# Web Services

Olav Lysne

# Til nå har dere hørt om...

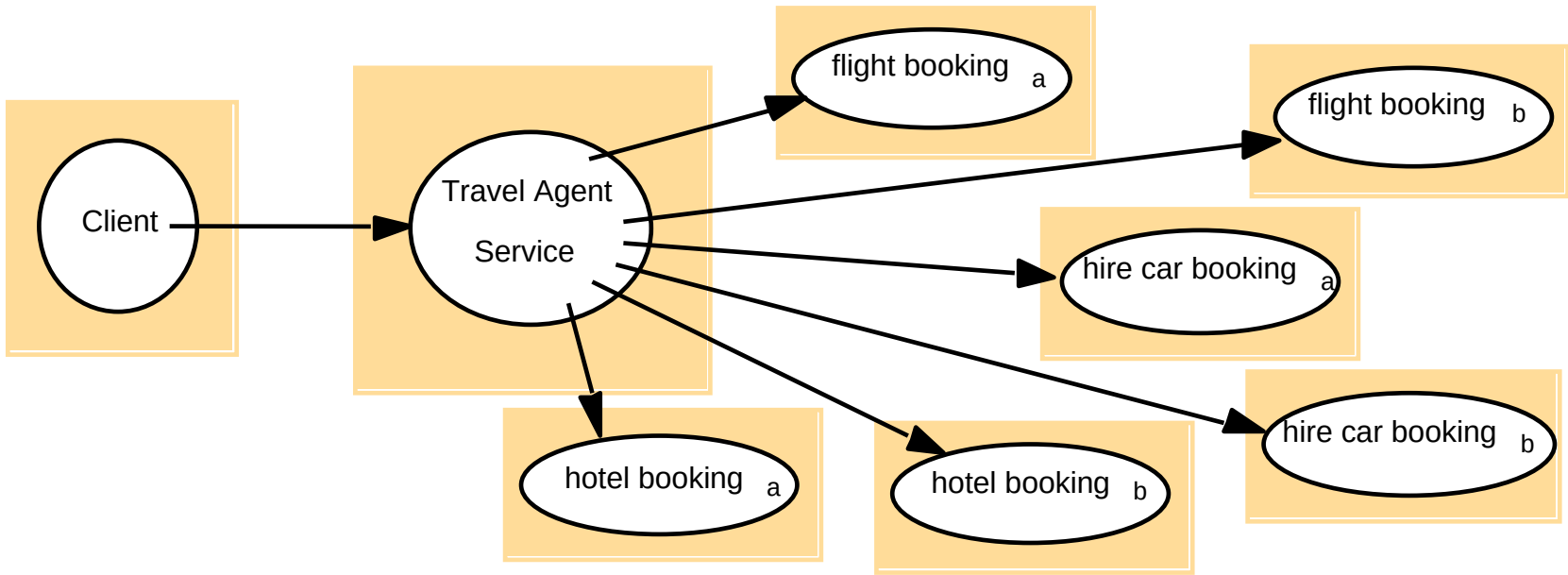
- Mellomvare for objektbasert kommunikasjon brukes vanligvis i anvendelser som er
  - innen én organisasjon, eller
  - innen et tett konsortium av samarbeidende organisasjoner
- Hvilke enheter som deltar er oversiktlig

- Behov for standardisering av grensesnitt for generelle tjenester.
- WWW er en generell tjeneste, men er dårlig egnet for annet enn menneskestyrt interaksjon

# Web services

- Tillater at en applikasjons-spesifikk klient snakker med en tjeneste over et funksjonalitetsspesifikt grensesnitt
- Tillater et klient program i en organisasjon å snakke med et tjenerprogram i en annen
- Åpner for komplekse applikasjoner som integrerer tjenester fra forskjellige andre tjenester
- Generaliteten i interaksjonen gjør at web-services ikke kan aksesseres direkte av browsere.

# Eksempel fra boken



# Brukes dette?

- Mange velkjente websteder tilbyr et Web-services grensesnitt til sine tjenester
- Boken nevner
  - Amazon, Yahoo, Google, eBay...

