Welcome to

3. SOAP, WSDL, XML data

KEA System Integration F2020 10 ECTS

Henrik Kramselund Jereminsen hkj@zencurity.com @kramse

Slides are available as PDF, kramse@Github

3-SOAP-WSDL-XML-system-integration.tex in the repo security-courses

Plan for today

- Data formats overview
- XML data, JSON
- Data Transformation
- WebServices
- SOAP, WSDL
- Investigate examples

Exercises

- Chapter 3 from the book
- Looking at Web Services, discuss web services

Reading Summary

Camel chapter 3: Transforming data with Camel

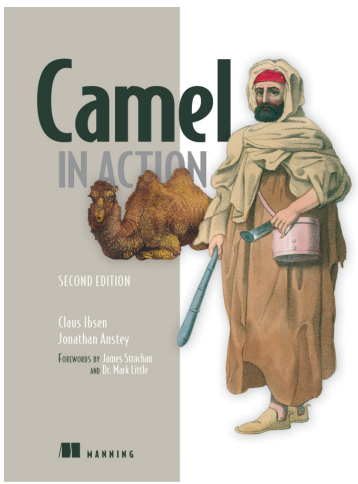
Contains a lot of different formats

We will use Wikipedia a lot today also

Todays Agenda - approximate time plan

- 45m Data formats overview XML and JSON, xsltproc mini exercise
- 45m Small Python exercises, try using Python with XML and JSON
- 10:00 Break 15m
- 45m Chapter 3 presented and discussed including the book examples
- 45m Chapter 3 presented and discussed including the book examples
- 11:45 Lunch Break
- 45m Chapter 3 presented and discussed including the book examples
- 45m WebServices, SOAP, WSDL
- 14:00 Break 15m
- 45m Investigate examples from the internet

Review Still got the Debian VM running



Running new tools and trying out examples like in the book are important.

You all did good, and helped each other!

The tasks performed can be automated using something called Ansible, but more on that later.

Chapter 1: file-copy example

```
hlk@debian-lab:~/file-copy$ find data/
data/
data/outbox
data/outbox/message1.xml
data/inbox
data/inbox/message1.xml
```

- We want to run the command for Maven to download tools, and *do stuff*
- `mvn compile exec:java`
- This might take some time!
- Note: this is a two step process, so split into `mvn compile` and `exec:java` if you have trouble running

Success Compile

```
hlk@debian-lab:~/file-copy$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------< com.camelinaction:chapter1-file-copy >----------------
[INFO] Building Camel in Action 2 :: Chapter 1 :: File Copy Example 2.0.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.4.3:resources (default-resources) @ chapter1-file-copy ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.6.1:compile (default-compile) @ chapter1-file-copy ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1.270 s
[INFO] Finished at: 2020-02-17T07:08:02+01:00
[INFO] ------------------------------------------------------------------------
```

Success Execute Java - shortened for slide

```
hlk@debian-lab:~/file-copy$ mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------< com.camelinaction:chapter1-file-copy >----------------
[INFO] Building Camel in Action 2 :: Chapter 1 :: File Copy Example 2.0.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.2.1:java (default-cli) @ chapter1-file-copy ---
[ion.FileCopierWithCamel.main()] DefaultCamelContext  INFO  Apache Camel 2.24.3 (CamelContext: camel-1) is starting
[ion.FileCopierWithCamel.main()] FileEndpoint         INFO  Using default memory based idempotent repository with cache max size: 10
[ion.FileCopierWithCamel.main()] DefaultCamelContext  INFO  Route: route1 started and consuming from: file://data/inbox?noop=true
[ion.FileCopierWithCamel.main()] DefaultCamelContext  INFO  Total 1 routes, of which 1 are started
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 11.908 s
[INFO] Finished at: 2020-02-17T07:11:18+01:00
[INFO] ------------------------------------------------------------------------
```

Success Execute Java - new files

```
$ find data/
data/
data/outbox
data/outbox/message1.xml
data/outbox/message2.txt
data/inbox
data/inbox/message1.xml
data/inbox/message2.txt
```

- Try adding a new file using editor, and re-run
  `echo "some data" > data/inbox/message2.txt`

Data overview XML data, JSON

Photo by Chris Lawton on Unsplash

XML data

> Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The World Wide Web Consortium's XML 1.0 Specification[2] of 1998[3] and several other related specifications[4]—all of them free open standards—define XML.[5]
>
> The design goals of XML emphasize simplicity, generality, and usability across the Internet.[6] It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures[7] such as those used in web services.

