# Arquitetura Orientada a Serviços

**Prof. Cidcley T. de Souza**

**cidcley@ifce.edu.br**

# Web Services
## *Prática com Java*

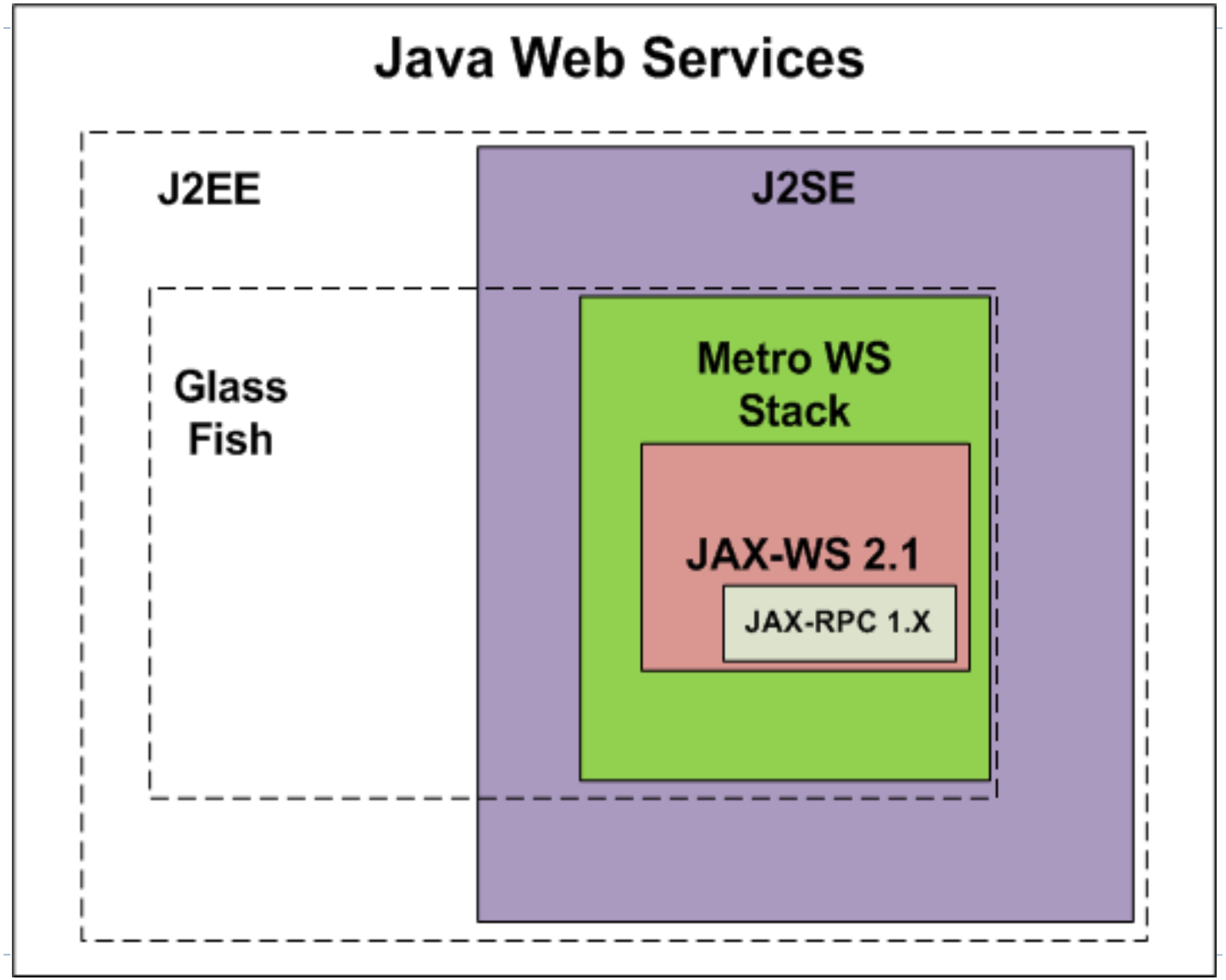
# Ferramentas Utilizadas

- JAX-WS
  - Java API for XML Web Services
  - API disponível no  (J2SE) >= 5
  - Parte do projeto GlassFish

# JAX-WS

- JAX-WS
  - Utilizada para construir Web Services / Clientes Java
  - Utiliza Mensagens SOAP sobre HTTP (ou SMTP)
  - Abrastrai do desenvolvedor a criação de mensagens SOAP
  - Fornece suporte a mensagens no estilo RPC

# JAX-WS


Java Web Services
J2EE
J2SE
Glass Fish
Metro WS Stack
JAX-WS 2.1
JAX-RPC 1.X

# Implementação utilizando JAX-WS

1) Codificar e compilar os arquivos do servidor
   - Service Endpoint Interface (SEI) (Optional)
   - Service Implementation Bean (SIB)
1) Implantar o serviço usando Endpoint. publish() (ou outras ferramentas como: Metro, Glass Fish, etc.)
   - Gera o arquivo WSDL

# Implementação utilizando JAX-WS

1) Executar “wsimport” para criar um proxy (stub) usando o WSDL do serviço

4)Codificar o cliente de acordo com o stub

5)Executar o Cliente

Serviço Exemplo: Inverter uma string

# Interface do Serviço (SEI)

- Criar um diretório para colocar os arquivos do serviço. No nosso caso “inverter”;
- Criar arquivo com o SEI (InverterITF.java)
- Criar arquivo SIB (InverterImpl.java)
- Compilar

# Interface do Serviço (SEI)

```
package inverter;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface InverterItf{
    @WebMethod String inverter(String name);
}
```

# Implementação do Serviço (SIB)

```
package inverter;
import javax.jws.WebService;

@WebService(endpointInterface = "inverter.InverterItf")
public class InverterImpl implements InverterItf{
      public String inverter(String msg) {
        StringBuffer strbuf = new StringBuffer(msg);
        System.out.println("Recebido: "+msg);
        String retorno = (strbuf.reverse()).toString();
        return retorno;
       }
}
```

# Publicador do Serviço

- Criar a classe “Publicador”
- Utilizar a classe Endpoint
- Executar o publicador
- Ver o arquivo WSDL gerado utilizando o seguinte endereço no browser:
  - http://localhost:9999/inverter?wsdl

# Publicador do Serviço

```
import javax.xml.ws.Endpoint;

import inverter.InverterImpl;

public class Publicador {
    public static void main(String[] args) {

   Endpoint.publish("http://localhost:9999/inverter",
   new
                              InverterImpl());
   }
}
```

# Consumidor do Serviço

- Criar a classe “Cliente”
- Utilizar o wsimport para gerar stubs a partir do wsdl
  - wsimport http://localhost:9999/inverter?wsdl
- Executar o Cliente

# Consumidor do Serviço

```
import inverter.InverterItf;
import inverter.InverterImplService;
public class Cliente {
    public static void main(String[] args){
    InverterImplService service = new
    InverterImplService();

    InverterItf inverterservice =
    service.getInverterImplPort();


    System.out.println(inverterservice.inverter("Test
    e"));
    }
}
```