# Hovedforskjeller fra distribuerte objekter

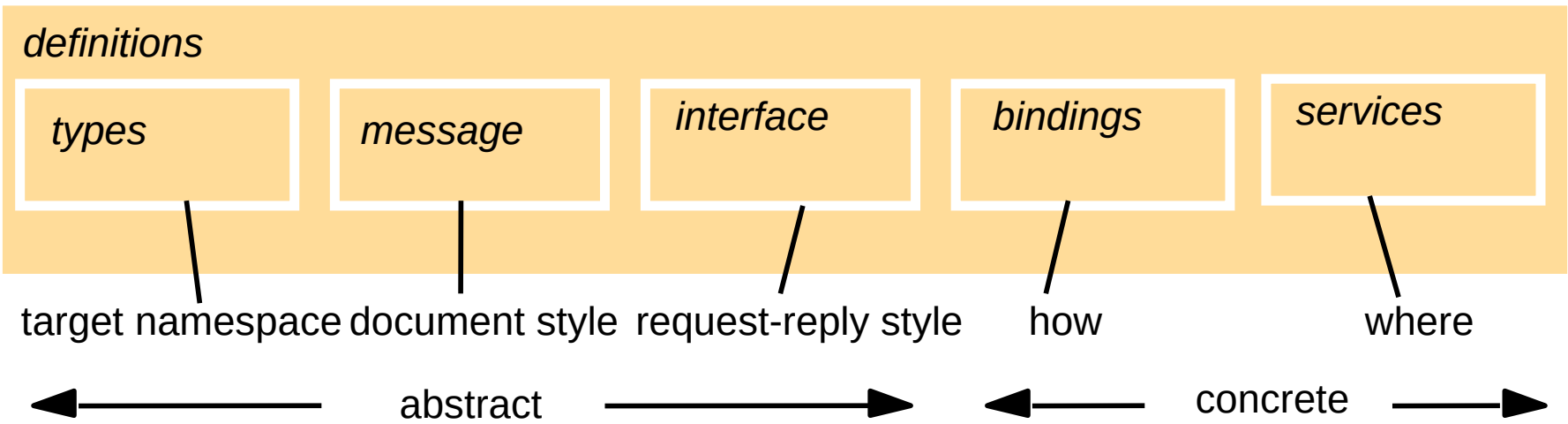
- Mye er likt, men
- Objektbegrepet finnes ikke,
  - Kan ikke instansiere fjerne objekter
  - Søppeltømming irrelevant
  - Remote Object References trengs ikke (men i stedet trengs noe annet...)

# Men hvordan henger WS sammen?

- WSDL
  - Beskrivelsesspråk for en tjeneste
- URI
  - URN og URL
  - Sammen med DNS identifiserer de og gir lokasjon til en tjeneste.
- XML
  - Beskrivelse av meldingsinnhold
- SOAP (alternativt REST)
  - Enkelt innkapslings-rammeverk for meldinger i XML
- HTTP
  - ”Transportprotokollformat” – (går over TCP)

# WSDL beskrivelse

- Beskrivelse av en tjeneste



# WSDL

- Types
  - Navn og Typer på verdiene som blir utvekslet (analogt til varaibeldeklarasjoner).
- Messages  (og operasjoner)
  - Identifiserer typen og formatet av meldinger som skal utveksles, og hvilke effekter de skal ha.
- Interface
  - Samling av operasjoner
    - In-Out, In-Only, Out-In, Out-only...
- Binding
  - Sier noe om formater på meldingene.
    - vanligvis SOAP, HTTP og MIME, men kan også være annet (GIOP for å snakke med Corba f. eks).
- Services
  - Sier HVOR tjenesten befinner seg
  - URL for oppslag via DNS, eller URN for bruk sammen med en navnetjeneste

# URI – Uniform Resource Identifier

- Kan være en URL
  - Oppslag via DNS
  - Delvis lokasjonsavhengig (I alle fall avhengige av domenenavnet på maskinen).
- URN (Uniform Resource Names) er en annen mulighet som er lokasjonsuavhengig.
  - Denne er avhengig av en Directory Service – ”Universal Directory and Discovery Service UDDI”

# XML – Extensible Markup Language

- Et språk for beskrivelse av meldingsformater, og hvordan de skal parseres
- UNICODE basert
  - Lesbart for både mennesker og maskiner
  - Ineffektivt – plasskrevende
- Et språk som er egnet til å representere en hierarkisk datastruktur på en flat UNICODE-basert form.

