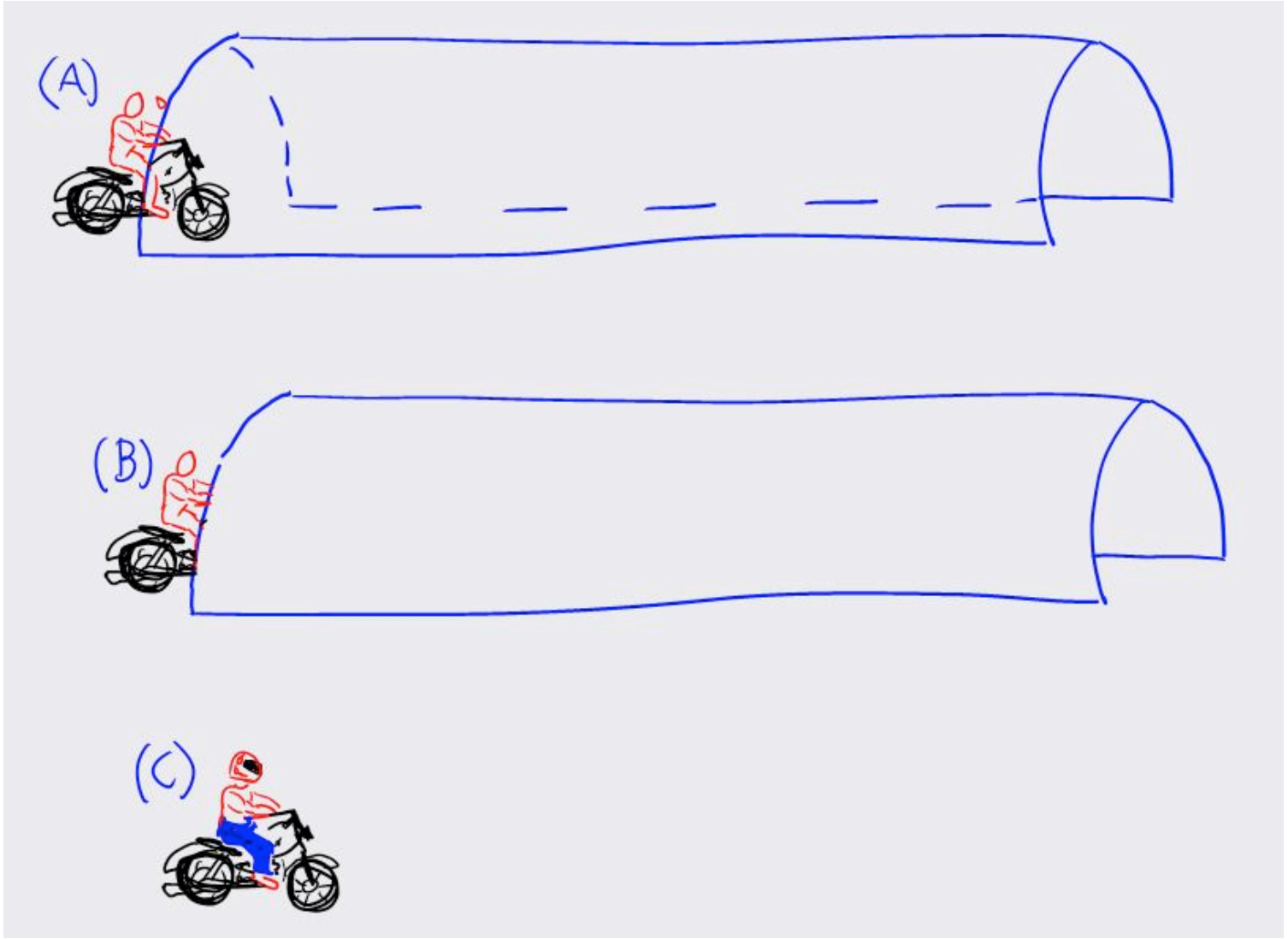
SoA – Service oriented architecture

Web Services

- WSDL – web services description language (describes the service itself)
- Strongly typed
- Remote method invocation in principal
- SOAP – Simple Object Access Protocol (transforms the request, parameters, and retunr value)
- XML-RPC
  - simpler,
  - XRDL exists (analogy to WSDL)

http://www.cloudidentity.com/blog/2005/04/25/END-TO-END-SECURITY-OR-WHY-YOU-SHOULDN-T-DRIVE-YOUR-MOTORCYCLE-NAKED/



WS-Security

- WS-Security standard
  - Compliant with XML Signature
  - http://www.ibm.com/developerworks/webservices/library/ws-security.html
- How to **sign** SOAP messages to assure integrity. Signed messages also provide non-repudiation (nepopiratelnost) and signer's identity
- How to **encrypt** SOAP messages to assure confidentiality.

