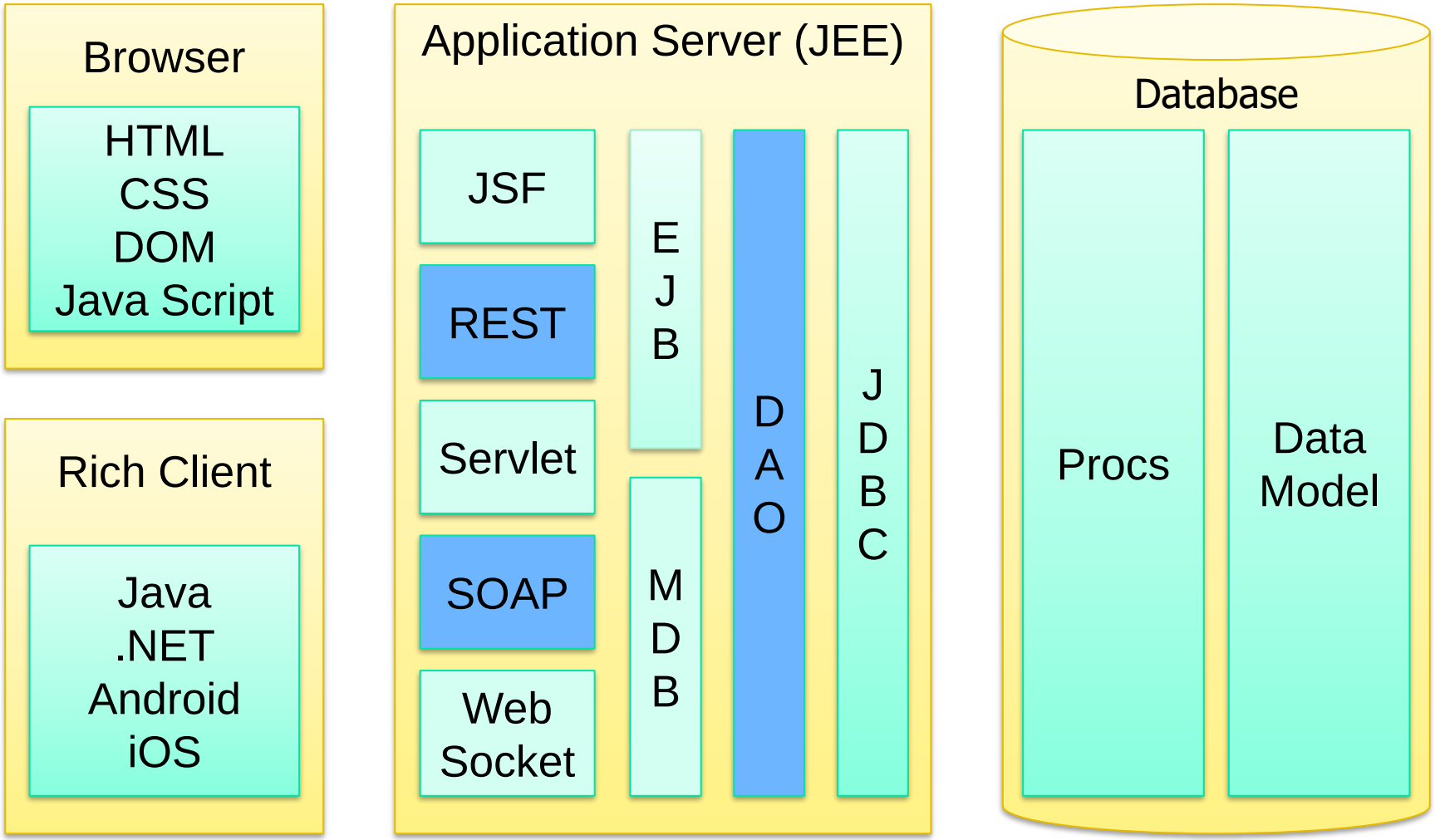
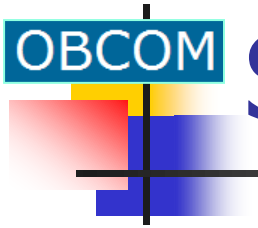
Aplicaciones Web (parte 4)

Eduardo Ostertag Jenkins, Ph.D.
OBCOM INGENIERIA S.A. (Chile)
Eduardo.Ostertag@obcom.cl

