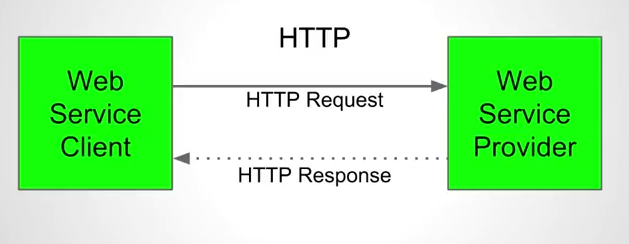
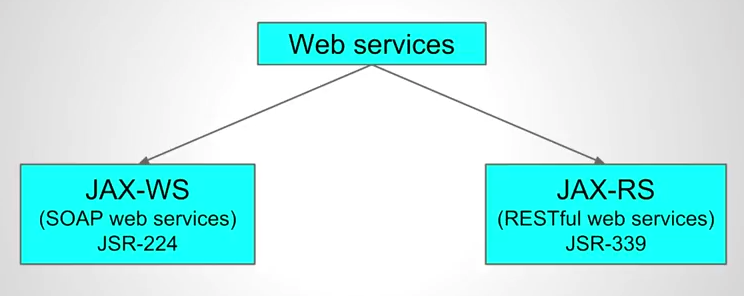
**Webservice’y w świecie Javy**

Webservice są usługami służącymi do wymiany komunikatów poprzez sieć między różnymi systemami.



**JAXB** służy do konwersji obiektów z i do XML’a.

W Javie mamy dwa rodzaje Web service’ow

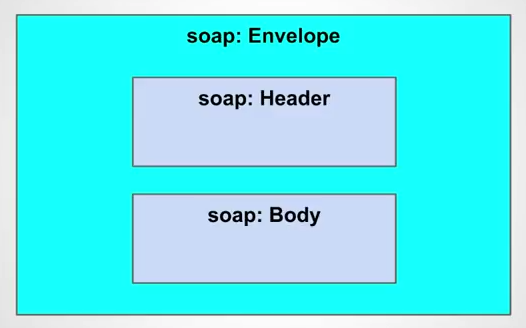


**JAX-WS**

Specyfikuje jak z Javovych service’ów i klas wygenerować WSDL i jak na podstawie WSDL’a wygenerować klasy JAVE’owe.

SOAP – Simple Object Access Protocol – opisuje wymianę komunikatów między naszymi web service’ami za pomocą tylko i wyłącznie XML’a.

Przykładowy format wiadomości SOAP:



WSDL (Web Service Description Language) – służy do opisu WebService’ów SOAP’owych. Może być przekazany komuś i służyć jako instrukcja do przygotowania wiadomości. Jest to XML’owy format, opisujący usługi sieciowe jako zbiór endpoint’ów operujących na wiadomościach, zawierających informacje zorientowane dokumentowo lub proceduralnie. WSDL jest bardzo rozszerzalny i pozwala na opis usług i ich wiadomości niezależni od formatu wiadomości, czy protokołu komunikacyjnego.

WSDL składa się z takich elementów:

* **Types** – kontener na definicje typów danych, wykorzystujący jakiś system typów np. XSD,
* **Message** – abstrakcyjna definicja przesyłanych danych, zgodnych ze zdefiniownaym typem,
* **Operation** – abstrakcyjny opis akcji obsługiwanych przez usługę,
* **Port Type** – abstrakcyjny zestaw operacji, obsługiwanych przez jeden lub więcej endpoint’ów,
* **Binding** – konkretny protokół i format danych wyspecyfikowany dla konkretnego typu portu,
* **Port** – pojedynczy endpoint zdefiniowany jako kombinacja bindinu i adresu sieciowego
* **Service** – kolekcja powiązanych endpoint’ów

**Przykładowe annotacje JAX-WS:**

* @javax.jws.WebService
* @javax.jws.Webparam
* @javax.jws.WebResult
* @javax.jws.soap.SOAPBinding
* @javax.jws.OneWay

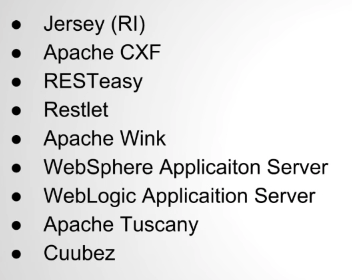
**WSDL Message Patterns**

* Request-Response
* One Way – bez odpowiedzi
* Notification – serwer wysyła notyfikacje bez odpowiedzi (nie jest w JAX-WS)
* Solicit-Response – serwer wysyła notyfikację i oczekuje odpowiedzi (nie jest w JAX-WS)

**JAX-RS (Representational State Transfer)**

Jest to taki pattern architektoniczny, który definiuje to jak powinny wyglądać nasze webservice’y.

