PATİKA TÖDEB
JAVA SPRİNG BOOTCAMP

ÖDEV 3

BATUHAN AYYILDIZ

1-

Soap bir protokoldür. Rest ise mimari tarzıdır. Bu yüzden Soap resti kullanamaz ama Rest Soap'ı kullanabilir. Hatta mimari bir konsept olmasından dolayı Rest herhangi bir protokolü kullanabilir. Örneğin http vb. Soap'ın takip edilmesi gereken daha katı standartları vardır. Ayrıca sadece Xml data tipine izin verir. Soap'ın aksine Rest çeşitli data tiplerine izin verir. Mesela Html, Xml, Json vb. Rest Soap'a göre daha az bant genişliği ister Rest business logic oluşturmak için URI kullanır. Soap da servisler ve interface'leri kullanır.

2-

Functional test --> Ürünün, geliştiricinin çalışacağını düşündüğü şekilde çalıştığını doğrular.

Acceptance test --> Ürünün, çözmek için üretildiği problemi çözdüğünü doğrular.

3-

Gerçek objelerin davranışları simüle veya taklit etmek için obje üretilmesidir. Genelde unit test için kullanılır.

4-

%100 'e yakın code coverage yapmaya çalışmak sıkıntı yaratabilir. Bakımı yapılması gereken low value testlerden technical debt oluşmasına sebep olur. İşlem yükünü de arttırır. %100 code coverage kapsama alanından branch'lerin ya da line'ların doğru test edildiğini garanti etmez. Düşük yüzde ise ürünün büyük kısımlarının tamamen test edilmediği anlamına gelir. Yüzdeler işin gerekliliklerine , ürünün veya ürünlerin karmaşıklığına göre değişebilir. ama yakalşık sayı verilmesi gerekirse %60 kabul edilebilir, %75 övgüye değer, ve % 90'sa doğru örnek olarak gösterilebilir.

5-

HTTP post ve put çok yakın işlevlere sahiptir . Put idempotent'tir. Birden fazla request yapıldığında tek request çağırmasına denk olacaktır. Post ise idempotent değildir. Yani A sayısı kadar request yollanırsa A resources'a sahip olunup server'da A tane farklı URI oluşturulacaktır. PUT belirli ve spesifik bir URI'ye dosya ve kaynak yerleştirmek orada kaynak, dosya varsa da onu değiştirmek için kullanılır. POST veriyi belirli bir URI'ya yollar ve URI'daki kaynağın isteği halletmesini bekler. PUT genelde update için POST da create için kullanılır.

Örnek put ve post command yazımı

PUT/questions/{question-id}

POST/questions

6-

Bir serverdaki durumu değişitirmiyorsa yani read-only operation yapıyorsa safe'dir. Tüm safe method'lar idempotent'tir. Ama bazı methodlar idempotent olup unsafe olabilir. Mesela PUT ve DELETE.

Unsafe Methods: PUT ,POST, DELETE, PATCH

Safe Methods: GET, HEAD, OPTIONS, TRACE

7-

Çalışma şekli alttaki gibidir.

1- Kullanıcı server'dan protected resource ister.
2- Server, kullanıcıdan kullanıcı adı ve şifre ister.
3- Kullanıcı, kullanıcı adı ve şifresini server'a yollar.
4- Server kullanıcının kimliğini doğrular ve eğer doğruysa istenen resource'u yollar.

8-

Rest template, spring'te client-side HTTP erişimi sağlayan bir sınıftır. HTTP methodlara karşılık gelen kendi methodlarını barındırır.

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-

@RestController RESTful Web Service'lerinde kullanılan bir anotasyondur ve @Controller ve @ResponseBody anotasyonlarının kombinasyonundan oluşur. @RestController @Controller'ın özelleşmiş versiyonudur. @Controller'sa @Component'ın özelleşmiş versiyonudur. @Controller için her method handler'da @ResponseBody kullanmak zorunludur ama @RestController'da böyle bir gereklilik yoktur. @RestController'da her istek işleme methodu dönen objeleri HttpResponse'a otomatik olarak serialize eder. @Controller anotasyonunda, bu işi yapmak için @ResponseBody kullanmak gereklidir.

10-

DNS spoofing, DNS ön zehirlenmesi olarak geçer. Saldıranın DNS server'a zararlı girdiler vererek DNS server'ın hedeflenen kullanıcıyı saldırganın kontrolündeki zararlı bir website'sine yönlendirmesini sağlamasıdır. Genelde halka açık kablosuz ağlarda bu saldırı gerçekleşir. Bu saldırıyı önlemek için DNSSEC(DNS Security) kullanılabilir. Domain sahibi DNS girdilerini kurduğunda , DNS aramaları doğrulanmadan önce DNSSEC çözümleyiciler tarafından istenen girdilere kriptografik imza ekler.

11-

Content Negotiation, HTTP'nin bir parçası olarak tanımlanmıştır. Aynı URI'de bir belgenin farklı sürümlerini sunmayı olası kılan mekanizmayı ifade eder. Böylelikle kullanıcı aracısı (user agent) kullanıcıya hangi sürümün veya temsilin en uygun olduğunu belirler.