WS-Security

```
<SignedInfo>
   <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
   <Reference URI="#sign_content_1043176028580">
      <Transforms>
         <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>FLuQTa/LqDIZ5F2JSaMRHSRuaiQ=</DigestValue>
   </Reference>
</SignedInfo>

<SignatureValue>
     kGlrrXjKku/WXKxID+JJkEXY+aGNYHc5dy8GwbLFtB5Msll2/MhwdnO9wastJ0gLPzLy3oHL
     7A8ggkMkjgAqnLg6PTzM7MdKoIAhe+xRHdOysamGucFJQRMrU+JQ4WATJt0bpdClwJy6mexT
     Su48mq1q5rM9YZh61P7UEUKt+EQ=
</SignatureValue>
```

WS-Security

```
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
   <KeyValue>
      <RSAKeyValue>
         <Modulus>
            2sW+eBjx5D2QMyr8ocZIZWNYHGf9zYhB4XWILPCTvhNV7dIe3l8ARepOA1ABFK2Omy
            pzb+Rb+nWQeo//yFz/28PmL63kdLiE72qmmQuzuPa5NXaV9pJ4JKw86QdLhGGpFIRH
            18Iugf3xLFwQEZqKYnblTUs7ftnTgW5r4HH492k=
         </Modulus>
         <Exponent>AQAB</Exponent>
      </RSAKeyValue>
   </KeyValue>
   <X509Data>
      <X509IssuerSerial>
         <X509IssuerName>OU=Java,O=IBM,L=Unknown,ST=Oklahoma,C=US</X509IssuerName>
         <X509SerialNumber>0</X509SerialNumber>
      </X509IssuerSerial>
      <X509SubjectName>CN=John Doe</X509SubjectName>
      <X509Certificate>
         MIIB0TCCAToCAQAwDQYJKoZIhvcNAQEEBQAwTzELMAkGA1UEBhMCVVMxETAPBgNVBAgTCE9rbGFo
         . . .
         VuzbLApPnXiehowYuA==
      </X509Certificate>
   </X509Data>
</KeyInfo>
```

WS-Security

- WS-Security standard
  - Compliant with XML Signature
  - http://www.ibm.com/developerworks/webservices/library/ws-security.html
- How to **sign** SOAP messages to assure integrity. Signed messages also provide non-repudiation (nepopiratelnost) and signer's identity **????**
- How to **encrypt** SOAP messages to assure confidentiality.

REST

- REST = Representational State Transfer
- Data oriented – not procedure (RPC) oriented
- Resources
  - Identified by URI
  - Accessed by HTTP methods
    - GET
    - PUT
    - POST
    - DELETE

RESTful services

	Collection URI	Element URI
GET	List the URIs of collection members	Retrieves the resource (collection member) represented by the requested media type
PUT	Replace the collection with another collection	Replace the collection member with the given resource. If the collection does not contain a resource with given URI, create a new collection element
POST	Create a new entry in the collection (its URL created automatically and usually returned as return value)	Typically not used Treat the addressed resource as a collection and create a new entry in it
DELETE	Delete the whole collection	Delete the resource (collection member)

See http://en.wikipedia.org/wiki/Representational_state_transfer

RESTful services

- Identification of resources by URI
- Representation of resources – typically XML, JSON, HTML
- PUT and DELETE – idempotent methods
- GET – safe method (or nullipotent) w.r.t. no side-effects
- Caching

RESTful services

- Easy to test (using http clients) like curl (command line), RESTClient (Firefox plugin)
- Security – typically through http
  - end-to-end
  - point-to-point

JAX-RS
RESTful Web Services Java API

- JAX-RS: Java API for RESTful Web Services
- JAX-RS - official part of Java EE 6 (version 1.1 +).
- For non-Java EE 6 environments a (small) entry in the web.xml deployment descriptor is required.
- Jersey: one of JAX-RS implementation
  - Jersey, the reference implementation from Sun (now Oracle).
- One of jersey tutorials:
  - http://jersey.java.net/nonav/documentation/latest/index.html

JERSEY - REST framework

Maven configuration – pom.xml

```
<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-bundle</artifactId>
  <version>1.15</version>
</dependency>

<!-- ASM - simple library that exposes the internal aggregate components of a given Java class through its visitor oriented API. -->
<dependency>
  <groupId>org.eclipse.jetty.orbit</groupId>
  <artifactId>org.objectweb.asm</artifactId>
  <type>jar</type>
  <version>3.3.1.v201105211655</version>
</dependency>

<dependency>
  <groupId>com.sun.jersey</groupId>
  <artifactId>jersey-json</artifactId>
  <version>1.15</version>
</dependency>
```

