Web Services

Service Oriented Architecture – SOA

Arquitetura orientada aos serviços

Definição: Arquitetura de sistemas distribuídos em que a funcionalidade é disponibilizada sob a forma de serviços (bem definidos e independentes)

Objetivos:

- Interoperabilidade de sistemas heterogéneos
- Independência de plataformas e linguagens

Service Oriented Architecture – SOA

Arquitetura implementada em:

CORBA: **Common Object Request Broker Architecture. (Especificação muito completa mas difícil de implementar e de utilizar)**

DCOM : **Distributed Component Object Model (MicroSoft) DCOM/Com+/ActiveX**

Sun-Oracle Java RMI **(Remote Method Invocation)**

- Desvantagens :

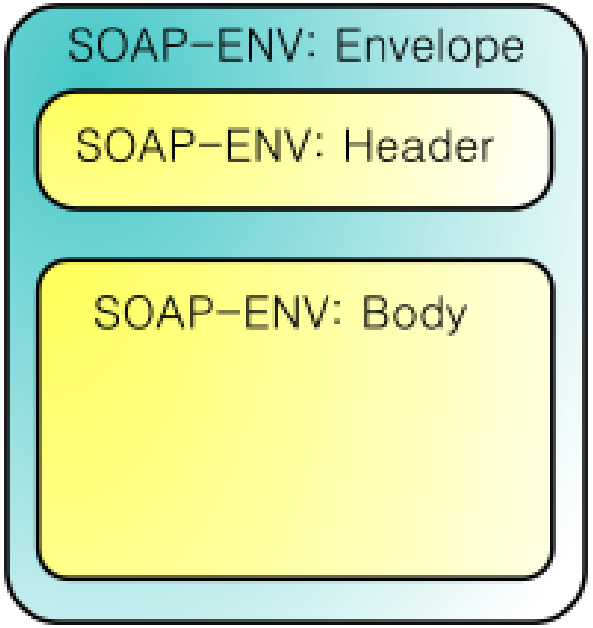
- Tecnologia proprietária
- Específica para uma plataforma
- Não interoperáveis
- Necessário configurar Firewalls

Os Web Services surgem como uma nova resposta SOA

- Usam HTTP como protocolo de transporte das mensagens
- Com a explosão da web torna-se natural a escolha do transporte over-HTTP
- São descritos através de uma linguagem de descrição (em XML): WSDL (Web Services Description Language)
  - . Definição das operações suportadas
  - . Definição dos Tipos utilizados
  - . URL (Endpoint) do Web Service
  - . etc.

Troca de Mensagens XML segundo o protocolo SOAP

(Simple Object Access Protocol)



Web Services - Definições da Internet

Definição do www consortium (W3C): 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.