12-

Statelessness olduğunda server, kullanıcı tarafındaki hiçbir bilgiyi server tarafında depolamaz. Bu yüzden kullanıcı tarafından yollanan her istek gerekli olan tüm bilgileri içermelidir. Server tarafında, ileride referans olarak kullanmak için bilgi depolanmaz.

Avantajları

- Karmaşıklık azalır çünkü State synchronization gerekli değildir.
- Performasn artar çünkü server kullanıcın request'lerini takip etmek zorunda değildir.
- Kolay adaptasyon. Herhangi bir server herhangi bir kullanıcı request'iyle ilgilenebilir.

Dezavantajları

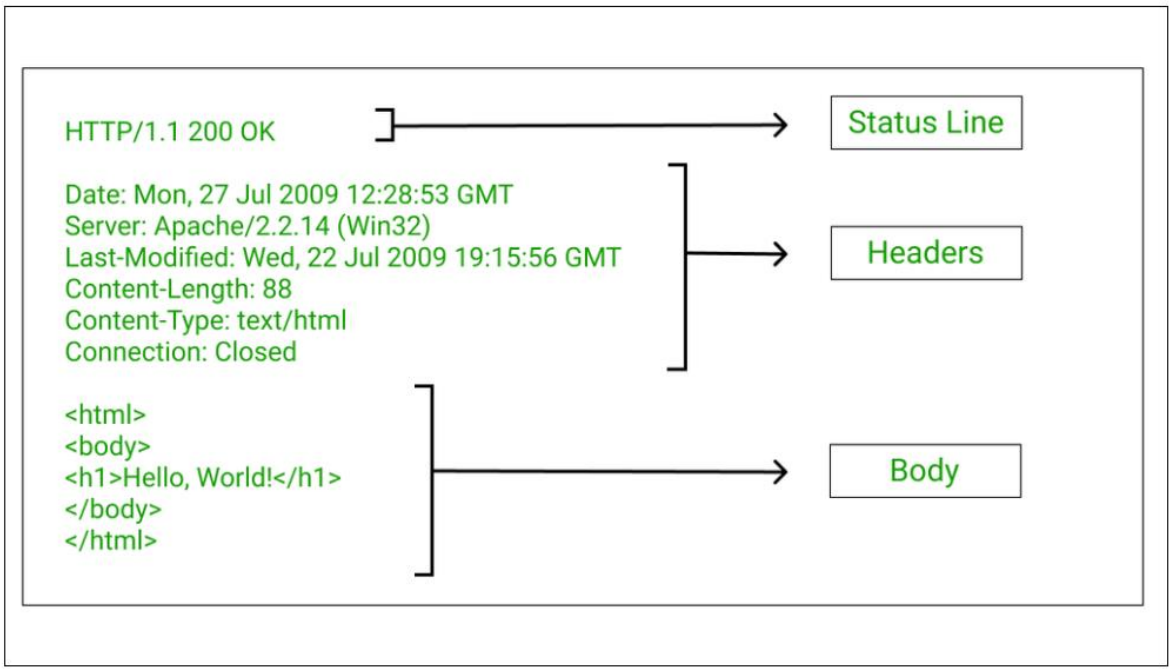
- Kullanıcı tüm gerekli bilgileri içeren bir request yollaması gerektiği için request boyutu artar.

13-

CSRF(Cross-site request forgery), kullanıcıdan web uygulamasına zararlı istek göndermek için kullanılır. Bu saldırılar veri hırsızlığına değil, olan durum değişikliklerine odaklanır. Saldıran HTTP response içeriklerine ulaşamaz. Bu saldırıdan korunmak için referer header kullanmak, HttpOnly flag kulllanmak gibi yollar vardır. Ama bu yollar yeterince iyi çözüm sağlamayabilir. Onların yerine belirli bir kullanıcıyla eşleştirilmiş token'ler kullanılabilir. Bu token web uygulamasındaki her durum değişikliğinde gizli olarak yollanır. Anti-CRSF token diye adlandırılır

14-

HTTP Response, 3 tane ana bileşene sahiptir. Bunlar status line (durum belirtmek için kullanılır), Headers (Tarih, server ismi gibi özellikleri göstermek için kullanılır), Body(Mesajı göstermek için kullanılır). Body isteğe bağlı oluşturulur.



HTTP request 5 tane ana bileşene sahiptir.

VERB ---> POST, DELETE gibi HTTP methodları kapsar

URI ---> Serverdaki kaynağı belirtmek için kullanılır

HTTP Version ---> HTTP versiyonu belirtmek için kullanılır. HTTP v1.1 gibi

Request Header ---> HTTP request için metadata içeriğidir. Örneğin Kullanıcı tipi, kullanıcı tarafından desteklenen format gibi içerikleri barındırır.

Request Body ---> Mesajın içeriği ya da kaynağın sunumunu içerir