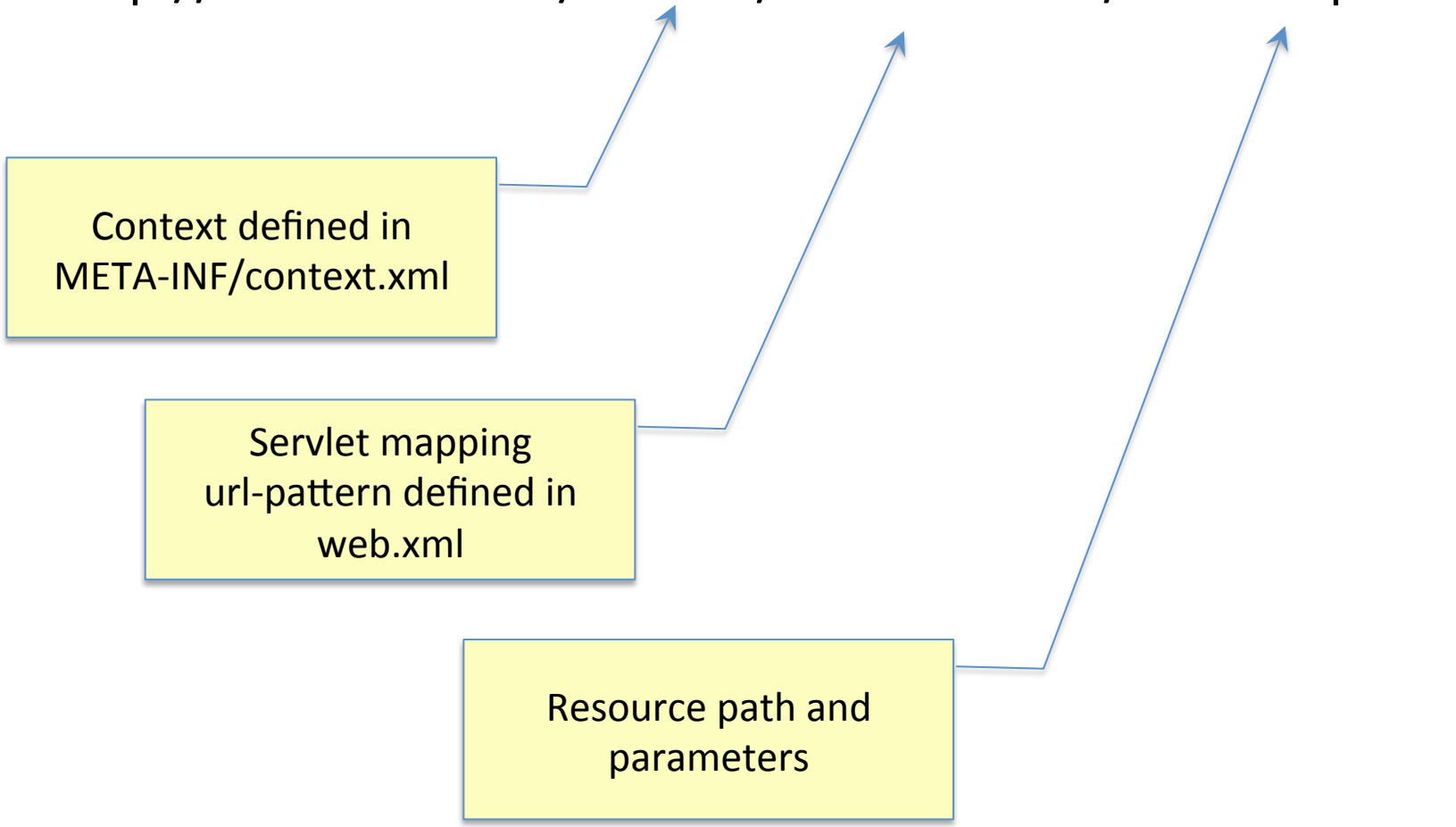
JERSEY - REST framework

web.xml

```
<servlet>
  <servlet-name>ServletAdaptor</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>cz.fel.kbss.jerseytutorialtomcat.resources</param-value>
  </init-param>
  <init-param>
    <param-name>com.sun.jersey.api.json.POJOMapping</param-name>
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>ServletAdaptor</servlet-name>
  <url-pattern>/webresources/*</url-pattern>
</servlet-mapping>
```