Web Services - Definições da Internet

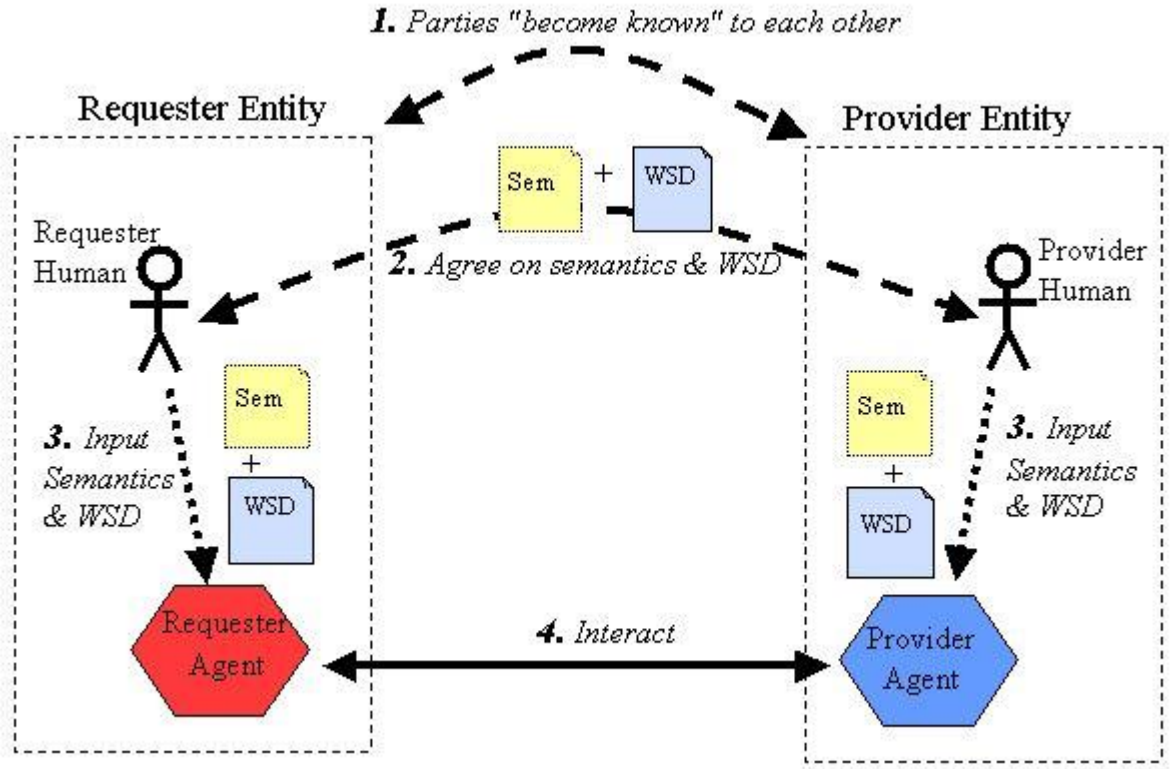
Outras definições:

- *Web services" are part of an effort to build a distributed computing platform for the Web*

- *Web services" are enabling technology for systematic application-to-application interaction on the Web*

- *A web service is a programmable component that provides a service and is accessible over the Internet.*

Web Services



(from http://www.w3.org/TR/ws-arch/#introduction)

Web Services – Vantagens

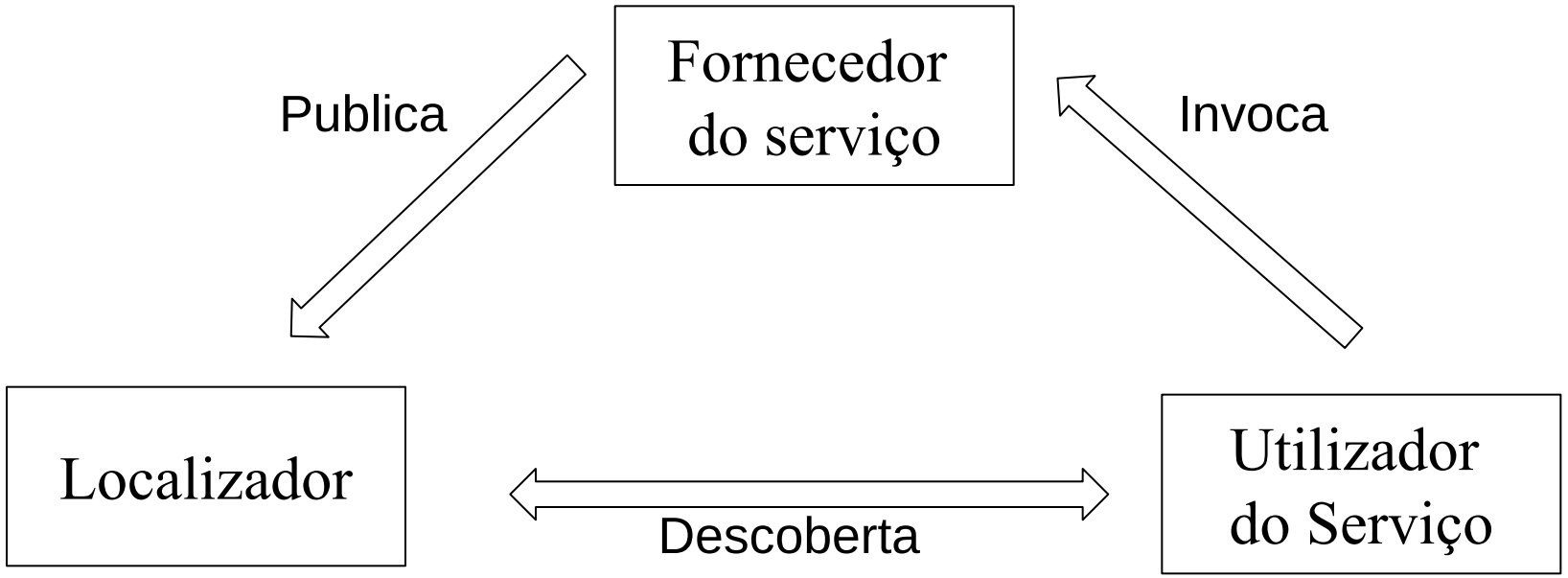
- Independência de plataformas e linguagens.