Source: `https://en.wikipedia.org/wiki/XML`

- We have seen XML used for configuration in Apache Tomcat and Camel
- Perfect for computers, less for humans

XML data example - Nmap output

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xml-stylesheet href="file:///usr/bin/../share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 7.70 scan initiated Sat Feb 22 23:35:53 2020 as: nmap -oA router -sP 10.0.42.1 -->
<nmaprun scanner="nmap" args="nmap -oA router -sP 10.0.42.1" start="1582410953"
 startstr="Sat Feb 22 23:35:53 2020" version="7.70" xmloutputversion="1.04">
<verbose level="0"/>
<debugging level="0"/>
<host><status state="up" reason="echo-reply" reason_ttl="62"/>
<address addr="10.0.42.1" addrtype="ipv4"/>
<hostnames>
</hostnames>
<times srtt="2235" rttvar="5000" to="100000"/>
</host>
<runstats><finished time="1582410953" timestr="Sat Feb 22 23:35:53 2020" elapsed="0.32"
 summary="Nmap done at Sat Feb 22 23:35:53 2020; 1 IP address (1 host up)
 scanned in 0.32 seconds" exit="success"/><hosts up="1" down="0" total="1"/>
</runstats>
</nmaprun>
```

XML data - documents

> Hundreds of document formats using XML syntax have been developed,[8] including RSS, Atom, SOAP, SVG, and XHTML. XML-based formats have become the default for many office-productivity tools, including Microsoft Office (Office Open XML), OpenOffice.org and LibreOffice (OpenDocument), and Apple's iWork[citation needed]. XML has also provided the base language for communication protocols such as XMPP. Applications for the Microsoft .NET Framework use XML files for configuration, and property lists are an implementation of configuration storage built on XML.[9]

Source: `https://en.wikipedia.org/wiki/XML`

- Document formats using XML may still be proprietary!
- Documents using XML can be validated, are they well-formed according to the Document Type Definition (DTD)

XML data - standards

> Many industry data standards, such as Health Level 7, OpenTravel Alliance, FpML, MISMO, and National Information Exchange Model are based on XML and the rich features of the XML schema specification. Many of these standards are quite complex and it is not uncommon for a specification to comprise several thousand pages.[citation needed] In publishing, Darwin Information Typing Architecture is an XML industry data standard. XML is used extensively to underpin various publishing formats.

Source: `https://en.wikipedia.org/wiki/XML`

- Dont worry too much about standards at this time
- The important standards will often be defined by the business area

XML data for Service-oriented architecture (SOA)

> XML is widely used in a Service-oriented architecture (SOA). Disparate systems communicate with each other by exchanging XML messages. The message exchange format is standardised as an XML schema (XSD). This is also referred to as the canonical schema. XML has come into common use for the interchange of data over the Internet. IETF RFC:3023, now superseded by RFC:7303, gave rules for the construction of Internet Media Types for use when sending XML. It also defines the media types application/xml and text/xml, which say only that the data is in XML, and nothing about its semantics.

Source: `https://en.wikipedia.org/wiki/XML`

- We will talk more about SOA later

Transforming XML using XSLT

> XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents,[1] or other formats such as HTML for web pages, plain text or XSL Formatting Objects, which may subsequently be converted to other formats, such as PDF, PostScript and PNG.[2] XSLT 1.0 is widely supported in modern web browsers.[3]

Source: `https://en.wikipedia.org/wiki/XSLT`

- Can be seen as a mapping between formats, different XML schemas
- Also is Turing complete, is a programming language

XSLT example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/persons">
    <root> <xsl:apply-templates select="person"/> </root>
  </xsl:template>
  <xsl:template match="person">
    <name username="{@username}"> <xsl:value-of select="name" /> </name>
  </xsl:template>
</xsl:stylesheet>
```

- XSLT uses XPath to identify subsets of the source document tree and perform calculations. XPath also provides a range of functions
- XSLT functionalities overlap with those of XQuery, which was initially conceived as a query language for large collections of XML documents
  Source: `https://en.wikipedia.org/wiki/XSLT`

xsltproc example using Nmap

```
$ su -
# apt install nmap xsltproc
# nmap -sP -oA /tmp/router 91.102.91.18
# exit
$ xsltproc /tmp/router.xml > /tmp/router.html
$ firefox /tmp/router.html
```

- We can use the command line tool `xlstproc` for transforming documents
- `apt install xsltproc`
- Its part of the package Libxslt `https://en.wikipedia.org/wiki/Libxslt`