Resource URL structure

http://localhost:8080/context/webresources/resourcepath



Simple resource class

```
@Path("/helloworld")
public class HelloWorldResource {

@GET
@Produces("text/html")
@Path("/{p1}/{p2}/{p3}")
public String getClichedMessage(@PathParam("p1") String p1,
                                @PathParam("p2") String p2,
                                @PathParam("p3") String p3) {
      // method body
}

@Path("/getPOJO/{p1}/{p2}/{p3}")
@GET
@Produces(“application/json”)
public SimplePOJO getPOJO(@PathParam("p1") String p1,
                          @PathParam("p2") String p2,
                          @PathParam("p3") Integer p3) {
      // method body
}

@Path("/putPOJO/{p1}")
@PUT
@Consumes(“application/json”)
@Produces(“application/json”)
public SimplePOJO putPOJO(SimplePOJO pojo,
                          @PathParam("p1") int p1) {
      // method body
}
```

SimplePOJO

```
@XmlRootElement
public class SimplePOJO {

    private String attribute1;
    private String attribute2;
    private Integer attribute3;

    public SimplePOJO() {}

    public String getAttribute1() {...}

    public void setAttribute1(String attribute1) {...}

    public String getAttribute2() {...}

    public void setAttribute2(String attribute2) {...}

    public Integer getAttribute3() {...}

    public void setAttribute3(Integer attribute3) {...}
}
```

Simple JAX-RS client

```
public class App {

private static final String SERVICE_URL_GET =
"http://localhost:8080/context/webresources/helloworld/getPOJO/p1/p2/3";

private static final String SERVICE_URL_PUT =
"http://localhost:8080/context/webresources/helloworld/putPOJO/2";

public static void main( String[] args ) {

   ClientConfig clientConfig = new DefaultClientConfig();
   clientConfig.getFeatures().put(JSONConfiguration.
                                  FEATURE_POJO_MAPPING, true);
   clientConfig.getProperties().put(ClientConfig.
                                    PROPERTY_FOLLOW_REDIRECTS, true);
   Client client = Client.create(clientConfig);

   /*********************************************/
   WebResource resource = client.resource(SERVICE_URL_GET);
   WebResource.Builder builder = resource.type("application/json");
   SimplePOJO pojo = builder.get(SimplePOJO.class);
   /*********************************************/
   resource = client.resource(SERVICE_URL_PUT);
   builder = resource.type("application/json");
   builder.accept(MediaType.APPLICATION_JSON);
   pojo = builder.put(SimplePOJO.class, pojo);
}

}
```

Simple JAX-RS client

```
public class App {

private static final String SERVICE_URL_GET =
"http://localhost:8080/context/webresources/helloworld/getPOJO/p1/p2/3";

private static final String SERVICE_URL_PUT =
"http://localhost:8080/context/webresources/helloworld/putPOJO/2";

public static void main( String[] args ) {

   ClientConfig clientConfig = new DefaultClientConfig();
   clientConfig.getFeatures().put(JSONConfiguration.
                                  FEATURE_POJO_MAPPING, true);
   clientConfig.getProperties().put(ClientConfig.
                                    PROPERTY_FOLLOW_REDIRECTS, true);
   Client client = Client.create(clientConfig);


/* etc */
```

FEATURE_POJO_MAPPING
public static final java.lang.String **FEATURE_POJO_MAPPING**
A ResourceConfig feature, which allows you to enable JSON/POJO mapping functionality in Jersey. If set to true, your application will be capable of transforming JSON data to and out of POJOs. This also includes any JAXB beans existing in your application.
i.e. all those beans would not be processed via XML, but rather directly marshaled and un-marshaled to and from JSON using the POJO mapping functionality.

Simple JAX-RS client

```
public class App {

private static final String SERVICE_URL_GET =
"http://localhost:8080/context/webresources/helloworld/getPOJO/p1/p2/3";

private static final String SERVICE_URL_PUT =
"http://localhost:8080/context/webresources/helloworld/putPOJO/2";

public static void main( String[] args ) {

   ClientConfig clientConfig = new DefaultClientConfig();
   clientConfig.getFeatures().put(JSONConfiguration.
                                  FEATURE_POJO_MAPPING, true);
   clientConfig.getProperties().put(ClientConfig.
                                    PROPERTY_FOLLOW_REDIRECTS, true);
   Client client = Client.create(clientConfig);


/* etc */
```

PROPERTY_FOLLOW_REDIRECTS
static final java.lang.String **PROPERTY_FOLLOW_REDIRECTS** Redirection property.
A value of "true" declares that the client will automatically redirect to the URI declared in 3xx responses. The value MUST be an instance of Boolean.
If the property is absent then the default value is "true".