- Standards abertos: . HTTP . SOAP . WSDL . XML

- Menos suscetível a restrições de segurança (e.g. a nível de *firewalls)*
  . Não há necessidade de configurar portos específicos
  . Tráfego trocado na rede é HTTP

- Críticos *argumentam que a performance (pelo facto de lidar com mensagens XML e HTTP) é inferior a um protocolo binário.*

Web Services – Arquitetura



Publicação – processo, opcional pelo qual o fornecedor de WS dá a conhecer a existência do serviço. Regista o serviço no reservatório de WS (UDDI – Universal Description, Discovery and Integration)

Descoberta – processo, opcional, através do qual uma aplicação cliente faz uma pesquisa sobre o reservatório de WS e toma conhecimento da existência do WS requerido.

Descrição – a aplicação cliente tem acesso à interface do WS, à descrição das funcionalidades disponibilizadas, e aos tipos de mensagens que permitem aceder a essas funcionalidades. (WSDL - Web Services Description Language)

Invocação – processo pelo qual cliente e servidor interagem, através da troca de mensagens.
(SOAP - Simple Object Access Protocol)

XML – eXtensible Markup Language

- Linguagem de anotação estruturada de documentos
- Apresentar a informação de forma estruturada facilita:

. Validação

. Reutilização

. Normalização

Ex.lo: <aluno>

<nome> Maria Sousa</nome>

<numero> 1111</numero>

<curso> Engenharia Informática</curso>

</aluno>

SOAP – Simple Object Access Protocol

- Protocolo para a troca de mensagens, em formato XML, entre processos
- Cada recetor de uma mensagem chama-se *endpoint*
- Cada *endpoint* quando recebe uma mensagem:

  . Analisa-a verificando se alguma parte lhe é endereçada

  . Verifica se das partes que lhe são dirigidas alguma é para processar e executa as operações

  . Se parte da mensagem é para reenviar para outro *endpoint* retira a parte já processada e envia.

WSDL – Web Services Description Language

- Formato XML para descrever WS e a forma de os invocar
- Quatro tipos básicos de operações (primitivas de transmissão):

  . Operação de pedido – o servidor recebe um pedido mas não envia resposta

  . Operação de pedido/resposta – o servidor recebe um pedido e envia uma resposta

  . Operação de solicitação/resposta – o servidor envia uma mensagem ao cliente e recebe uma resposta

  . Operação de notificação – mensagem enviada pelo servidor e para a qual não é esperada nenhuma resposta

UDDI – Universal Description, Discovery and Integration

- Projeto iniciado pela IBM, Microsoft e Ariba para criar um método standard capaz de divulgar e descobrir os WS (2000)
- Consistia numa base de dados replicada de informação sobre WS (diversas empresas disponibilizavam réplicas da Base de Dados: IBM, Microsoft, SAP, Oracle)
- Com um serviço de registo com três funções básicas:

. Divulgação . Procura . Mapeamento

Entretanto IB, SAP e Microsoft encerraram os seus nós (2006/2007)

RESTful Web Services

REST - Representational State Transfer

- SOAP um arquitetura baseada em mensagens.

- REST é uma arquitetura para sistemas distribuídos de hypermedia – sistemas onde recursos como texto, imagens, vídeo são guardados numa rede e ligados via hyperlinks (por ex.lo a WWW).

