1.7 API-teknologier

Ulike API-teknologier som Soap og Restful, verktøy for utvikling/verifisering.

Dette kapittelet er mest ment for utviklere og IKT-folk, men greit for prosjektdeltakere å ha noe begrepskunnskap om saken.

Ut fra foregående kapittel kan rammebetingelsene for ett prosjekt medføre at en ikke kan benytte åpne realtime API for en integrasjon, men i dette kapittelet forutsetter vi bruk av API av typen REST eller SOAP. En vil kunne finne de som fortsatt benytte API av type SOAP, samt asynkrone meldings-baserte API, men pr 2021 er REST å anse som en standard.

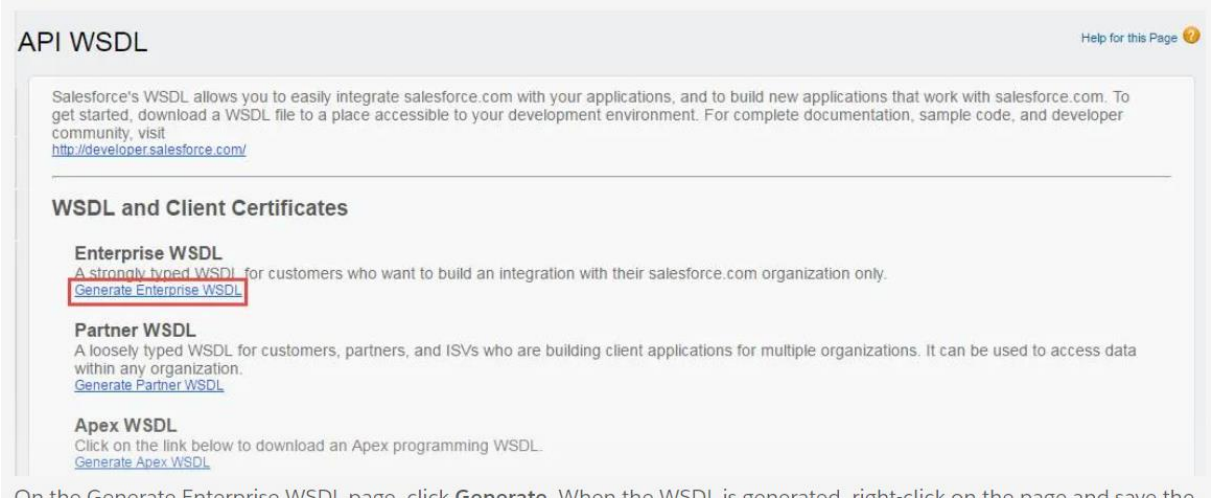
SOAP

SOAP betyr Simple Object Access Protocol

SOAP er sterkt avhengig av XML, og definerer sammen med skjemaer et funksjonsrikt meldings-rammeverk.

Hver operasjon tjenesten gir er eksplisitt definert, sammen med XML -strukturen til forespørselen og svaret for den operasjonen.

Hver inngangsparameter er på samme måte definert og bundet til en type: for eksempel et heltall, en streng eller et annet komplekst objekt.



Alt dette er kodet i WSDL - Web Service Definition Language (eller Definition, i senere .WSDL forklares ofte som en kontrakt mellom leverandøren og forbrukeren av tjenesten. Med hensyn til programmering kan WSDL betraktes som en metodesignatur for webtjenesten.

Mer om SOAP:
https://trailhead.salesforce.com/en/content/learn/modules/api_basics/api_basics_soap

https://www.uio.no/studier/emner/matnat/ifi/INF5040/h04/gruppe/pres/group5/SOAP.pdf

REST For Webutviklere
Uttrykket «REST» står for Representasjonsoverføring. Dette kan høres litt forvirrende, og wikipedia-beskrivelsen gjør det kanskje enda mer forvirrende. Men det er mulig å forenkle terminologien.

REST er bare en rekke retningslinjer og arkitektoniske stiler som brukes til dataoverføring. Det brukes vanligvis til webapplikasjoner, men kan også til datautveksling mellom applikasjoner.
Uttrykket «API» står for «Application Programming Interface», som er metoder til bruk for utvikler for å kunne lage tilkobling mot andre biblioteker eller applikasjoner. Windows har flere API-er, og Twitter har også en web-API, selv om de utfører forskjellige oppgaver med forskjellige mål.