- Try installing the tools Nmap and `xsltproc` and reproduce the above
- This is an easy tool to try, and very useful too

Data overview JSON

> JavaScript Object Notation (JSON, pronounced / d e sən/; also / d e s n/[note 1]) is an open-standard file format or data interchange format that uses **human-readable text** to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as replacement for XML in AJAX systems.[6]

Source: `https://en.wikipedia.org/wiki/JSON`

- I like JSON much better than XML
- Many web services can supply data in JSON format

JSON example

```
{
  "first name": "John",
  "last name": "Smith",
  "age": 25,
  "address": {
    "street address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal code": "10021"
  },
  "phone numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
  ],
}
```

- This is a basic JSON document, new data attribute-value pairs can be added
  Source: `https://en.wikipedia.org/wiki/JSON`

Note about frameworks and libraries

```
import xml.etree.ElementTree as ET
tree = ET.parse('testfile.xml')
root = tree.getroot()

print(root.tag)
print('Nmap version: \t\t{:s} '.format(root.attrib['version']))
print('Nmap started: \t\t{:s} '.format(root.attrib['startstr']))
print('Nmap command line: \t{:s} '.format(root.attrib['args']))

hosts = tree.findall('./host')
for host in hosts:
    print(host.tag)
    print(host.attrib)
    for hostvalues in host:
        print(hostvalues.tag)
        print(hostvalues.attrib)
```

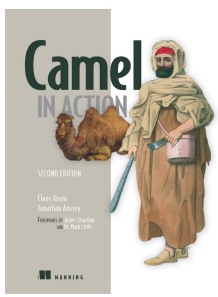
- Dont import JSON or XML using home made programs
- Example uses xml.etree.ElementTree from Python `https://docs.python.org/3.7/library/xml.etree.elementtree.html`

Convert XML to JSON

```
import xml.etree.ElementTree as ET
import json
def etree_to_dict(t):
    d = {t.tag : map(etree_to_dict, t.getchildren())}
    d.update(('@' + k, v) for k, v in t.attrib.items())
    d['text'] = t.text
    return d

tree = ET.parse('testfile.xml')
root = tree.getroot()
mydict = etree_to_dict(root)
print(type(tree))
print(type(root))
print(type(mydict))

print(mydict)

with open('testfile.json', 'w') as json_file:
  json.dump(mydict, json_file)
```

Converting using Python is easy

Chapter 3: Transforming data with Camel



This chapter covers

- Transforming data by using EIPs and Java
- Transforming XML data
- Transforming by using well-known data formats
- Writing your own data formats for transformations
- Understanding the Camel type-converter mechanism

Dependencies

Some exercises uses deprecated APIs not in JDK 11,

```
Unresolvable class definition; nested exception is
java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException
```

- to avoid problems it is possible to add dependencies to POM, JAXB
  see the JDK 9+ pom.xml `https://github.com/apache/camel/blob/master/parent/pom.xml#L5741`
- This is a common problem, version mismatch - but also what Maven supports
- Using an older JDK 8 would also initially help this, but then what?!

Discussion: Legacy code

Options:

- We have legacy code, Camel book sources
- Can run under JDK 8
- We can create a virtual machine using old JDK running old code

- We can upgrade the code, what does that include?
- Using the same libraries, are they supported, known bugs
- Upgrade/change into new APIs - how much work?

Forked and testing

I decided that I wanted to change the code ☺

I have forked the original repository:
`https://github.com/kramse/camelinaction2`

YMMV - and no guarantees

Main changes so far:

- Parent top-level pom.xml updated version numbers for
- Parent top-level pom.xml added JAXB stuff, for making chapter 3 example working
- `https://github.com/kramse/camelinaction2/commit/35e840110af175c2de8b1093c9b48b673b4921cc`

3.1 Data transformation overview

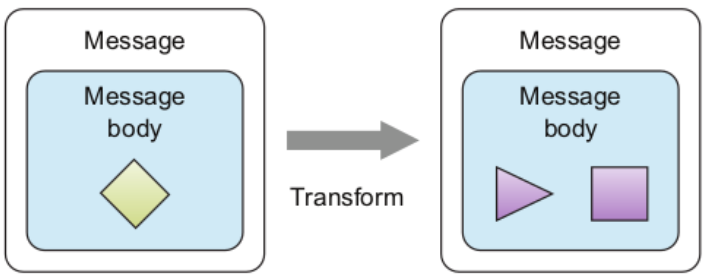


Figure 3.1 Camel offers many features for transforming data from one form to another.