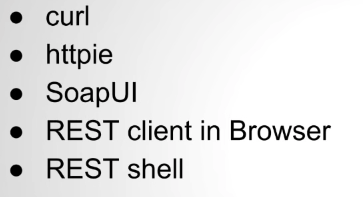
Implementacje JAX-RS:



REST (Representational State Transfer)

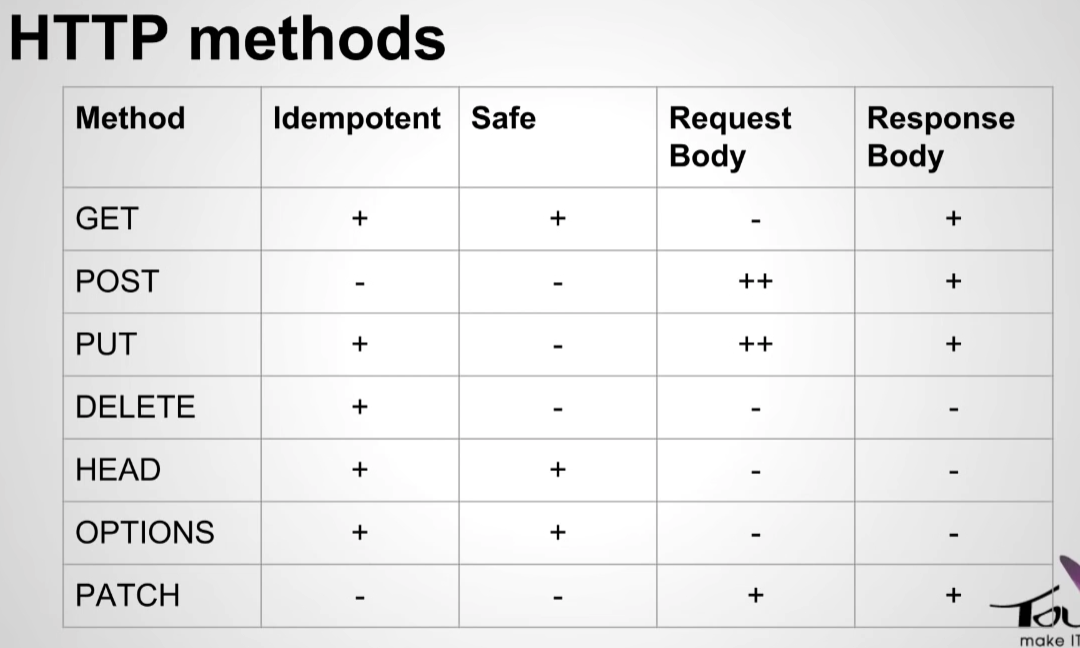
* Architektura klient-serwer,
* Bezstanowość,
* Cache – bardziej po stronie klienta,
* Jednolity kontrakt -
* Layeres System,
* Code-On-Demand,

Klienci REST’owi



Przykładowe annotacje JAX-RS

* @javax.ws.rs.GET
* @javax.ws.rs.POST
* @javax.ws.rs.DELETE
* @javax.ws.rs.PUT
* @javax.ws.rs.HEAD
* @javax.ws.rs.OPTION
* @javax.ws.rs.HttpMethod
* @javax.ws.rs.Path
* @javax.ws.rs.Consumes
* @javax.ws.rs.Produces



Parametry

* @javax.ws.rs.QueryParam
* @javax.ws.rs.PathParam
* @javax.ws.rs.HeaderParam
* @javax.ws.rs.CokkieParam
* @javax.ws.rs.MatrixParam
* @javax.ws.rs.FormParam
* @javax.ws.rs.BeanParam – do strzelania całym Bean’em

**2. Metody REST'owe?**

POST, GET, PUT, PATCH, DELETE

**3. Różnica między POST a PUT**

Post zazwyczaj wykorzystywany do tworzenia zasobów, a PUT do aktualizacji, przy czym w praktyce często są stosowane zamiennie.

PUT puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one. PUT is [idempotent](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.1.2), but paradoxically PUT responses are not cacheable.

indempotent - Describing an action which, when performed multiple times, has no further effect on its subject after the first time it is performed.

POST sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource. The POST method is not [idempotent](http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.1.2), however POST responses are cacheable so long as the server sets the appropriate Cache-Control and Expires headers.

The fundamental difference between the POST and PUT requests is reflected in the different meaning of the Request-URI. The URI in a POST request identifies the resource that will handle the enclosed entity. That resource might be a data-accepting process, a gateway to some other protocol, or a separate entity that accepts annotations. In contrast, the URI in a PUT request identifies the entity enclosed with the request -- the user agent knows what URI is intended and the server MUST NOT attempt to apply the request to some other resource. If the server desires that the request be applied to a different URI, it MUST send a 301 (Moved Permanently) response; the user agent MAY then make its own decision regarding whether or not to redirect the request.