# XML definition eksempel

```
<person id="123456789">
        <name>Smith</name>
        <place>London</place>
        <year>1934</year>
        <!-- a comment -->
</person >
```

# Namespace i *Person* strukturen

```
<person pers:id="123456789"
                    xmlns:pers = "http://www.cdk4.net/person">
    <pers:name> Smith </pers:name>
    <pers:place> London </pers:place >
    <pers:year> 1934 </pers:year>
</person>
```

# Skjema for Person-strukturen

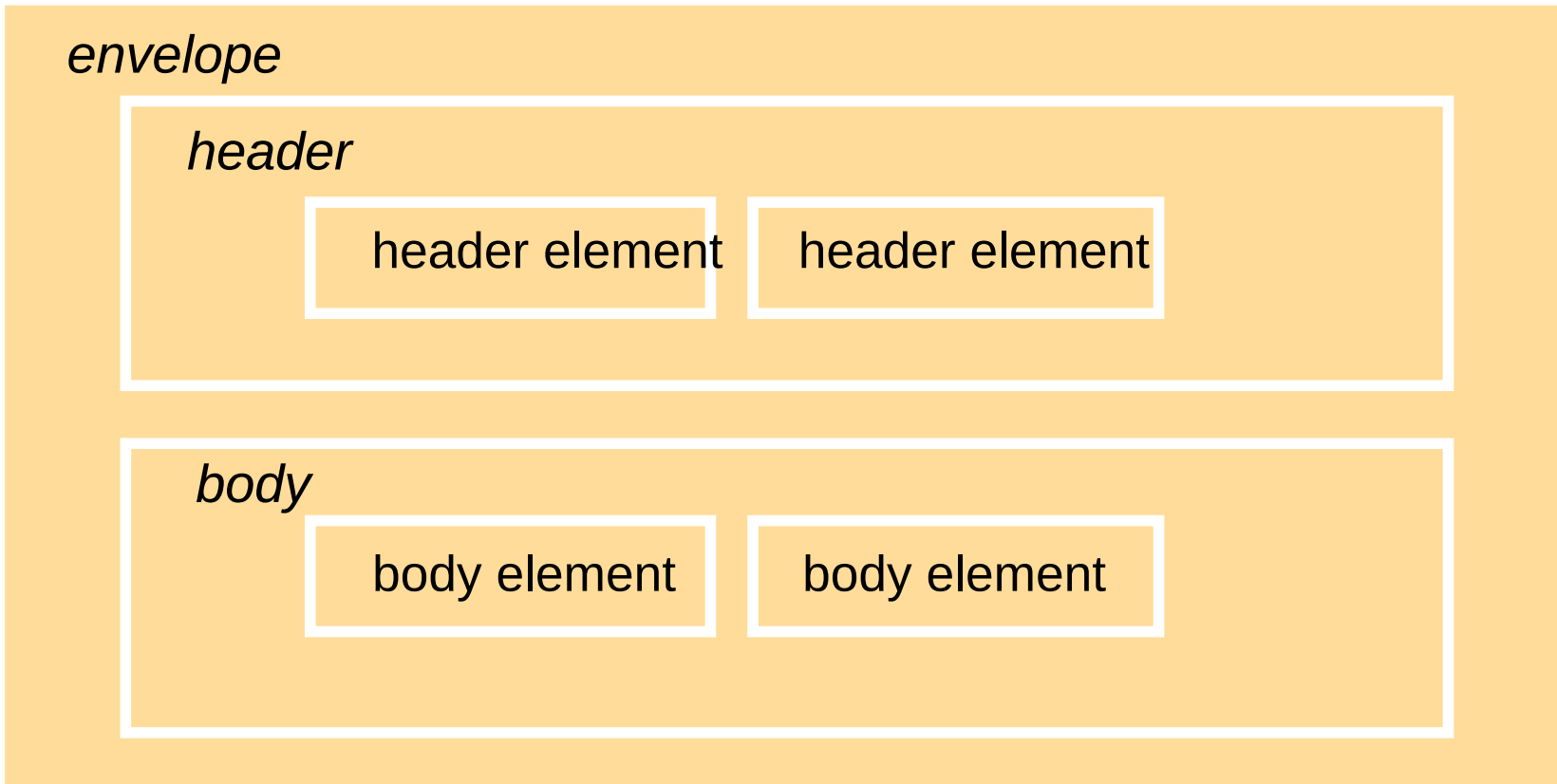
```
<xsd:schema  xmlns:xsd = URL of XML schema definitions  >
    <xsd:element name= "person" type ="personType" />
        <xsd:complexType name="personType">
            <xsd:sequence>
                <xsd:element name = "name"  type="xs:string"/>
                <xsd:element name = "place"  type="xs:string"/>
                <xsd:element name = "year"  type="xs:positiveInteger"/>
            </xsd:sequence>
            <xsd:attribute name= "id"   type = "xs:positiveInteger"/>
        </xsd:complexType>
</xsd:schema>
```

