ANDREAS LÖNNHOLM – WEB SERVICES – INLÄMNING 1

VAD ÄR EN WEB-SERVICE?

En webservice är i grund och botten en server som erbjuder en mjukvarutjänst utåt, alltså förmedlar information mellan en klient och en server på Internet.
De vanligaste teknikerna som används är SOAP och RESTful.

Exempel på detta är Amazon, Bank-ID, Spotify eller Google.

VAD ÄR ETT API?

API – mer exakt kallat *Application Programming Interface* – är som ett mellan-lager för att olika system ska kunna utbyta information mellan varandra. En sorts tolk mellan två olika tjänster som ska prata med varandra.

Lite lätt förenklat kan man säga att det är en funktion som väntar på frågor den ska förmedla vidare.

Ett exempel är om ingående fråga (request) vill ha all information om en användare, och om detta finns i en yttre databas, då ska API:et fråga databasen i sin tur om detta. Om svar finns ska det förmedlas vidare till klienten i ett format som kan hanteras. Om inget kan förmedlas, ska ett meddelande om att detta inte är möjligt skickas, samt en angiven felkod i http-format.

HUR ANVÄNDS VANLIGTVIS ETT WEB-API?

När du hämtar information från ett API får du rådata, alltså inget grafiskt utan bara ren information. Oftast i JSON- eller XML-format.

Ett exempel kan vara en vädertjänst, där du kan få information kring temperatur vid olika tillfällen för en ort, men även vindhastighet och riktning samt när solen går upp med mera.
Detta gör det lättare för dig som utvecklare att fokusera på din app eller hemsida utseende och upplägg och sen låta en yttre API-tjänst förmedla själva informationen.

Exempel på webb-API är:

- ***Google Maps* API**– för att kunna visa kartor på en hemsida, men även för att kunna välja optimal rutt mellan olika platser och visa upp detta.

Slutsats är att ett Web-API:er erbjuder en tjänst till utvecklaren, som denne inte behöver bygga själv.

VAD ÄR ETT REST-API?

Ett REST-API är ett begrepp som beskriver tjänstens uppbyggnadssätt.

För att uppfylla kraven måste följande vara uppfyllt:

- **Stateless**
  Innebär att ingen kommunikation som sker mellan klient och server lagrar någon sorts sessionshantering, alla nya request ses som en helt nya, inga tidigare uppgifter lagras.

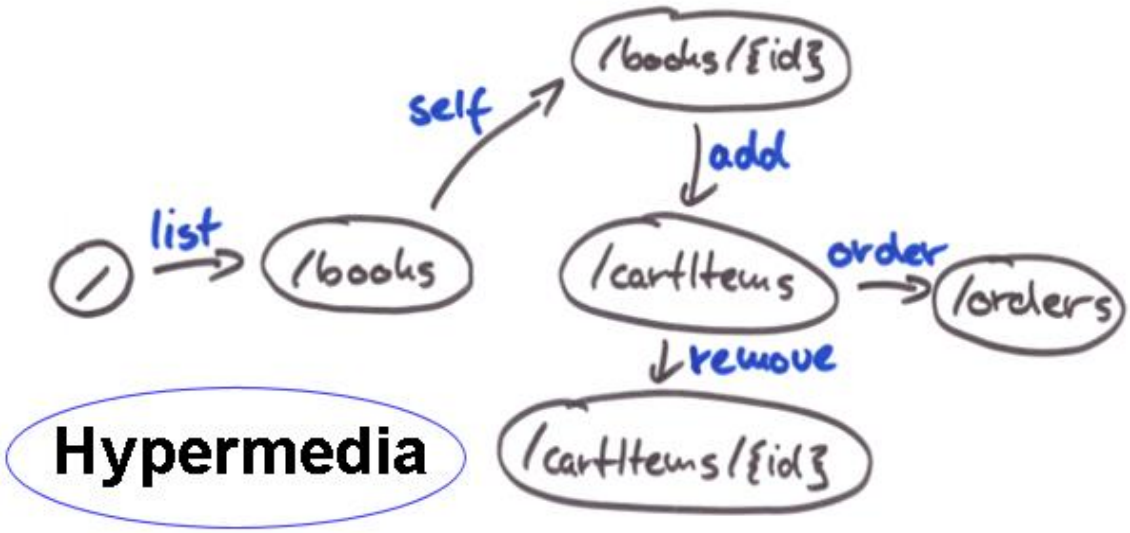
- **Cacheable data**
  Caching är förmågan att lagra information som används ofta på ett ställe så det går snabbare att få tillgång till det, en sorts proxy-tjänst.

