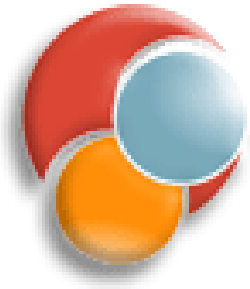


Servicios Web

Sesión 3: Servicios Web avanzados

© 2008-2009 Depto. Ciencia de la Computación e IA

Puntos a tratar

- Introducción a WSIT
- Optimización de mensajes
- Envío fiable
- Servicios con estado
- Servicios web seguros
- Transacciones en servicios web

© 2008-2009 Depto. Ciencia de la Computación e IA

WSIT

- *Web Services Interoperability Technologies*
- Plataforma Java para servicios web
  - Incorpora tecnologías de segunda generación

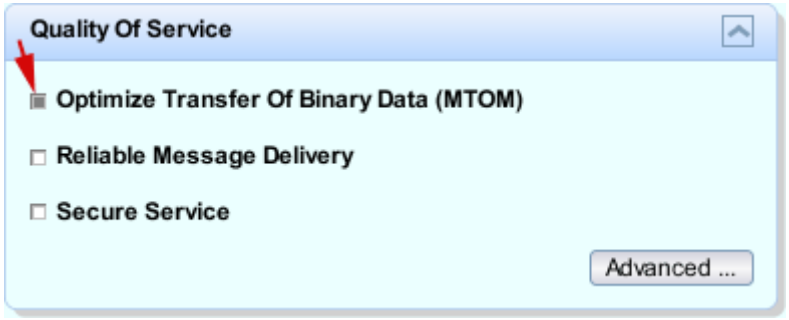
Optimización	MTOM
Fiabilidad	WS-ReliableMessaging
Seguridad	WS-Security, WS-Trust
Transacciones	WS-AtomicTransaction

  - Mejorar QoS
  - Interoperabilidad con .NET 3.0
- Equivalente a WCF de Microsoft
  - *Windows Communication Foundation*

© 2008-2009 Depto. Ciencia de la Computación e IA

Optimización de mensajes

- Mejorar eficiencia en el envío
- Los mensajes SOAP son muy extensos
  - La información se envía como texto
  - Gran cantidad de “envoltorios” en el XML
- Especialmente ineficiente para enviar datos binarios
- MTOM
  - *Message Transmission Optimization Mechanism*
  - Optimización del envío de datos binarios
  - Se envían como adjunto



© 2008-2009 Depto. Ciencia de la Computación e IA

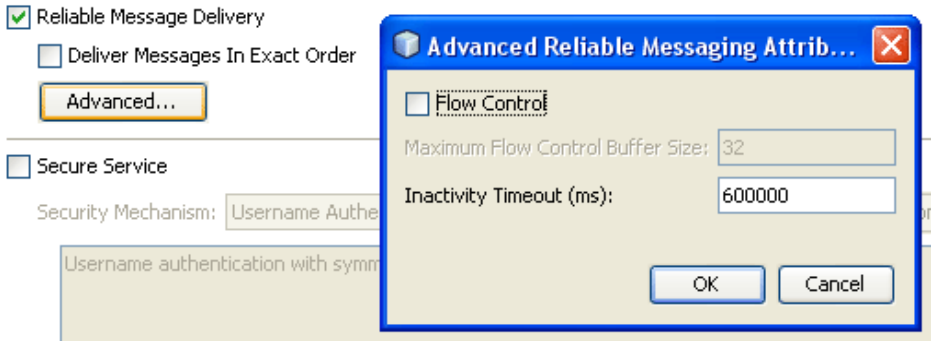
Otras formas de optimización

- *Fast Infoset*
  - Formato binario análogo a XML
  - No hay pérdida de información
  - Los mensajes son más compactos
  - Mayor eficiencia para serializar/deserializar
  - Adecuado para dispositivos limitados
- Transmisión vía TCP
  - Utiliza directamente protocolo TCP en lugar de HTTP
- Ambos métodos se usan cuando son soportados tanto por el cliente como por el servidor

© 2008-2009 Depto. Ciencia de la Computación e IA

Envío fiable de mensajes

- Las comunicaciones pueden fallar
  - ¿Reintentar la invocación?
  - Peligroso cuando sólo deba ejecutarse una vez
  - En las operaciones *oneway* no habrá confirmación
- *Reliable Messaging*
  - Los mensajes se entregan 1 y sólo 1 vez.
  - Se crea canal de datos
  - Se puede controlar que lleguen en orden (opcional)