# SOAP

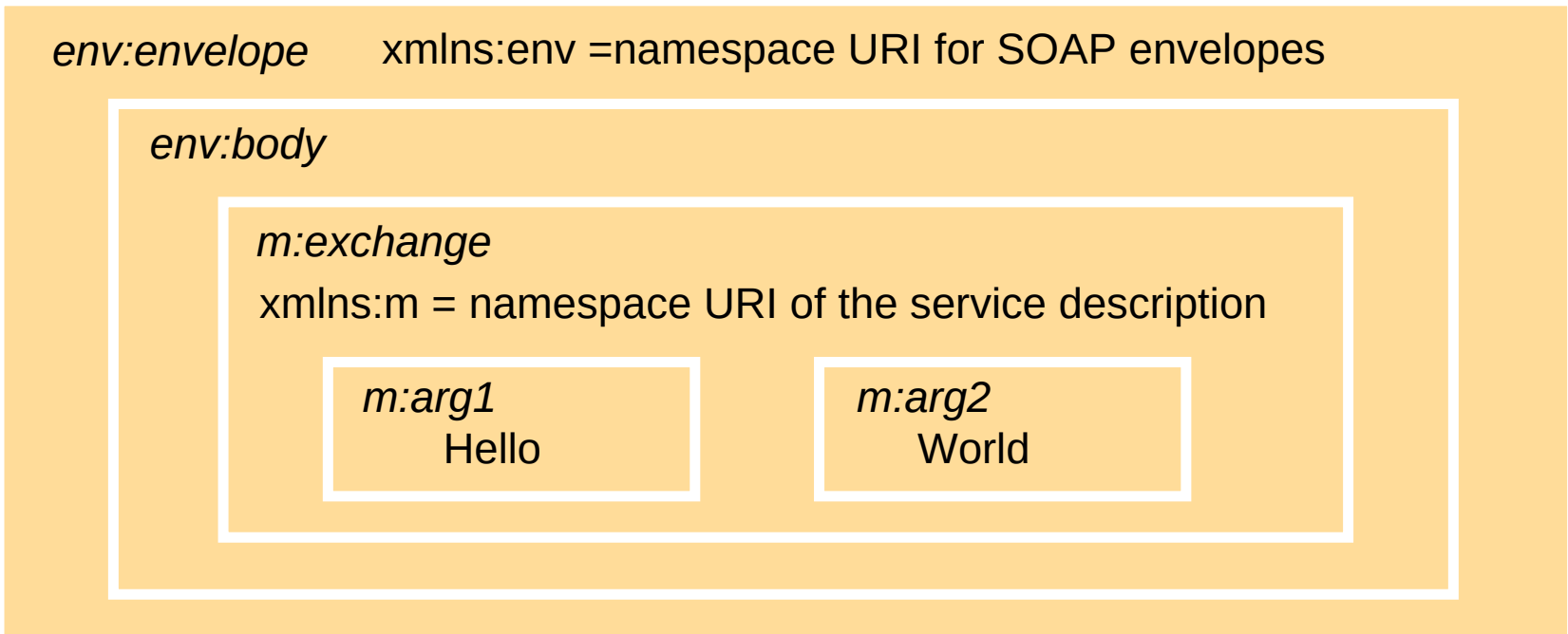
- Definerer hvordan XML skal brukes til å representere innholdet i meldinger.
- Definerer hvordan et par av meldinger kan kombineres til et request/reply mønster
- Reglene for hvorledes mottakere av meldinger skal prosesere de XML-elementene de inneholder
- Hvordan HTTP (eller SMTP) skal brukes til å kommunisere SOAP meldinger.