- Data format transformation – The data format of the message body is transformed from one form to another. For example, a CSV record is formatted as XML.
- Data type transformation – The data type of the message body is transformed from one type to another. For example, java.lang.String is transformed into javax. jms.TextMessage .

Six ways data transformation typically takes place in Camel

- Data transformation using EIPs and Java
- Data transformation using components
- Data transformation using data formats
- Data transformation using templates
- Data type transformation using Camel's type-converter mechanism
- Message transformation in component adapters

3.2 Message Translator EIP

Using the Message Translator EIP

The Message Translator EIP is illustrated in figure 3.2.

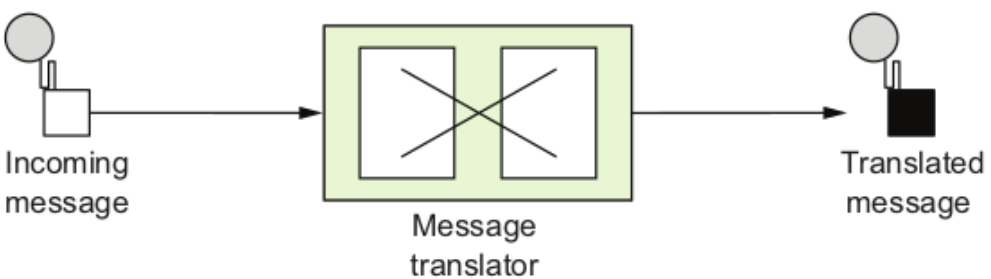


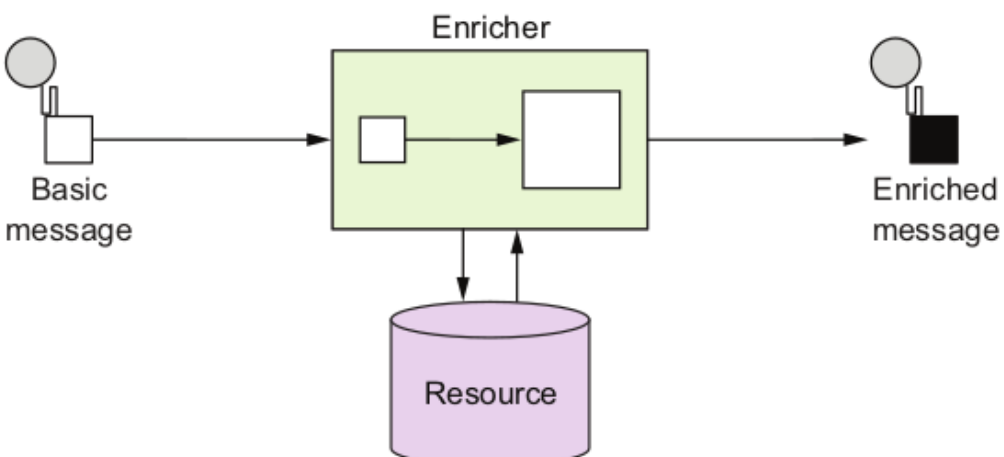
Figure 3.2 In the Message Translator EIP, an incoming message goes through a translator and comes out as a translated message.

- 3.2 Transforming data by using EIPs and Java
- Data mapping, the process of mapping between two distinct data models, is a key factor in data integration.
- Picture shown is using the Message Translator EIP

3.2 Content Enricher EIP

Using the Content Enricher EIP

The Content Enricher EIP is illustrated in figure 3.3. This pattern documents the scenario in which a message is enriched with data obtained from another resource.



- Using the Content Enricher EIP

3.3 Transforming XML

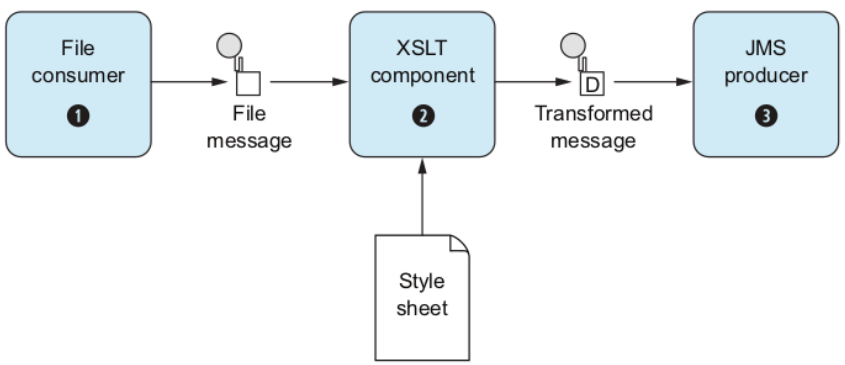


Figure 3.5 A Camel route using an XSLT component to transform an XML document before it's sent to a JMS queue