Kombinerer vi dette, er RESTful API-er å betrakte som API-er som følger REST-arkitekturen.

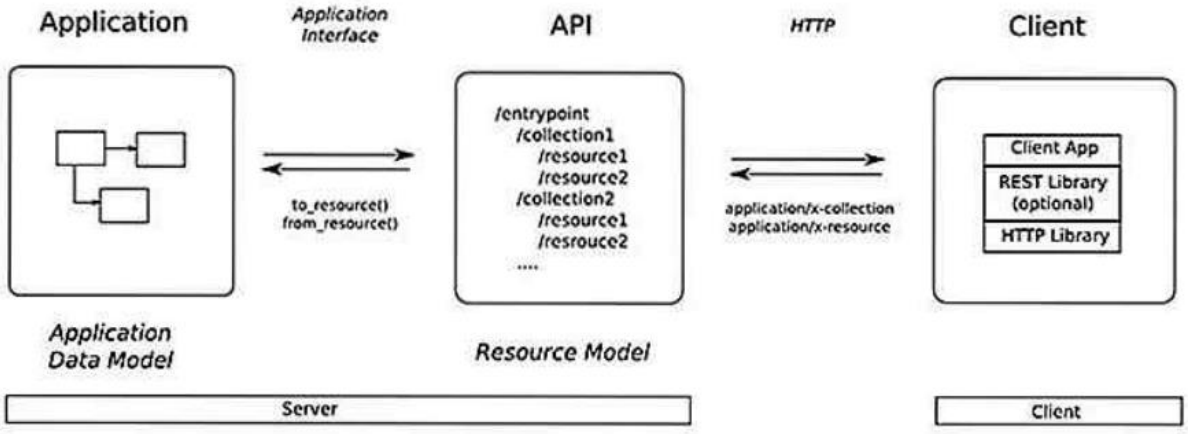
Hva er så REST-arkitekturen?
I henhold til REST-arkitekturen (Representational "State" Transfer) lagrer ikke serveren noen tilstand om klientprosessen på serversiden.
Applikasjonens sesjonstilstand beholdes derfor utelukkende på klienten. Klienten er ansvarlig for å lagre og håndtere sesjonsrelatert informasjon på sin egen side.
Denne begrensningen kalles Statelessness
Statelessness innebærer at hver HTTP-forespørsel (GET,POST,PUT etc) skjer i fullstendig isolasjon.
Når klienten foretar en HTTP-forespørsel, inkluderer den all informasjon som er nødvendig for at serveren skal kunne oppfylle forespørselen.
Serveren trenger ikke å vite noe om klientens status

Statelessness hjelper til med å skalere API-ene til millioner av samtidige brukere ved å distribuere den til flere servere. Enhver server kan håndtere enhver forespørsel fordi det ikke er noen prosess-relatert avhengighet.

Et statsløst API er også enkelt å bruke til mellomlagring data, betegnet som «caching».
Klientprogramvaren kan bestemme om resultatet av en HTTP-forespørsel skal bufres eller ikke, bare ved å se på den ene forespørselen. Det er ingen usikkerhet om at status fra en tidligere forespørsel kan påvirke hurtig-bufringen til siste spørring. Buffring forbedrer ytelsen til applikasjoner som er basert på REST-API ved at samme data ikke hentes flere ganger.

Det finnes ikke altfor spesifikke retningslinjer rundt W3C HTML5-spesifikasjonen, noe som kan føre til forvirring og usikkerhet rundt REST-terminologi. Det vi har Også listen ovenfor bør ikke betraktes som bastante regler, selv om de er sanne for de fleste moderne RESTful API-er.

REST er en lettvekts-metodikk som gjør den perfekt til HTTP-data (trafikk via internett-protokoll). Det er derfor REST ble så populært på nettet, og hvorfor det er allment ansett som det beste valget for API-utvikling. Eksperter mener at REST er et API for utviklerens brukergrensesnitt. Alt skal være enkelt å bruke, og gi en god brukeropplevelse.



Nøkkelfunksjoner for RESTful APIs
Disse funksjonene er spesielt sterke sett i sammenheng med bruk av API-er i webapplikasjoner. Dette betyr at http/HTTPS er et krav, og det betyr ofte at API-dataene er vert på en ekstern server.

RESTful API fungerer som motor for API-brukeren.
API-brukeren er nettutvikleren som kan bygge et skript som kobles til en ekstern API-server, og de nødvendige dataene sendes over http/HTTPS. Utvikleren kan da vise data på deres nettside uten å ha personlig tilgang til den eksterne serveren (som å trekke Twitter-data).