  Exempelvis lagra data i en array i stället för att konstant fråga databasen om samma information om och om igen. Detta för att minska användningen av bandbredd, minska väntetider samt reducera belastning på servrar och system.

- **Self-descriptive**
  Är när varje funktion förklarar vad den gör på ett tydligt sätt.
  Detta är ett sätt att låta koden i sig vara dess dokumentation.

- **Hypermedia**
  Handlar om att REST inte bara ska leverera data utan också länkar till relevanta källor. Till exempelvis hyperlänken till sig själv och andra relevanta närliggande källor. (se bild nedanför)

- **Layered system**
  Innebär att delar upp vilka delar av koden som gör vad.
  Du har kanske en klass för controller, en för service och en tredje för repository. Utöver det kan du ha en server för API, en annan för databas och en tredje för autensifiering.
  Med andra ord delas ansvar och funktion upp mellan olika delar.



NÄMN MINST TVÅ SKILLNADER MELLAN SOAP OCH REST.

- REST är arkitektur-baserat och returnerar data i form av bland annat JSON, vars upplägg bestäms av den som programmerar.
  SOAP är protokoll-baserat och returnerar data som är uppställd enligt satta regler.
- REST använder HTTPS och SOAP WS Security
- GET, POST, PUT med flera jämfört med SMTP, HTTP Post, MQ

NÄMN MINST TRE SÄKERHETSHOT FÖR WEB-SERVICES.

1. **SQL Injektion**

   Vad är det:
   En SQL injection-attack utnyttjar en säkerhetsbrist som har sin grund i att utvecklaren har misslyckats med att isolera extern fientlig information från webbapplikationens inre maskineri. Det gör det möjligt för en angripare att köra kod eller kommandon.

   Hur löser man det:
   - Validera all indata från användare
   - Skapa prepared-statement för att undvika att användare ger direkt-kommandon till DB, på så sätt validera input.

Liknande lösning finns som en stored procedure, som bara accepterar vissa indata som parameter.

2. **Validering**

   Vad är det:
   Att bekräfta att rätt värden förs in och att koden uppfyller den standard som krävs. En validering är en sorts kvalitetskontroll.

   Hur löser man det:
   - man validerar sin kod, både input så som kodupplägg.

3. **Osäkra meddelanden**

   Vad är det:
   - Okrypterad kommunikation. Kan avlyssnas.
   - Icke autensifierad kommunikation. Kan avlyssnas och ändras på vägen.
   - att webbläsaren visar en röd triangel, att HTTPS med certifikat inte har bekräftats gentemot pålitligt yttre företag. Du kan vid dessa tillfällen inte helt lita på att vad du skickar hanteras säkert.

   Hur löser man det:
   - Se till att använda krypterad kommunikation när du skickar och mottager kritiskt information.
   - Om man är websidans ägare behöver man ansöka om ett certifikat som bekräftats av ett yttre företag. Som användare kan man dock bara vara försiktig med vilken info man ger ut.

4. **Intern Attack**

   Vad är det:
   Intern attack är när personer inom det egna företaget/organisation läcker ut eller stjäl information som är värdefullt för företaget. Ibland kanske utan att ens mena det.

   Till exempel när en anställd råkar ge ut sitt användarnamn och lösenord till en yttre part, som sen då har åtkomst till företaget.

   Hur löser man det:
   - Ett sätt är två-vägs-autensifiering, för att vara säker på att rätt person loggar in.
   - Gärna en fysisk koppling vid inloggning till tjänster, i form av ett kort eller liknande.

5. **Dictionary Attack**

   Vad är det:
   När man försöker få tillgång till till exempel ett konto, så provar du alla olika kombinationer i en lång lista (ett dictionary / ordlista). Kan även kallas brute-force-attack, då den vare-sig är snabb eller effektiv jämförelse med andra metoder.