REST, SOAP y DAO





Servicios REST JAX-RS

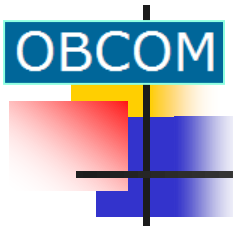
- Requiere una clase “Aplicación REST”
  - Extender javax.ws.rs.core.**Application**
  - Decorar con @**ApplicationPath**("/path")
  - Set<Object> **getSingletons**() {…}
- Require una clase “Servicio REST”
  - Decorar @**Consumes**(MediaType.APPLICATION_JSON)
  - Decorar @**Produces**(MediaType.APPLICATION_JSON)
  - void setServletContext(ServletContext context) {…}
  - Decorar métodos con @**GET** o @**POST**
  - Decorar métodos con @**Path**("/path")

Clase Aplicación REST

```
@ApplicationPath("/sample")
public class SampleRestApp extends Application
{
    private Set<Object> singletons;

    public SampleRestApp()
    {
        singletons = new HashSet<>();
    }

    @Override
    public Set<Object> getSingletons()
    {
        if (singletons.isEmpty()) {
            singletons.add(new EmpRest());
        }
        return singletons;
    }
}
```

Clase Servicio REST

```
@Path("/rest")
@Consumes({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_XML,MediaType.APPLICATION_JSON})
public class SampleRest
{
    private String param;

    @Context
    public void setServletContext(ServletContext context)
    {
        param = context.getInitParameter("paramName");
    }

    @POST
    @Path("/buscarPorId")
    public BuscarResult buscarPorId(BuscarArgs args)
    {
        ...
    }
}
```

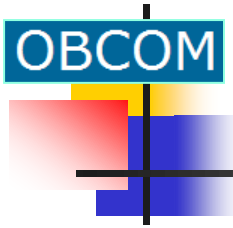


Servicios SOAP JAX-WS

- Requiere una clase “Servicio SOAP”
  - @**WebService**(targetNamespace="...")
- Cada método del servicio debe decorarse
  - @**WebMethod**(operationName="...")
- Cada parámetro de los métodos debe decorarse
  - @**WebParam**(name="...",targetNamespace="...")
- Se necesita un **DataSource** que administre las conexiones a la base de datos (importante)

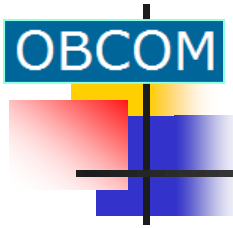
Clase Servicio SOAP

```
@WebService(targetNamespace="…")
public class SampleSoap
{
    @Resource(name="SampleDataSource")
    private DataSource dataSource;

    @WebMethod(operationName="buscarPorId")
    public BuscarResult buscarPorId(
        @WebParam(name="args") BuscarArgs args)
        throws SampleSoapException
    {
        try {
            return BuscarPorId.execute(dataSource, args);
        } catch (Exception ex) {
            logger.error("Error en SampleSoap.buscarPorId", ex);
            throw new SampleSoapException(ex);
        }
    }
}
```



Seguridad REST y SOAP

- Servicios REST y SOAP deben invocarse usando sólo protocolo **HTTPS** (**encriptado**)
  - HTTPS no permite observar los mensajes ☺
  - HTTPS no permite modificar los mensajes ☺
  - HTTPS one-way no autentica al invocador ☹
- Servicios REST y SOAP se deben **autenticar** en cada invocación
  - HTTP "Authentication: …" (REST, SOAP)
  - <Envelope><Header><Security>… (SOAP)
- Además se deben **autorizar** en cada invocación

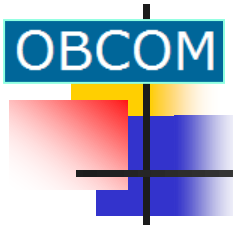


DataSource JDBC

- Conjunto de conexiones a base de datos
  - Los maneja el Application Server
  - Los define el usuario administrador
- Archivo WEB-INF/**web.xml**
  - Forma parte del estándar JEE
  - Define el **nombre interno** del DataSource
- Archivo WEB-INF/**server.xml**
  - Específico a cada Application Server
  - Define el **nombre externo** del DataSource

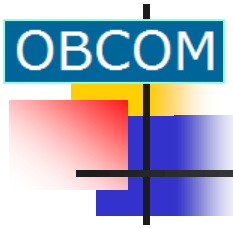
Archivo web.xml estándar

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …>

  ...