Kilde: https://no.hideout-lastation.com/basics-rest-restful-api-development

Generelt er det fire kommandoer som brukes til å få tilgang til RESTful APIs :

1. GET for å hente en gjenstand
2. POST for å opprette en ny gjenstand
3. PUT for å endre eller erstatte et objekt
4. DELETE for å fjerne en gjenstand

Hver av disse metodene bør sendes med API-anropet for å fortelle serveren hva han skal gjøre. De aller fleste web-APIer tillater bare GET forespørsler for å trekke data ut fra en ekstern server. Autentisering er valgfri, men absolutt en god ide når du tillater potensielt skadelige kommandoer, spesielt PUT (oppdater) eller DELETE (slett).

Men ikke mange RESTful API-er går selv så langt at du kan slette eller oppdatere. Test gjerne Pokéapi som er en gratis Pokémon API-database. Den er åpen for publikum med brukervennlig tilgangsstyring, hvor brukeren har begrenset tilgang til et bestemt antall API-forespørsler over en tidsperiode. Bruker har kun tilgang til GET metoden for uthenting av data.

Returtyper er også viktige, og skal beholde homogenitet for alle ressurser. JSON er en populær returtype med elektroniske spesifikasjoner som forklarer riktig datastrukturer.

RESTful API bruker substantiver for API objekter, og verb for å utføre handlinger på disse objektene. Godkjenning kan være en del av dette, tilgangsstyringen kan også være en del av dette. Men en veldig enkel API kan fungere bra uten store brukerbegrensninger.

Hvordan få tilgang til API-ressurser?
Offentlige APIer er vanligvis tilgjengelige direkte fra nettadresser, også kalt «endpoints». Dette betyr at nettadressestrukturen er viktig, og bør bare brukes til API-forespørsler.
Enkelte nettadresser kan inneholde et prefiks katalog som /v2/ for en oppdatert versjon 2 av en tidligere API. Dette er vanlig for utviklere som ikke vil avskrive sin 1.x API, men vil fortsatt tilby den nyeste strukturen.

Mer om REST: https://restfulapi.net/

SOAP vs REST

Hva er forskjellen mellom SOAP og REST APIer, og hvilken er riktig for mitt prosjekt?

Begrepet web -API refererer generelt til begge sider av datasystemer som kommuniserer over et nettverk: API -tjenestene som tilbys av en server, samt APIet som tilbys av klienten, for eksempel en nettleser. Server-delen av web-API-et er et programmatisk grensesnitt til et definert system for forespørselssvar, og kalles vanligvis Web-tjenesten. Det finnes flere designmodeller for webtjenester, men de to mest dominerende er SOAP og REST.

SOAP gir følgende fordeler sammenlignet med REST:

- Uavhengig av språk, plattform og transport (REST krever bruk av HTTP)
- Fungerer godt i distribuerte bedriftsmiljøer (REST forutsetter direkte punkt-til-punkt-kommunikasjon)
- Standardisert
- Gir betydelig skalering ihht web-service-standardene
- Innebygd feilhåndtering
- Automatisering når den brukes med visse språkprodukter

REST gir følgende fordeler sammenlignet med SOAP:

- For det meste lettere å bruke og er mer fleksibel.
- Enklere å forstå standarder som dokumentasjonen i form av «Swagger» og «OpenAPI Specification»
- Kortere læringskurve
- Effektivt da SOAP bruker omfattende XML for alle meldinger, mens REST bruker for det mindre meldingsformater som «JSON»
- Rask (ingen omfattende behandling nødvendig)
- Nærmere andre webteknologier innen designfilosofi

Som en opplæring i REST API uttrykte det: SOAP er som en konvolutt mens REST bare er et postkort. Et postkort er sikkert raskere og billigere å sende enn en konvolutt, men det kan fortsatt pakkes inn i noe annet, til og med en konvolutt. Du kan også bare lese et postkort, mens en konvolutt tar noen ekstra trinn, for eksempel å åpne eller pakke ut for å få tilgang til det som er inni. Dette er bare «VG-versjonen», men du kan utvide kunnskapen med å lese i kilden og få flere detaljer om de to formatene.

Kilde inkl video: https://www.soapui.org/learn/api/soap-vs-rest-api/