SOA - Web Service - Restful Service - HTTP methods kavramlarını örneklerle açıklayınız.

...

Service-Oriented Architecture (SOA):

SOA temel olarak her hizmetin farklı birimler tarafından birbirinden bağımsız olarak çalışmasını ifade eder. İş uygulamaları oluşturmak için hizmetler adı verilen yazılım bileşenlerini kullanan bir yazılım geliştirme yöntemi olduğu da söylenebilir. Her hizmet, bir iş özelliği sunar ve hizmetler, farklı platform ve diller arasında birbiriyle iletişim de kurabilir. Geliştiriciler, hizmetleri farklı sistemlerde yeniden kullanmak veya birkaç bağımsız hizmeti karmaşık görevleri gerçekleştirmek amacıyla birleştirmek için SOA kullanır.

SOA geniş kapsamı olan bir mimari yaklaşım olup çeşitli prensiplere sahiptir.

Loose Coupling: Servislerin birbirine gevşek olarak bağlı olduğunu belirtir. Böylece bir servis diğer servisten bağımsız bir şekilde çalışabilir.

Interoperability: Servislerin diğer servislerle birlikte çalışabilir olduğunu belirtir. Birlikte çalışabilirlik için ortak bir biçim kullanılır.

Reusability: Servislerin tekrar kullanılabilir olduğunu belirtir.

Abstraction: Servis iç yapısının servis kullanıcıları tarafından gizlenmesidir.

Facade: Servis ve servisi kullanan arasındaki bir bileşen/kabuk olduğunu belirtir.

Autonomy: Bir servisin diğer servislerden bağımsız olarak çalışabilir olduğunu belirtir.

Statelessness: Servislerin durumsuz olduğunu belirtir. Servislerin durum bilgisi servis isteğine göre şekil alacağını ve sürekli aynı olmadığını belirtir.

Discoverability: Servislerin keşfedilebilir olduğunu belirtir.

Web Service:

Verileri web sayfası dışında tüm cihazlara göndermek istenildiğinde devreye Web Service kavramı girer. Web Service ile platform bağımsız tüm cihazlara veri aktarımı gerçekleştirilir. Web servisleri bir ağ (genellikle internet) üzerinden farklı yazılım uygulamaları arasında iletişim ve veri alışverişini sağlayan bir teknolojidir. Web servisleri, farklı programlama dilleri kullanılarak geliştirilen ve farklı platformlarda çalışan farklı sistemlerin birbirleriyle sorunsuz bir şekilde etkileşime girmesine olanak tanır.

Web hizmetleri, bir uygulamanın veri veya işlevsellik isteyen istemci gibi davrandığı ve başka bir uygulamanın istenen veri veya işlevi sağlayan sunucu gibi davrandığı istemci-sunucu modelinde çalışır. Bu etkileşimler genellikle HTTP, XML, JSON, SOAP ve REST gibi standart protokoller ve formatlar kullanılarak gerçekleştirilir.

İki ana web hizmeti türü vardır: SOAP ve REST

Restful Service:

Representational State Transfer (REST) ilkelerine dayanan RESTful Web Hizmetleri, ölçeklenebilirliği, basitliği ve güvenilirliği teşvik etmek için tasarlanmış bir dizi mimari yönergeye uyan bir web hizmeti türüdür. RESTful Web Hizmetlerinin temel kavramlarının ve özellikleri şöyle özetlenebilir:

Statelessness:

REST'in temel ilkelerinden biri durumsuzluktur, yani bir istemciden sunucuya gelen her isteğin, bu isteği anlamak ve yerine getirmek için gerekli tüm bilgileri içermesi gerekir. Sunucu, istekler arasında herhangi bir istemci içeriğini saklamaz. Bu, sunucu uygulamasını basitleştirir ve ölçeklenebilirliği artırır.

Resource-based:

RESTful hizmetleri, bir URI tarafından tanımlanabilecek herhangi bir bilgi veya kavram olabilen kaynaklar etrafında odaklanır. Kaynaklar GET, POST, PUT ve DELETE gibi standart HTTP metodları kullanılarak işlenir. Her kaynak genellikle JSON veya XML gibi farklı temsillerle temsil edilir.

Uniform Interface:

RESTful hizmetleri, kaynaklar üzerinde eylemler gerçekleştirmek için genellikle HTTP metodlarını kullanarak istemciler ve sunucular arasında tek tip bir arayüz sağlar. Bu tekdüzelik mimariyi basitleştirerek anlaşılmasını ve ölçeklendirilmesini kolaylaştırır.

Representation:

RESTful hizmetlerindeki kaynaklar, HTTP isteklerinin ve yanıtlarının yükü olarak gönderilen JSON veya XML gibi bir formatta temsil edilir. İstemciler ve sunucular, HTTP başlıkları gibi içerik anlaşma mekanizmalarını kullanarak temsil formatı üzerinde anlaşabilirler.

State Transfer:

RESTful hizmetleri, bir kaynağın durumunu sunucudan istemciye veya tam tersi şekilde aktarır. Bu durum verileri, meta verileri veya hipermedya bağlantılarını içerebilir. İstemciler, URI'lere istek göndererek kaynaklarla etkileşime girer ve sunucular bu kaynakların temsilleriyle yanıt verir.