   Hur löser man det:
   - Långa lösenord med udda symboler, där man helst undviker normala ord (städer, platser, namn osv..)
   - Se till att tjänsten som hanterar lösenord har dem hashade, gärna med salt eller en bra, säker hash-metodik.

6. **Öppen dokumentation**

   Vad är det:
   antar det är att dokumentation är så informativ att man förstår systemet så bra att man då får tillgång till mer än man borde, och just därför är det lättare att utföra en attack mot systemet.

   Hur löser man det:
   - Undvik allt för öppen dokumentation och låt funktioner i systemet i stället vara självförklarande.

VAD ÄR EN HYPERLINK?

En hyperlink är en länk som refererar till information som en användare kan få ta del av om den klickar på den med musknappen. Lite kort kan man säga att det är en sorts referens-text som hänvisar till en annan plats på nuvarande sida eller till en annan källa.

HUR KAN MAN ANVÄNDA HYPERLINKS I API:ER?
VARFÖR KAN DET VARA BRA?

- *Hur kan man använda hyperlinks i API:er?*

  Ett första exempel är om användaren är inne i sin varukorg. I det fallet kanske personen vill se sin leveransadress eller kontouppgifter genom att hänvisas till länkar med denna info.

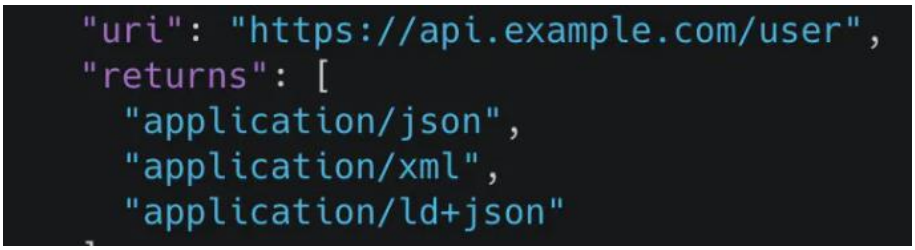
  Ett annat exempel från nuvarande kurs är att när information kring olika länder hämtas, så visar hyperlinks en direktlänk till information om huvudstäderna.

- *Varför kan det vara bra?*

Du kan visa relaterade ämnen för att ge användaren en större helhetsförståelse kring hur API:et är uppbyggt och för att därmed lättare navigera mellan olika komponenter personen kan behöva ha tillgång till. Detta minskar behovet av ytterligare dokumentation och förklarningar.

Exempel som visar relevanta hyperlänkar.

Ett annat exempel kan vara att visa i vilket format man får svar



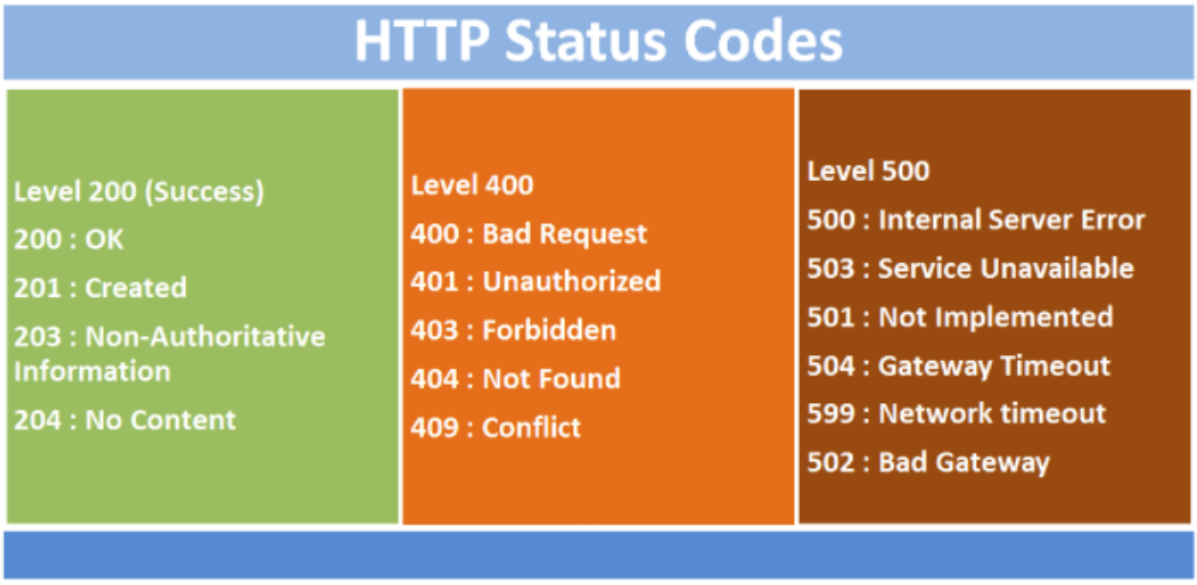
…och ett sista men ack så vanligt sätt är när du paginerat en visning (har flera sidor av samma information) att helt enkelt visa en länk till både nästa och föregående sida.