© 2008-2009 Depto. Ciencia de la Computación e IA

Canal de datos

- Deben enviarse mensajes adicionales
  - Para controlar el flujo
  - Alto coste
    - Sobretodo si se controla el orden
  - Debemos cerrar el flujo
    - Se cierra después de un tiempo sin actividad

```
ConversionSW port = service.getConversioSWPort();
int result = port.euro2ptas(18.95);
out.println("Result: " + result);
((Closeable)port).close();
```


© 2008-2009 Depto. Ciencia de la Computación e IA

Servicios con estado

- Mantienen información de estado de cada cliente
  - Por ejemplo un carrito de la compra
    Cada llamada al servicio añade un producto al carrito
  - Disponible a partir de JAX-WS 2.1
- Cada cliente accede a una instancia del servicio
  - El estado se mantiene mediante variables de instancia
- Basado en *WS-Addressing*. Permite especificar:
  - Dirección del *endpoint*
  - Instancia concreta del servicio a la que acceder

© 2008-2009 Depto. Ciencia de la Computación e IA

Ejemplo de servicio *stateful*

```
@Stateful
@WebService
@Addressing
public class CuentaSW {
  private int id; private int saldo;

  public CuentaSW(int id) { this.id = id; this.saldo = 0; }

  public void ingresar(int cantidad) { saldo += cantidad; }
  public int saldo() { return saldo; }
  public void cerrar() { manager.unexport(this); }

  public static StatefulWebServiceManager<CuentaSW> manager;
}
```


© 2008-2009 Depto. Ciencia de la Computación e IA

Crear instancias

- Utilizamos un servicio adicional

```
@WebService
public class BancoSW {
  static Map<Integer, CuentaSW> cuentas = new HashMap();

  @WebMethod public synchronized W3CEndpointReference
                                          abrirCuenta(int id) {
    CuentaSW c = cuentas.get(id);
    if (c == null) {
      c = new CuentaSW(id);
      cuentas.put(id, c);
    }
    W3CEndpointReference endpoint = CuentaSW.manager.export(c);
    return endpoint;
  }
}
```


© 2008-2009 Depto. Ciencia de la Computación e IA

Cliente de servicios stateful

```
BancoSWService bService = new BancoSWService();
CuentaSWService cService = new CuentaSWService();
BancoSW bPort = bService.getBancoSWPort();

W3CEndpointReference endpoint = bPort.abrirCuenta(1);
CuentaSW c = cService.getPort(endpoint,CuentaSW.class);

c.ingresar(10);
c.ingresar(5);
out.println("Saldo: " + c.saldo());
c.ingresar(20);
out.println("Nuevo saldo: " + c.saldo());
c.cerrar();
```


© 2008-2009 Depto. Ciencia de la Computación e IA

Seguridad en servicios web

- Diferenciamos 3 aspectos
  - Confidencialidad
    Los datos transmitidos no deben poder ser vistos sin autorización
    Solución: cifrado con clave simétrica/asimétrica
  - Integridad
    Los datos no deben poder ser alterados
    Solución: huella/firma digital
  - Autentificación
    Conocer la identidad del otro extremo
    Solución: Credenciales (*login/password*), firma con certificado digital
- Seguridad en la red
  - Se permite invocar operaciones a través de HTTP
  - Los *firewalls* no impiden el acceso

© 2008-2009 Depto. Ciencia de la Computación e IA

Confidencialidad e integridad

- La información contenida en los mensajes SOAP puede ser confidencial
- Solución:
  - Como los mensajes se envían por HTTP, podemos cifrarlos y firmarlos con SSL
- Problema:
  - Si el mensaje debe atravesar una cadena de servicios, debe ser descifrado dentro de cada uno de ellos

    Los datos estarán inseguros dentro de cada nodo intermedio
  - Solución:

    Cifrar y firmar partes del mensaje por separado

© 2008-2009 Depto. Ciencia de la Computación e IA

Transporte vs Mensaje

- Seguridad a nivel de transporte
  - SSL (*https*)
  - Sólo ofrece seguridad durante el transporte
  - Más eficiente
  - No plantea problemas con punto-a-punto
- Seguridad a nivel de mensaje
  - *WS-Security*
  - Se protege hasta la llegada al *endpoint*
  - Independiente del protocolo de transporte
  - Requiere que los actores soporten *WS-Security*

© 2008-2009 Depto. Ciencia de la Computación e IA

Autentificación

- Podemos necesitar identificar al usuario
  - Para prestarle un servicio personalizado
  - Para comprobar si tiene permiso para usar el servicio
  - etc...
- *Tokens* de autentificación

Username token	Login/password
X.509 token	Certificado digital
SAML token	Autentificación y autorización