HTTP Methods:

RESTful hizmetleri, kaynaklar üzerinde CRUD (Oluşturma, Okuma, Güncelleme, Silme) işlemlerini gerçekleştirmek için standart HTTP metodlarını kullanır:

GET: Bir kaynağın temsilini alır.

POST: Yeni bir kaynak oluşturur.

PUT: Mevcut bir kaynağı günceller.

DELETE: Bir kaynağı siler.

PATCH, HEAD, OPTIONS vb. gibi ek HTTP metodları da belirli amaçlar için kullanılabilir.

HATEOAS:

HATEOAS, RESTful hizmetlerinde, istemcilerin kaynak temsillerine gömülü hipermedya bağlantıları aracılığıyla hizmetin yeteneklerinde dinamik olarak gezinmesine olanak tanıyan bir mimari kısıtlamadır. Bu, hizmeti daha keşfedilebilir ve kendini açıklayıcı hale getirir.

Ölçeklenebilirlik ve Performans:

RESTful hizmetleri, durumsuzlukları ve kaynak tabanlı yapıları nedeniyle doğası gereği ölçeklenebilirdir. Dağıtılmış sistemler ve mikro hizmet mimarileri için çok uygundurlar. Ek olarak, RESTful API'lerin hafif doğası çoğu zaman diğer web hizmetleri türlerine kıyasla daha iyi performansa yol açar.

Özetle RESTful Web Hizmetleri, web üzerinde dağıtılmış sistemler ve API'ler oluşturmaya yönelik esnek, ölçeklenebilir ve standartlaştırılmış bir yaklaşım sağlar. Geliştiriciler, REST ilkelerine bağlı kalarak anlaşılması, bakımı ve ölçeklendirilmesi kolay sistemler oluşturabilir.

HTTP methods:

Yukarıda da biraz bahsedildiği üzere http metodları HTTP (Hypertext Transfer Protocol), bir URI tarafından tanımlanan bir kaynak üzerinde gerçekleştirilmesi istenen eylemi belirten çeşitli metodları tanımlar. Bu metodlar istemci-sunucu iletişiminde kaynak üzerinde gerçekleştirilecek işlemi belirtmek için kullanılır. Aşağıda örneklerle birlikte yaygın http metodlarının bir açıklaması verilmiştir:

GET:

GET methodu belirtilen kaynağın temsilini ister. Verileri değiştirmeden sunucudan alır.

Örnek:

GET /api/users/123 HTTP/1.1

Host: example.com

Bu örnekte istemci, sunucudan ID 123'e sahip kullanıcının ayrıntılarını istiyor.

POST:

POST methodu, sunucu tarafından işlenecek verileri gönderir. Genellikle yeni bir kaynağın oluşturulmasıyla veya sunucuda belirli bir eylemin yürütülmesiyle sonuçlanır.

Örnek:

POST /api/users HTTP/1.1

Host: example.com

Content-Type: application/json

{"username": "john_doe", "email": "john@example.com"}

Bu örnek, sağlanan kullanıcı adı ve e-posta adresiyle yeni bir kullanıcı oluşturmak için sunucuya veri gönderir.

PUT:

PUT methodu, belirtilen kaynağı sağlanan verilerle günceller. Varsa kaynağın tamamını değiştirir, yoksa yenisini oluşturur.

Örnek:

PUT /api/users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{"username": "jane_doe", "email": "jane@example.com"}

Bu örnek, ID 123'e sahip kullanıcının ayrıntılarını sağlanan kullanıcı adı ve e-posta adresiyle günceller.

DELETE:

DELETE methodu belirtilen kaynağı sunucudan siler.

Örnek:

DELETE /api/users/123 HTTP/1.1

Host: example.com

Bu örnekte istemci, ID 123'e sahip kullanıcının sunucudan silinmesini talep etmektedir.

PATCH:

PATCH methodu, bir kaynağa kısmi değişiklikler uygulamak için kullanılır. Yalnızca kaynağın belirtilen alanlarını günceller.

Örnek:

PATCH /api/users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{"email": "updated_email@example.com"}

Bu örnek, diğer alanları değiştirmeden, ID 123'e sahip kullanıcının e-posta adresini günceller.

OPTIONS:

OPTIONS metodu, desteklenen HTTP metodları veya CORS ilkeleri gibi belirtilen kaynak için kullanılabilen iletişim seçenekleri hakkında bilgi ister.

Örnek:

OPTIONS /api/users HTTP/1.1

Host: example.com:

Bu örnekte istemci, sunucudan /api/users kaynağıyla etkileşime geçmek için desteklenen metodlar hakkında bilgi sağlamasını istiyor.

En sık kullanılan HTTP metodları yukarda anlatılmıştır.

Sonuç olarak bu HTTP metodları, istemcilerin sunucudaki kaynaklarla etkileşime girmesi için standartlaştırılmış bir yol sağlayarak çeşitli CRUD (Oluşturma, Okuma, Güncelleme, Silme) işlemlerinin ve web uygulamaları ve API'lerdeki diğer eylemlerin uygulanmasına olanak tanır.