- Transforming XML with XSLT
- Transforming XML with object marshaling

3.4 Transforming with data formats

- Data formats provided with Camel
- Using Camel's CSV data format
- Using Camel's Bindy data format
- Using Camel's JSON data format
- Configuring Camel data formats

Camel CSV

```
from("file://rider/csvfiles")
  .unmarshal().csv()
  .split(body()).to("jms:queue:csv.record");
```

- Camel has built-in support for many formats
- CSV is a very basic method of moving data

Camel's JSON data format

Listing 3.9 An HTTP service that returns order summaries rendered in JSON format

```
<bean id="orderService" class="camelinaction.OrderServiceBean"/>
<camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
  <dataFormats>
    <json id="json" library="Jackson"/>
  </dataFormats>

  <route>
    <from uri="jetty://http://0.0.0.0:8080/order"/>
    <bean ref="orderService" method="lookup"/>
    <marshal ref="json"/>
  </route>
</camelContext>
```

- Jetty is the small Web server and javax.servlet container `https://www.eclipse.org/jetty/`

Configuring Camel data formats

```
public void configure() {
  CsvDataFormat myCsv = new CsvDataFormat()
      .setDelimiter(';')
      .setHeader(new String[]{
        "id", "customerId", "date", "item", "amount", "description"});
  from("direct:toCsv")
      .marshal(myCsv)
      .to("file://acme/outbox/csv");
}
```

- CSV format can change over time, but often programs are hard to maintain
- Also choosing the seperator needs thought

Side note: Zeek Security Monitor handles formats differently

Zeek has files formatted with a header:

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   trans_id
        rtt     query   qclass  qclass_name     qtype   qtype_name      rcode   rcode_name      AA
        TC      RD      RA      Z       answers TTLs    rejected

1538982372.416180 CD12Dc1SpQm42QW4G3 10.xxx.0.145 57476 10.x.y.141 53 udp 20383
0.045021 www.dr.dk 1 C_INTERNET 1 A 0 NOERROR F F T T 0
   www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93 60.000000,20409.000000,20.000000 F
```

Note: this show ALL the fields captured and dissected by Zeek, there is a nice utility program bro-cut which can select specific fields:

```
root@NMS-VM:/var/spool/bro/bro# cat dns.log | bro-cut -d ts query answers | grep dr.dk
2018-10-08T09:06:12+0200 www.dr.dk www.dr.dk-v1.edgekey.net,e16198.b.akamaiedge.net,2.17.212.93
```

3.5 Transforming with templates

Using Velocity in a Camel route is as simple as this:

```
from("direct:sendMail")
  .setHeader("Subject", constant("Thanks for ordering"))
  .setHeader("From", constant("donotreply@riders.com"))
  .to("velocity://rider/mail.vm")
  .to("smtp://mail.riders.com?user=camel&password=secret");
```

- Using Apache Velocity is a way to build text using template languages

Side note: Jinja templates

Using Velocity in a Camel route is as simple as this:

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

- An often used other library is Jinja `https://jinja.palletsprojects.com/`

3.6: Understanding Camel type converters

- How the Camel type-converter mechanism works
- Using Camel type converters
- Writing your own type converter

Camel type-converter system

How the Camel type-converter mechanism works

To understand the type-converter system, you first need to know what a type converter in Camel is. Figure 3.7 illustrates the relationship between `TypeConverterRegistry` and the `TypeConverters` it holds.

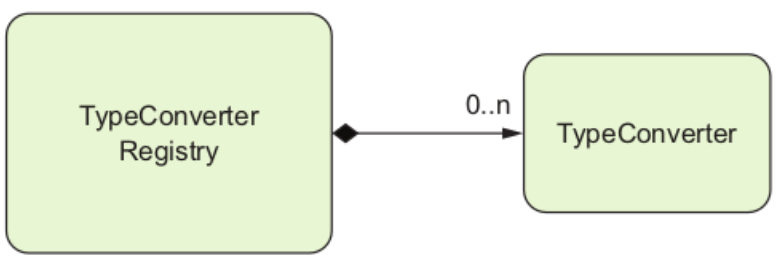


Figure 3.7 The `TypeConverterRegistry` contains many `TypeConverters`

- How the Camel type-converter mechanism works

Example Type Conversion