© 2008-2009 Depto. Ciencia de la Computación e IA

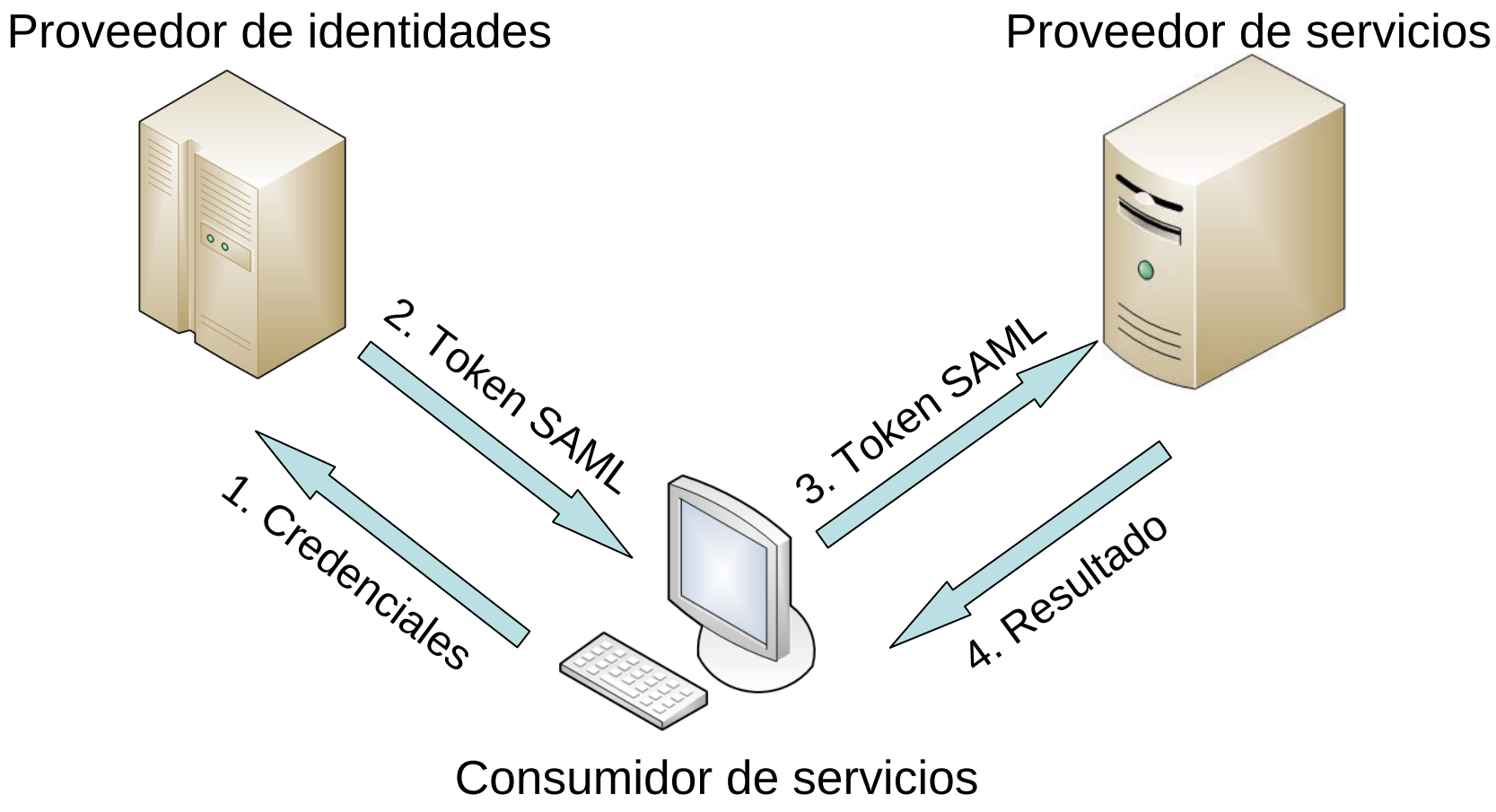
Contexto compartido

- Problema:
  - Si utilizamos servicios de distintos servidores, tendremos que autentificarnos para cada uno por separado
- Solución:
  - Crear contexto compartido (proveedor de identidades) donde puedan consultar información sobre la autentificación

    MS Passport, Liberty Project
- Sun Access Manager
  - Puede ser instalado como *addon* de Glassfish

© 2008-2009 Depto. Ciencia de la Computación e IA

Single Sign-On (SSO)



© 2008-2009 Depto. Ciencia de la Computación e IA

Modos de seguridad con WSIT

- *Username Authentication with Symmetric Keys*
  - Seguridad a nivel de mensaje.
  - Autentificación mediante *login* y *password*.
  - Clave simétrica para cifrar y firmar.
- *Mutual Certificates Security*
  - Seguridad a nivel de mensaje.
  - Autentificación mediante un certificado X.509.
- *Transport Security (SSL)*
  - Seguridad a nivel de transporte
  - Se protege la ruta en `web.xml`.