RESTful Web Services

- Abstração fundamental é a noção de “Recurso”

- Um recurso é uma entidade que tem um URI (Universal Resource Identifier)

Roy Fielding (Co-Founder Apache)
Architectural Styles and the Design of Network-based Software Architectures. PhD 2000.
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

RESTful Web Services

- REST web services comunicam (cliente/servidor)
sobre o protocolo HTTP usando o vocabulário HTTP

É uma arquitetura baseada na transferência de representações de recursos através de pedidos (requests) e respostas (responses).

Os recursos são manipulados através de um conjunto fixo de operações:

Métodos: PUT, GET, POST, DELETE.

RESTful Web Services

Dados e funcionalidades são considerados recursos, e são acedidos pelos seus URIs (Uniform Resource Identifiers)

- HTTP URI syntax (paths, parameters, etc.)
- HTTP Response codes.

As mensagens são auto-descritivas:

Os recursos são desacoplados da sua representação, podendo o seu conteúdo ser acedido em vários formatos:
xml, json (javascript object notation), html, plain text, PDF, JPEG, etc

Java API para RESTful Web Services – **Jax-RS**

Anotações Java-RS são usadas para definir **recursos** a as **ações** que podem ser realizadas sobre esses recursos:

Uma **"Root resource class"** é um POJO que tem a anotação **@path** ou tem pelo menos um método anotado com um **"request method designator"** como: **@GET, @PUT, @POST, @DELETE**.

"Resource methods" são métodos anotados com um "request method designator"

Java API para RESTful Web Services – **Jax-RS**

Anotações

@Path - caminho relativo para a localização da classe

exemplo: @Path (“/HelloWorld”)

@Path (“/users/{username}”) - username é o nome de uma variável que é passada à aplicação.

@GET - O método Java anotado com @ GET vai processar pedidos HTTP GET.

Java API para RESTful Web Services – **Jax-RS**

Exemplo:

```
/**
 * Root resource (exposed at "HelloWorldApplication" path)
 */
```

@Path("HelloWorldApplication") ← A classe vai ser armazenada em / HelloWorldApplication

```
public class HelloWorld {
    @Context
    private UriInfo context;
    /** Creates a new instance of HelloWorld */
    public HelloWorld() {
    }
```

Java API para RESTful Web Services – **Jax-RS**

```
/**
 * Retrieves representation of an instance of HelloWorldApplicationWorld.HelloWorld
 * @return an instance of java.lang.String
 */

 @GET
@Produces("text/html")  ← especifica o "MIME media type" em que o recurso vai ser enviado
                          para o cliente
public String getHtml() {
    return "<html lang=\"en\"><body><h1>Hello, World!!   PPX </body></h1></html>";
}
```

