcıya görüntüler. Her bölge farklı kimlik doğrulama bilgileri gerektirebilir. Web istemcileri, son kullanıcıların her istek için bilgileri yeniden yazmalarına gerek kalmaması için her bölge için kimlik doğrulama bilgilerini depolayabilir.

Web istemcisi bir kullanıcı kimliği ve parola aldığında, orijinal isteği bir Yetkilendirme başlığıyla yeniden gönderir. Alternatif olarak, istemci orijinal isteğini yaptığında Yetkilendirme başlığını gönderebilir ve bu başlık sunucu tarafından kabul edilerek sorgulama ve yanıt sürecinden kaçınılabilir.

Yetkilendirme başlığının biçimi:

```
Authorization: Basic userid:password
```

---

**8** - Define RestTemplate in Spring ?

Spring Boot RestTemplate öğreticisi, RestTemplatebir Spring uygulamasında senkronize HTTP isteklerinin nasıl oluşturulacağını gösterir.

*Spring* , popüler bir Java uygulama çerçevesidir ve *Spring Boot* , Spring'in bağımsız, üretim düzeyinde Spring tabanlı uygulamaları kolayca oluşturmaya yardımcı olan bir evrimidir.

RestTemplateHTTP isteklerini gerçekleştirmek için senkronize bir istemcidir. JDK HttpURLConnection, Apache HttpComponents ve diğerleri gibi temel HTTP istemci kitaplıkları üzerinde basit bir şablon yöntemi API'si kullanır.

---

**9** – What is the difference between @Controller and @RestController ?

| @Controller | @RestController |
|---|---|
| @Controller, sınıfları Spring MVC Controller olarak işaretlemek için kullanılır. | @RestController ek açıklaması, RESTful Web hizmetlerinde kullanılan özel bir denetleyicidir ve @Controller ile @ResponseBody ek açıklamasının birleşimidir. |
| @Component ek açıklamasının özel bir sürümüdür. | @Controller ek açıklamasının özel bir sürümüdür. |
| @Controller'da Spring Web MVC'de bir görünüm döndürebiliriz. | @RestController'da bir görünüm döndüremiyoruz. |
| @Controller ek açıklaması, sınıfın bir web denetleyicisi gibi bir "kontrolör" olduğunu gösterir. | @RestController ek açıklaması, sınıfın, @RequestMapping yöntemlerinin varsayılan olarak @ResponseBody semantiğini varsaydığı bir denetleyici olduğunu gösterir. |
| @Controller'da, her işleyici yönteminde @ResponseBody kullanmamız gerekir. | @RestController'da, her işleyici yönteminde @ResponseBody kullanmamız gerekmez. |
| Spring 2.5 sürümüne eklendi. | Spring 4.0 sürümüne eklendi. |

---

**10** – What is DNS Spoofing ? How to prevent ?

İnternete genel 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 şifreleme imzası ekler.

---

**11** – What is content negotiation ?

HTTP'de ***içerik anlaşması*** , kullanıcı aracısının kullanıcı için hangi gösterimin en uygun olduğunu (örneğin, hangi belge dili, hangi görüntü formatı veya hangisi) belirlemesine yardımcı olmak için bir kaynağın farklı temsillerini aynı URI'ye sunmak için kullanılan mekanizmadır. içerik kodlaması).