Se även svar och bild kring hypermedia från frågan om REST-API!

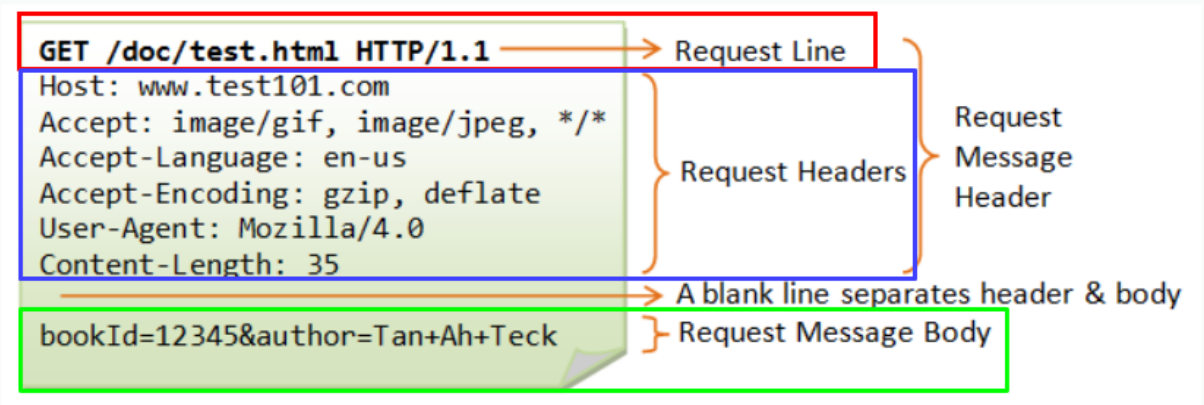
VAD ÄR HTTP?

HTTP – Hypertext Transfer Protocol - är ett protokoll i OSI-nätverksmodellens applikations-lager. Protokollet används när främst webbläsare kommunicerar med en webbserver, och http tar då hand om kommunikationen för HTML-sidor, filer och grafik över Internet.

När vi jobbar med SpringBoot tänker vi nog mest på HTTP i form av de http-status-reponser vi får tillbaka som svar när vi gör en request mot API:et.



VAD KALLAS DE TRE DELAR SOM HTTP-MEDDELANDEN ÄR UPPBYGGDA AV?



Exempel där ***Startline*** är ***rött***, ***Headers blått*** och ***Body*** är ***grönt***.

1. **Startline**
   Innehåller en http-metod (get, put, post, patch…) som beskriver vad som ska utföras, samt angiven version av protokollet (exempel är HTTP/1.1)

2. **Headers**
   Kan innehålla många olika värden. Vanligt är att ange User-agent (webbläsare eller applikation som används), accepterade språk, encoding samt hur man önskar svaret man får tillbaka ska vara (ofta önskar du nog JSON-format här)

3. **Body**
   Body innehåller vad du skickar med eller tar emot tillbaka för värden/innehåll. Kort sammanfattat – den data som rör sig mellan klient och API.

VAD ÄR DET FÖR SKILLNAD MELLAN HTTP OCH HTTPS?

HTTPS är HTTP med kryptering i form av TLS-SSL, vilket resulterar i att HTTPS är säkrare. I OSI-nätverksmodellen ligger http på applikationslagret och https på transportlagret.

I webbläsarens adressfält ser du om du om sidan använder http eller https både genom att se det i text, och om hänglås-ikonen vid sidan av är låst (säker = https) eller olåst (http = osäker). Se bild nedanför.