```
from("file://riders/inbox"){\bf
  .convertBodyTo(String.class)}
  .to("jms:queue:inbox");
```

3.7 Summary and best practices

From the book, and my experience, Here are a few key tips you should take away from this chapter:

- Data transformation is often required—Integrating IT systems often requires you to use different data formats when exchanging data. Camel can act as the mediator and has strong support for transforming data in any way possible. Use the various features in Camel to aid with your transformation needs.
- Java is powerful—Using Java code isn't a worse solution than using a fancy map- ping tool. Don't underestimate the power of the Java language. Even if it takes 50 lines of grunt boilerplate code to get the job done, you have a solution that can easily be maintained by fellow engineers.
- This chapter, along with chapter 2, covered two crucial features of integration kits: routing and transformation.

- Use JSON for humans, Only use CSV as a last resort
- Test and keep up to date - dont rely on old APIs and functionality that will go away
  Example Python2 => Python3, LaTeX old packages and ways, upgrade!

Web services SOAP, WSDL

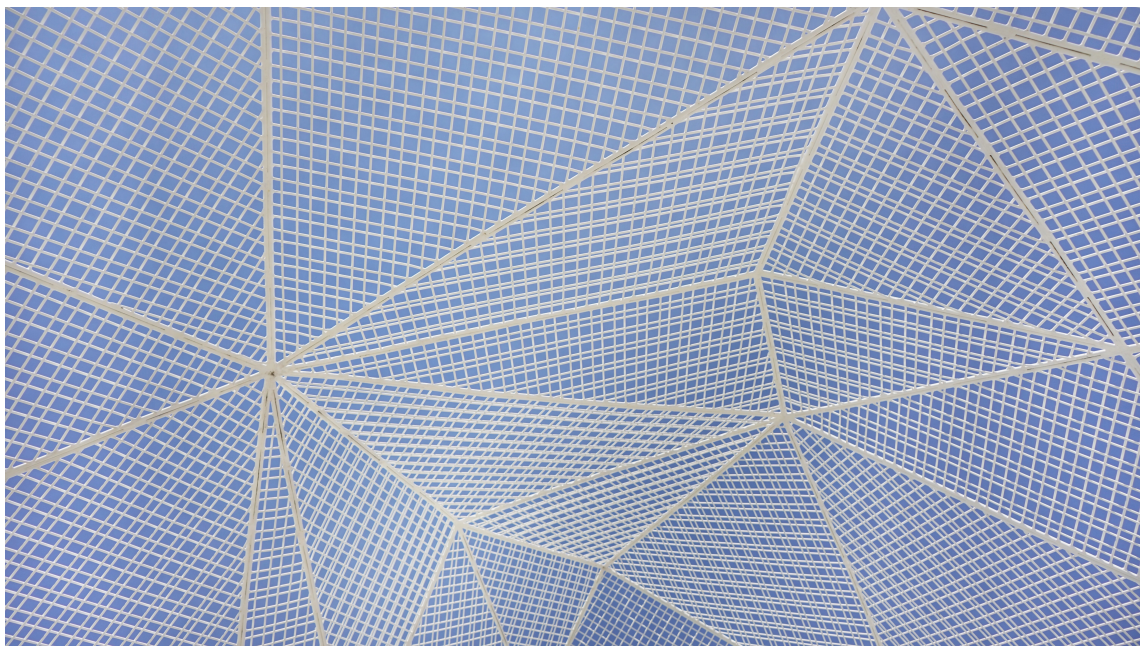


Photo by Josefina Di Battista on Unsplash

Web Services

The term Web service (WS) is either:

- a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web, or
- a server running on a computer device, listening for requests at a particular port over a network, serving web documents (HTML, JSON, XML, images), and creating web applications services, which serve in solving specific domain problems over the Web (WWW, Internet, HTTP)

Source: `https://en.wikipedia.org/wiki/Web_service`

- Today a generic name for services using the internet
- Web servers such as Apache HTTPD, Nginx etc. provide a service to thew internet allowing access using HTTP
- Source for some parts on this slide, `https://en.wikipedia.org/wiki/Web_service`

W3C Web Services

> A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.

Source – W3C, Web Services Glossary[3]

WSDL - Web Services Description Language

The Web Services Description Language (WSDL / w z dəl/) is an XML-based interface description language that is used for describing the functionality offered by a web service. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. Therefore, its purpose is roughly similar to that of a type signature in a programming language.

The current version of WSDL is WSDL 2.0. The meaning of the acronym has changed from version 1.1 where the "D"stood for "Definition".

Source: `https://en.wikipedia.org/wiki/Web_Services_Description_Language`

WSDL XML

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl"
             xmlns:tns="http://www.tmsws.com/wsdl20sample"
             xmlns:whttp="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:wsoap="http://schemas.xmlsoap.org/wsdl/soap/"
             targetNamespace="http://www.tmsws.com/wsdl20sample">

<documentation>
    This is a sample WSDL 2.0 document.
</documentation>
```