# En SOAP melding

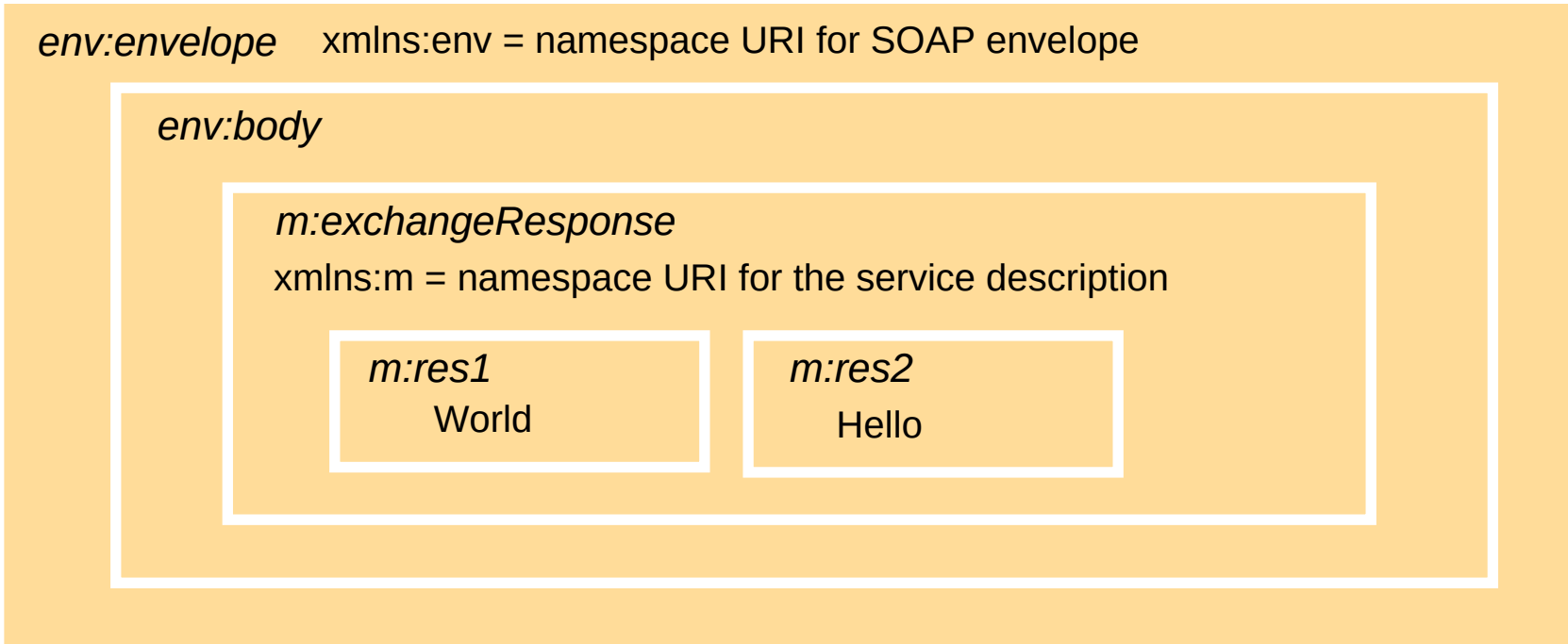
Envelope, Header og Body



# SOAP-melding uten header



# SOAP-response



# Eksempel på SOAP melding inne i HTTP

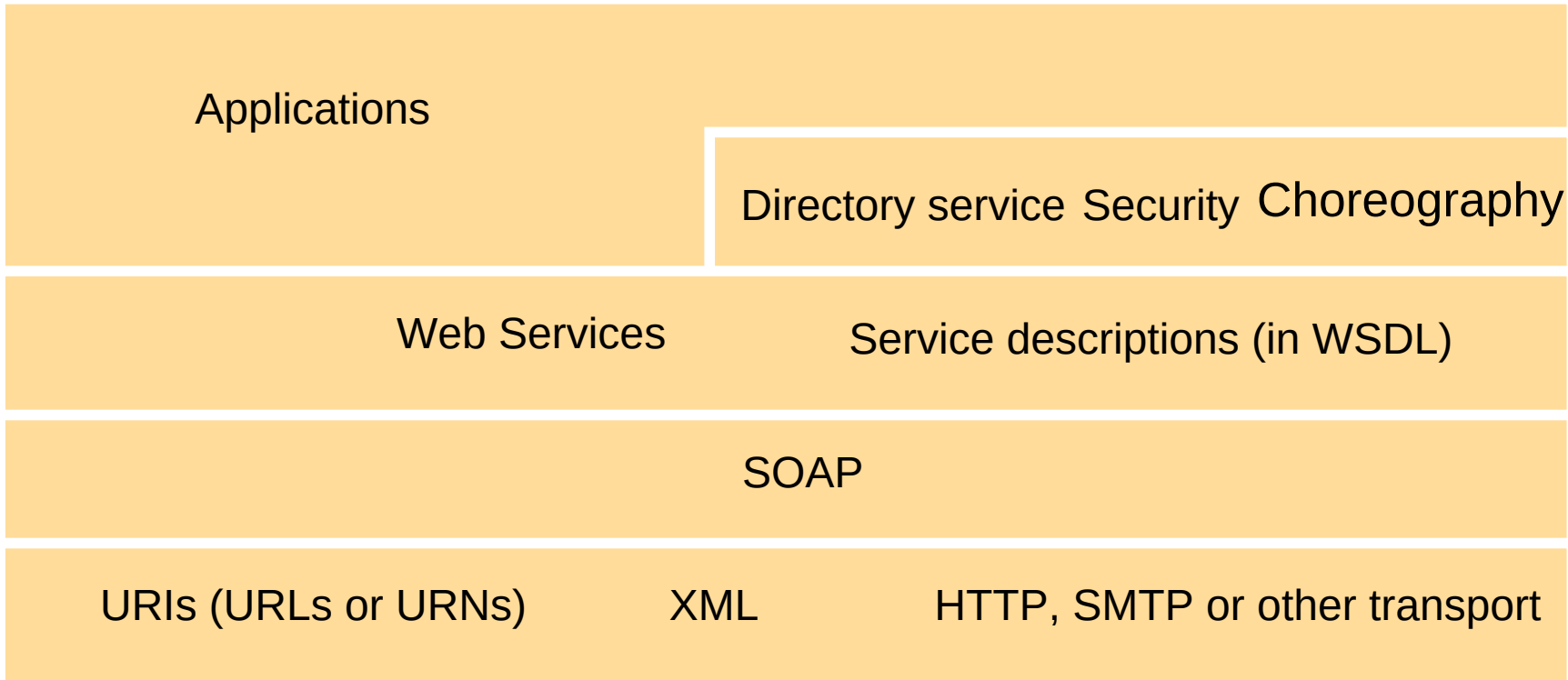
*POST /examples/stringer* ← endpoint address
*Host: www.cdk4.net*
*Content-Type: application/soap+xml*
*Action: http://www.cdk4.net/examples/stringer#exchange* ← action

HTTP header

*<env:envelope xmlns:env=* namespace URI for SOAP envelope *>*
*<env:header> </env:header>*
*<env:body> </env:body>*
*</env:Envelope>*

Soap message

# Infrastruktur og komponenter - oppsummering



# Web-services hovedpunkter

- Baserer navningen sin på DNS, som er skalerbar og global
- Referansebegrepet URI er enklere enn ROR er for fjerne objekter
- Enkelt å bruke
  - Baseres bare på HTTP og XML infrastrukturer som allerede er på plass i de fleste eksisterende operativsystemer
  - Meldingene er lesbare for mennesker
  - Trenger bare et praktisk API for SOAP
- Ikke veldig effektivt
  - Lange meldinger som er tunge å parse.
  - Opp mot 1000 ganger tregere enn CORBA er rapportert