Java API para RESTful Web Services – **Jax-RS**

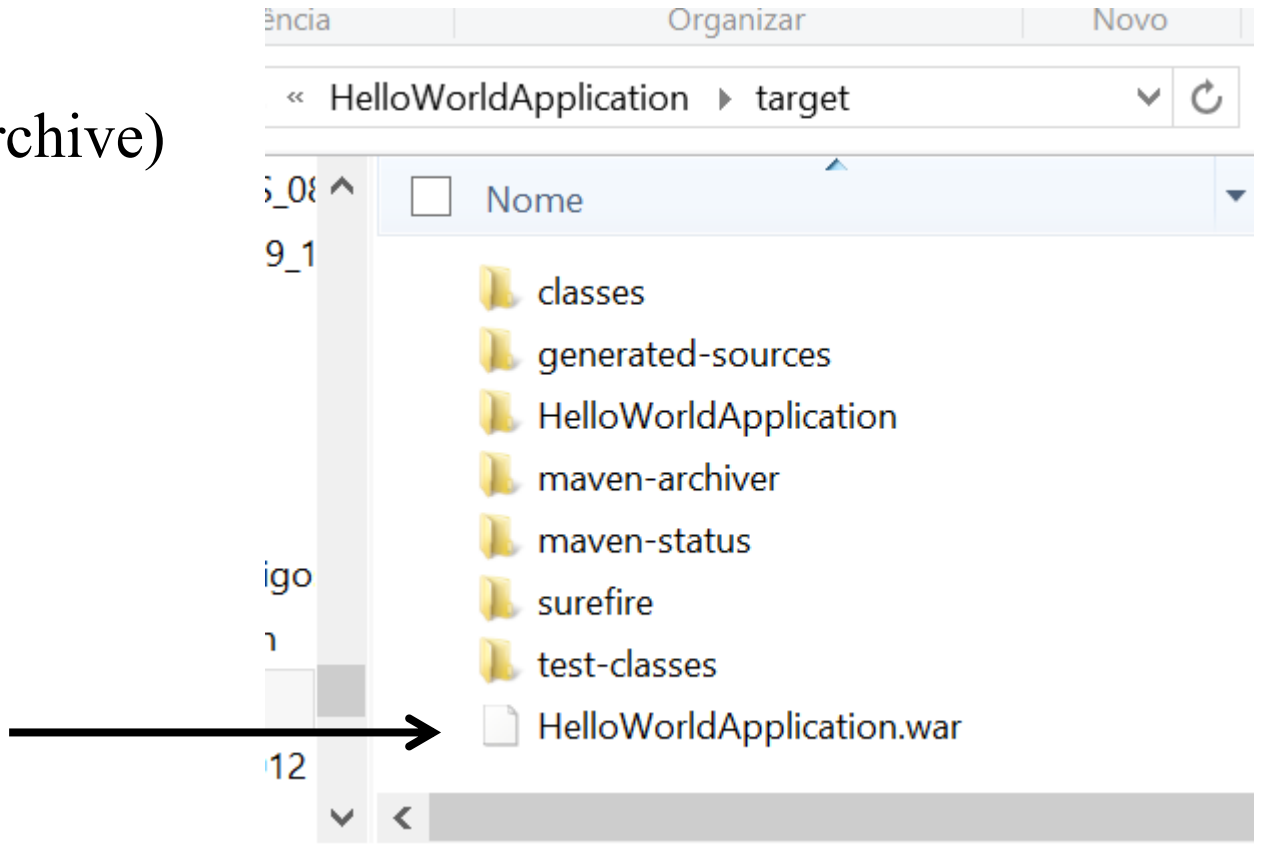
@POST

@**Consumes**(“text/plain”) ← especifica o “MIME media type” enviado pelo cliente para o recurso.

```
public void postHtml(String message) {
        // …
}
```

Java API para RESTful Web Services – Jax-RS

Uma aplicação Jax-RS consiste em pelo menos um package com uma classe dentro de um war (web archive)



Java API para RESTful Web Services – Jax-RS

O endereço base a partir do qual os vários recursos de uma aplicação respondem pode ser construído usando a anotação,

@ ApplicationPath

numa subclasse de javax.ws.rs.core.Application arquivada dentro do ficheiro war:

```
15      * @author ievans
16      */
17     @ApplicationPath("/")
18     public class HelloApplication extends Application {
19
20     }
21
```

Java API para RESTful Web Services – Jax-RS

Exemplo do exercicio A (FP11):

Para cada entity criada a partir das tabelas da base de dados, foi criado um stateless bean que expõe os recursos para aceder à respectiva tabela:

```
@Stateless
@Path("entities.customer")
public class CustomerFacadeREST extends AbstractFacade<Customer> {
    @PersistenceContext(unitName = "ws1PU")
    private EntityManager em;
        public CustomerFacadeREST() {
                super(Customer.class);
        }
```

Java API para RESTful Web Services – Jax-RS

```
@POST
 @Override
 @Consumes({"application/xml", "application/json"})
 public void create(Customer entity) {
     super.create(entity);
 }

@PUT
 @Path("{id}")
 @Consumes({"application/xml", "application/json"})
 public void edit (  @PathParam("id") Integer id, Customer entity) {
     super.edit(entity);
 }
```

...

Java API para RESTful Web Services – Jax-RS

```
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }
    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    } …
```

Java API para RESTful Web Services – Jax-RS

```
@javax.ws.rs.ApplicationPath("webresources")      ← URI base
public class ApplicationConfig extends Application {
    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> resources = new java.util.HashSet<>();
        addRestResourceClasses(resources);
        return resources;
    }                                          //      As “resource classes” são registadas na aplicação.

    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(entities.service.CustomerFacadeREST.class);
        resources.add(entities.service.DiscountCodeFacadeREST.class);
        resources.add(entities.service.MicroMarketFacadeREST.class);
    }
```