Source: https://en.wikipedia.org/wiki/Web_Services_Description_Language

WSDL XML types

```
<!-- Abstract type -->
   <types>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://www.tmsws.com/wsdl20sample"
                 targetNamespace="http://www.example.com/wsdl20sample">

         <xs:element name="request"> ... </xs:element>
         <xs:element name="response"> ... </xs:element>
      </xs:schema>
   </types>
```

Source: `https://en.wikipedia.org/wiki/Web_Services_Description_Language`

Types Describes the data. The XML Schema language (also known as XSD) is used (inline or referenced) for this purpose.

WSDL XML interfaces

```
<!-- Abstract interfaces -->
   <interface name="Interface1">
      <fault name="Error1" element="tns:response"/>
      <operation name="Get" pattern="http://www.w3.org/ns/wsdl/in-out">
         <input messageLabel="In" element="tns:request"/>
         <output messageLabel="Out" element="tns:response"/>
      </operation>
   </interface>
```

Source: `https://en.wikipedia.org/wiki/Web_Services_Description_Language`

Interface Defines a Web service, the operations that can be performed, and the messages that are used to perform the operation.

WSDL XML the binding

```
<!-- Concrete Binding Over HTTP -->
   <binding name="HttpBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/http">
      <operation ref="tns:Get" whttp:method="GET"/>
   </binding>

<!-- Concrete Binding with SOAP-->
   <binding name="SoapBinding" interface="tns:Interface1"
            type="http://www.w3.org/ns/wsdl/soap"
            wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"
            wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-response">
      <operation ref="tns:Get" />
   </binding>
```

Source: `https://en.wikipedia.org/wiki/Web_Services_Description_Language`

Binding Specifies the interface and defines the SOAP binding style (RPC/Document) and transport (SOAP Protocol). The binding section also defines the operations.

WSDL XML the Service

```
<!-- Web Service offering endpoints for both bindings-->
   <service name="Service1" interface="tns:Interface1">
      <endpoint name="HttpEndpoint"
                binding="tns:HttpBinding"
                address="http://www.example.com/rest/"/>
      <endpoint name="SoapEndpoint"
                binding="tns:SoapBinding"
                address="http://www.example.com/soap/"/>
   </service>
```

Source: https://en.wikipedia.org/wiki/Web_Services_Description_Language

Service Contains a set of system functions that have been exposed to the Web-based protocols.

SOAP - Simple Object Access Protocol

> SOAP (abbreviation for Simple Object Access Protocol) is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to provide extensibility, neutrality, verbosity and independence. It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP), although some legacy systems communicate over Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