© 2008-2009 Depto. Ciencia de la Computación e IA

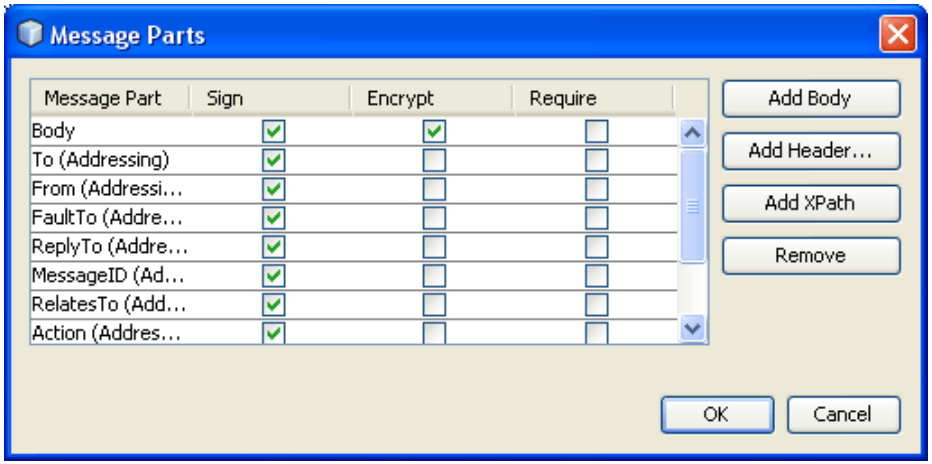
Otros modos

- Basados en *tokens* SAML
  - *SAML Authorization over SSL*
  - *SAML Sender Vouches with Certificates*
  - *SAML Holder of Key*
  - Se pueden utilizar junto a *Sun Access Manager*
- Basados en STS (*Secure Token Service*)
  - Crear un servicio de tipo STS
  - El STS proporciona un *token* de seguridad
  - Accedemos al servicio proporcionando dicho *token*

© 2008-2009 Depto. Ciencia de la Computación e IA

Configuración de la seguridad

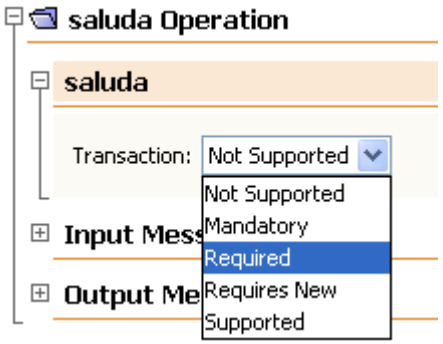
- Crear usuario en *realm file (username token)*
- Instalar certificados X.509 v3
  - *Keystore* (certificados propios)
  - *Truststore* (certificados raíz)
- Partes del mensaje



© 2008-2009 Depto. Ciencia de la Computación e IA

Transacciones en servicios web

- Basadas en *WS-AtomicTransaction*
- No se necesitan nuevas APIs
  - Se usa JTA
  - Se comportan igual que en los EJBs
- Se configuran a nivel de operación
  - Mismos modos que en EJBs



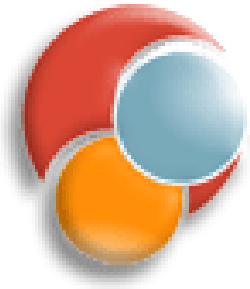
© 2008-2009 Depto. Ciencia de la Computación e IA

Cliente con JTA

- Desde servlet o EJB
  - `UserTransaction`
- Sólo desde EJB
  - *Container-Managed Transaction* (CMT)

```
@Resource UserTransaction ut;
HotelSW pHotel = sHotel.getHotelSW();
VuelosSW pVuelos = sVuelos.getVuelosSW();
ut.begin();
boolean hotelReservado = pHotel.reservaHotel(datosHotel);
boolean vueloReservado = pVuelos.reservaVuelo(datosVuelo);
if(!hotelReservado || !vueloReservado) {
  ut.rollback();
} else {
  ut.commit();
}
```


© 2008-2009 Depto. Ciencia de la Computación e IA



¿Preguntas...?

© 2008-2009 Depto. Ciencia de la Computación e IA