---

**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.

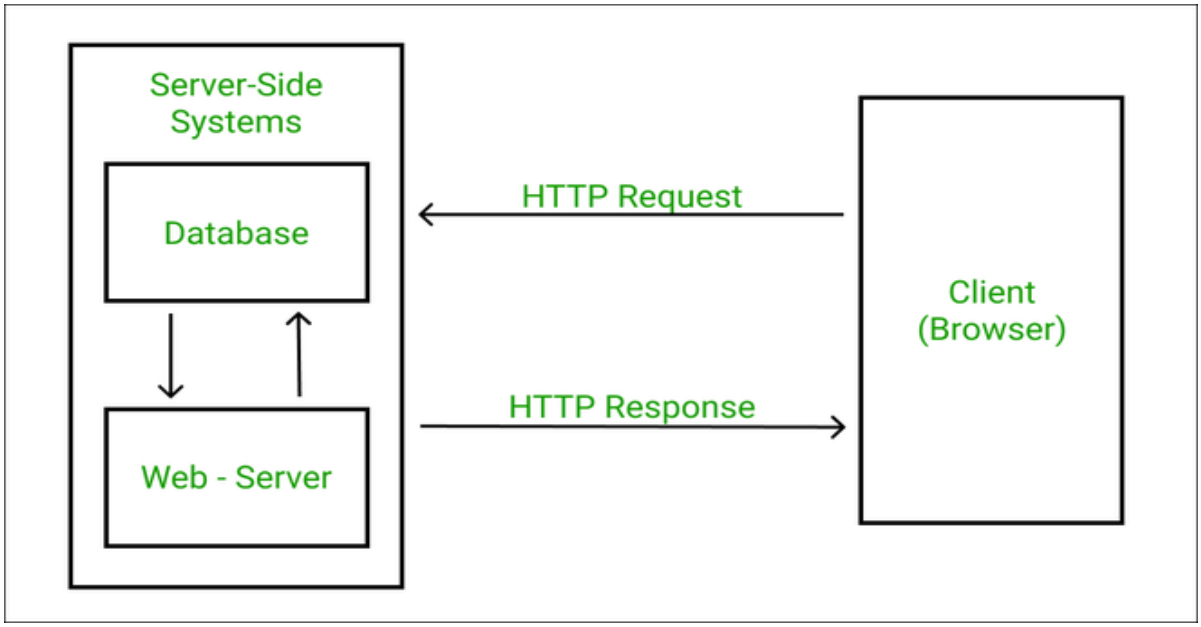
---

**13** - What is CSRF attack? How to prevent ?

CSRF, Sea Surf veya XSRF olarak da bilinen Siteler Arası İstek Sahteciliği , bir saldırganın bir kurbanı kendi adına eylemler gerçekleştirmesi için kandırdığı bir saldırıdır. Saldırının etkisi, kurbanın sahip olduğu izinlerin düzeyine bağlıdır. Bu tür saldırılar, bir web sitesinin kullanıcıya tamamen güvendiği gerçeğinden yararlanır ve bu, kullanıcının gerçekten söylediği kişi olduğunu onaylayabildiğinde. Sunucu tarafından doğrulanabilen bir değere sahip ek bir parametre ekleyerek CSRF saldırılarını önleyebiliriz. **Aşağıda, siteler arası istek sahteciliği saldırılarını engellemek için kullanabileceğiniz bazı yöntemlerin bir listesi bulunmaktadır.**

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

Aşağıda İstek-Yanıt Döngüsünü gösteren bir resim bulunmaktadır:

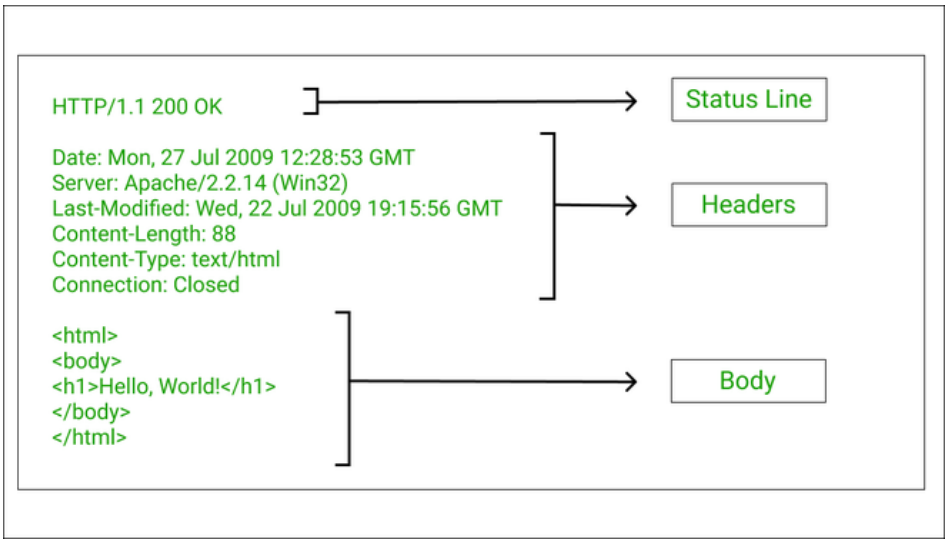



| Classification | ID / Severity |
|---|---|
| PCI v3.2 | 6.5.9 |
| OWASP 2013 | A8 |
| OWASP 2017 | A5 |
| CWE | 352 |
| CAPEC | 62 |
| WASC | 9 |
| HIPAA | 164.306(a) |
| ISO27001 | A.14.2.5 |
| Invicti | Low |
| | |

**HTTP Yanıtının Yapısı:** Yukarıda istek-yanıt gibi, HTTP Yanıtı, istemcinin kolayca anlayabilmesi için takip edilen özel bir yapıya sahiptir. Herkesin takip ettiği bir Evrensel Dil vardır ki insanlar arasında iletişim kopukluğu olmasın. HTTP Yanıtı genel olarak 3 ana bileşene sahiptir:

- Status Line
- Headers
- Body (Optional)

Bir bütün olarak bir HTTP Yanıtı aşağıdaki resme benzer:




---

Kaynak:

1. https://www.invicti.com/blog/web-security/csrf-cross-site-request-forgery/
2. https://www.netsparker.com.tr/blog/web-guvenligi/http-guvenlik-headerlari/
3. https://www.geeksforgeeks.org/state-the-core-components-of-an-http-response/#:~:text=HTTP%20Response%20broadly%20has%203,Body%20(Optional)