Source: `https://en.wikipedia.org/wiki/SOAP`

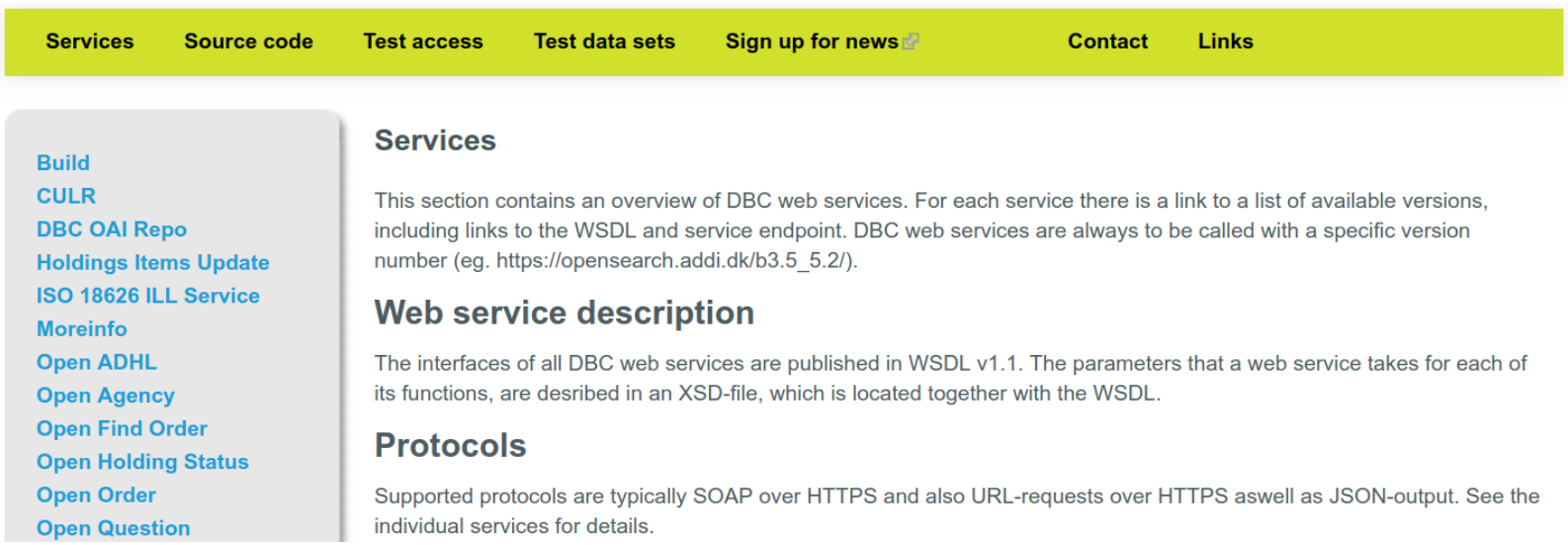
Utilizes UDDI (Universal Description, Discovery, and Integration)

Web Service Explained

The term "Web service"describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet Protocol backbone. XML is the data format used to contain the data and provide metadata around it, SOAP is used to transfer the data, WSDL is used for describing the services available and UDDI lists what services are available.

Source:`https://en.wikipedia.org/wiki/Web_service`

Sample WebServices



- Dansk Bibliotekscenter provides web services, lookup into various databases using Web services
- `https://opensource.dbc.dk/services/services`

Again frameworks and libraries

A simple “Hello World” http SOAP server:

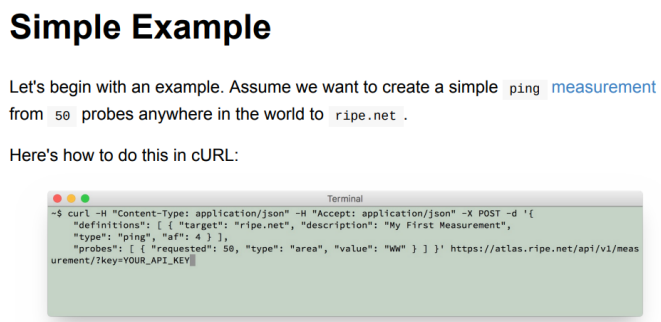
```
import SOAPpy
def hello():
    return "Hello World"
server = SOAPpy.SOAPServer(("localhost", 8080))
server.registerFunction(hello)
server.serve_forever()
```

And the corresponding client:

```
import SOAPpy
server = SOAPpy.SOAPProxy("http://localhost:8080/")
print server.hello()
```

- Dont process SOAP manually using home made programs
- `https://pypi.org/project/SOAPpy/`

Investigate examples from the internet



Simple Example

Let's begin with an example. Assume we want to create a simple `ping` measurement from `50` probes anywhere in the world to `ripe.net`.

Here's how to do this in cURL:

Terminal

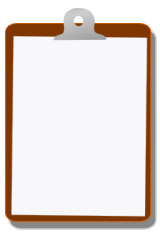
```
~$ curl -H "Content-Type: application/json" -H "Accept: application/json" -X POST -d '{
    "definitions": [ { "target": "ripe.net", "description": "My First Measurement",
    "type": "ping", "af": 4 } ],
    "probes": [ { "requested": 50, "type": "area", "value": "WW" } ] }' https://atlas.ripe.net/api/v1/meas
urement/?key=YOUR_API_KEY
```

- Which web services do you use? Can we find examples of XML and JSON web services
- Look into the services provided by DBC, what languages, formats, services - look at their Github account DBCDK
- We will use internet data from RIPE NCC Atlas service, if nothing else:

  `https://ripe-atlas-tools.readthedocs.io/en/latest/`

  `https://atlas.ripe.net/docs/api/v2/manual/pdf/ripe_atlas_api_V2_manual.pdf`
- If using the command line to set the API key: `ripe-atlas configure --set authorisation.create=e7fd3981-9592-481e-8bcd-f50d17e65de8`

For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools