**Præsentation**

* Traditionelle data loggere – geostationære eller off-line.
* Ofte er hvor er målingen taget ligeså vigtigt som hvad målingen er.
* Samlet overvågning vs. enkeltstående loggere
* Forklar hvordan data indsamles.

**Device repræsentation**

* Et device er en ressource.
* SOAP vs. REST (Ved at kortlægge om ens system kan udtrykkes i form af resourcer kan man bestemme hvilken).
* ID, placering, ejer, målinger, status, location (zone), dato/tid, – hele XML.

**Devices præsentation**

* To brugerflader: service og web

Service:

* XML er valgt som interface (passer med web service, zip kan anvendes til pakning). Proprietær format vs. XML – selv små devices kan klare en web service/client. XML -> mere typefrihed. Definere schema for type i XML.
* geolog.xsd + index.html beskriver API (mere løs en WSDL)
* Push vs pull. Devices skal melde ind på fast interval ellers mangler status disconnected, alternativ pull (bedre load balancing (kontrol med hvornår de sender – kombination, event plus pull).
* PUT vs. POST
* HTTP if-modified-since -> 304 not modified

Web-site:

* Website er bare en custom client (giver rent snit)
* I gamle dage – 1 device, 1 hjemmeside – manuel opdatering.
* Nu: device liste og placering i google maps (nuværende for dynamiske). Grafisk repræsentation af målinger. Historik over placeringer.
* XML gør det nemt, Javascript gør det hurtigt, AJAX gør det opdaterbart.
* XSLT fordele/ulemper (kompleksitet, dateTime)
* HTTPS / HTTP authentication / none. Også device -> service, men hvad er risken og hvor følsom er data. Ved device -> service skal der DNS spoofing til. AES key i config, og kun config via TLS.
* Stylesheet (et sprog for sig)
* Tomcat (pre/post processing) – generere det hele på runtime med templates (JSP).
* HTTP if-modified-since -> 304 not modified + andre load reduction ideas.
* Caching af svar I hukommelsen, så hvis service returnerer 304 kan vi med det same returnerer siden.

**Teknologier:**

* REST
  + Godt til ressourcer, WSDL til API ikke afprøvet – skulle være muligt i ny version (2.0) - XMLSchema
* XML
  + God ide. Godt at arbejde med. Læsbart. Kendt standard. Zip til mindre comm. Binary eller JSON mindre.
  + XML database kunne bruges, og er faktisk det vi gør.
* XML Schema
  + Nem til API definition. dateTime skal komme i rækkefølge kan ikke udtrykkes.
* XSLT
  + Godt til XML -> XHTML, mindre godt til JavaScript og komplekse beregninger.
* HTML/XHTML
  + Kendt og uundgåeligt i websites.
* CSS
  + Godt til grafisk layout: farve, fint, … - sikre konsistens
* JavaScript
  + Imponerende og hurtigt. Ikke så egnet til XSLT, da det ikke er XML.
* AJAX
  + Godt og nemt med JQuery (cross browser)
* JQuery
  + Gør Javascript og AJAX lidt nemmere
* JQGraph
  + Første eksperiment med grafik i JavaScript. Virker OK, men vil kræve lidt mere analyse.
* Java
  + Kendt programmeringssprog. Stor frihed i muligheder i forhold til XSLT.
* Java Servlet
  + Java’s frihed. Man skal generere HTML/XHTML også det statiske.
* JSP
  + Bedre kobling mellem HTML og java, men grimt ved høj kompleksitet.
* Tomcat
  + Nem og hurtig til servlet. Alternativt findes plugin til Apache
* JDOM
  + Godt API til at arbejde med XML dokumenter. XPath (saxon) godt til lookup, hvis det virkede.

<device id="device1" xmlns="http://www.pa.com/geolog" xmlns:k="http://www.opengis.net/kml/2.2">

<geologCollection>

<geolog dateTime="2001-10-26T21:32:52.1267+02:00">

<readings>

<reading id="sensor1">

<key>altitude</key>

<value>125.0001</value>

<type>xs:double</type>

<unit>m</unit>

</reading>

<reading id="sensor2">

<key>light</key>

<value>25</value>

<type>xs:int32</type>

<unit>lm</unit>

</reading>

</readings>

<status>OK</status>

<k:Point>

<k:coordinates>-90.969,48.2545</k:coordinates>

</k:Point>

</geolog>

</geologCollection>

</device>