Enterprise System Integration
(MTAT.03.229)

LECTURE 5: WEB SERVICES - SOAP

MOHAMAD GHIRIB
UNIVERSITY OF TARTU
Distributed Computing Technologies

Functionality
Distributed Computing Technologies
Distributed Computing Technologies
Distributed Computing Technologies

Server A

showProducts()

Package (JAR)

Interface

Server B

showProducts()
Distributed Computing – brief history

Java RMI (Remote Method Invocation) performs remote method invocation with support for direct transfer of serialized Java classes and distributed garbage-collection.

Common Object Request Broker Architecture (CORBA) – OMG: CORBA is language- and OS-independence, freedom from technology-linked implementations, and freedom from the details of distributed data transfers. CORBA specification is complex, expensive to implement entirely, and often ambiguous. Also, it faced problems with Firewalls.

Web Services (W3C) provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.  

A Web Service is a *standards-based, language-agnostic* software entity, which accepts *well-defined* formatted *requests* from other software entities, *producing well-defined* formatted *responses*. 
A Web Service is a **standards-based, language-agnostic** software entity, which accepts **well-defined** formatted **requests** from other software entities, **producing well-defined** formatted **responses**.
Web Services advantages

Solve most communication problems, as interaction is built on standardized languages/protocols (e.g., XML, WSDL, HTTP).

Reuse, a service within the system is only ever coded once and used over and over again by various applications.

Interoperability, web service are not designed to work on a specific language nor platforms.

Integration, web services enable easier communications within and across organisations.

Composability, several web services can be composed to deliver services than none of these services can be provide by its own.
Web Services key features

Discoverability, web services need to be "effectively" discovered to allow other software entities to consume them.

Abstraction, web services hide their logic from service consumers, they share only what is required to deliver the service.

Autonomy, a web service has almost complete control over the logic it encapsulates.

Loosely coupled, each service exists independently of the other services that make up the application.

Statelessness, stateless services treat each request separately regardless of any previous interaction, which is very important for providing scalable services².

Web services:

**SOAP (Simple Object Access Protocol)** - JAX-WS (Java API for XML Web Services) is a standardized API for creating and consuming SOAP.

**REST (Representational state transfer)** - JAX-RS (Java API for RESTful Web Services) is an architecture that provides support in creating web services according to REST.
SOAP is an **XML-based protocol** for accessing web services over HTTP. The **SOAP architecture** contains three key entities: 1- **Service provider**, 2- **Service requester**, and 3- **Service registry**.
The **service provider** creates a **SOAP service** and publishes its **description** in the **service registry**.
The **service requester** query the **service registry** to find the **service**, retrieves the **service description**, and then uses the description to **bind** to the **service implementation** and start **using** it.
SOAP architecture

WSDL provides information on how to use a web service, including a description of the service operations and binding information. UDDI defines a set of APIs for both service publication and discovery.
A **SOAP message** is normally auto-generated by the web service when it is called. A simple **SOAP message** has:

- The **envelope** element;
- The **header** element;
- The **body** element;
- The **fault** element (Optional).
A **SOAP message** is normally auto-generated by the web service when it is called.

A simple **SOAP message** has:

- The **envelope** element;
- The **header** element;
- The **body** element;
- The **fault** element (Optional).

```xml
<?xml version="1.0"?>
SOAP Envelope
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"

SOAP Header

<soap:Header>
...
</soap:Header>

SOAP Body

<soap:Body>
...
Optional SOAP Faults
</soap:Body>
</soap:Envelope>
```
Thank You for your attention

Mohamad Gharib
mohamad.gharib@ut.ee

unitartu
tartuuniversity