  <resource-ref>
    <res-ref-name>SampleDataSource</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
  </resource-ref>

  ...

</web-app>
```



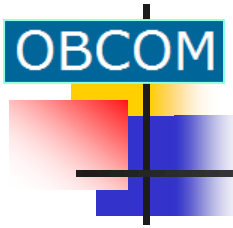
GlassFish: glassfish-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<glassfish-web-app>

  ...

  <resource-ref>
    <res-ref-name>SampleDataSource</res-ref-name>
    <jndi-name>jdbc/NombreDeAlgunDataSource</jndi-name>
  </resource-ref>

  ...

</glassfish-web-app>
```



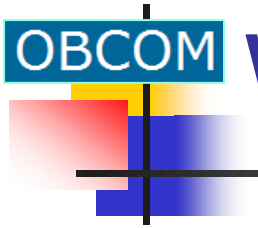
JBoss: jboss-web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>

  ...

  <resource-ref>
    <res-ref-name>SampleDataSource</res-ref-name>
    <jndi-name>java:/jdbc/NombreDeAlgunDataSource</jndi-name>
  </resource-ref>

  ...

</jboss-web>
```



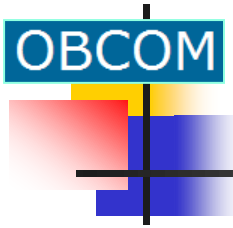
WebLogic: weblogic.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-web-app …>

  ...

  <resource-description>
    <res-ref-name>SampleDataSource</res-ref-name>
    <jndi-name>jdbc/NombreDeAlgunDataSource</jndi-name>
  </resource-description>

  ...

</weblogic-web-app>
```



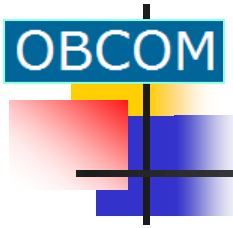
WebSphere: ibm-web-bnd.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-bnd …>

  ...

  <resource-ref name="SampleDataSource"
    binding-name="jdbc/NombreDeAlgunDataSource"/>

  ...

</web-bnd>
```



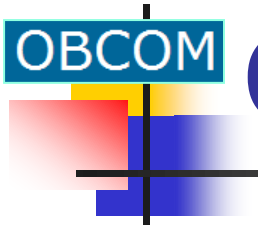
Servicios DAO/JDBC (1)

- Usar sólo API estándar JDBC
  - Clases en el paquete **java.sql**
  - Clases en el paquete **javax.sql**
- No usar API del servidor de datos
  - **MySQL**: com.mysql.cj.jdbc
  - **Oracle**: oracle.jdbc
  - **PostgreSQL**: org.postgresql.jdbc
  - **MS-SQL**: com.microsoft.sqlserver.jdbc



Servicios DAO/JDBC (2)

- No retornar status OK, ERROR, etc.
- Mejor "disparar una excepción" que se traduce por **SQLException**, la cual posee un Mensaje, SQL State y Code
- Sentencias para disparar una excepción:
  - **MYS**: SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Mensaje de error';
  - **ORA**: RAISE_APPLICATION_ERROR(-20001, 'Mensaje de error');
  - **PGS**: RAISE EXCEPTION 'Mensaje de error';
  - **SQL**: RAISERROR('Mensaje de error', 16, 1);



Clase DAO/JDBC (1)

```
public class BuscarPorId
{
    public BuscarResult execute(DataSource dataSource, BuscarArgs args)
        throws SQLException
    {
        BuscarResult result = new BuscarResult();
        try (Connection conn = dataSource.getConnection()) {
            String SQL = "{call SAMPLE$BUSCAR_POR_ID(?,?)}";
            try (CallableStatement call = conn.prepareCall(SQL)) {
                call.setInt(1, args.getId());
                call.registerOutParameter(2, Types.NVARCHAR);
                call.execute();
                …obtener ResultSets en la próxima diapositiva…
                result.setNombre(call.getString(2));
            }
        }
        return result;
    }
}
```

Clase DAO/JDBC (2)

```
                …
                try (ResultSet rset = call.getResultSet()) {
                    List<RowItem> rows = new ArrayList<>();
                    while (rset.next()) {
                        RowItem row = new RowItem();
                        row.setNombre(rset.getString("nombre"));
                        row.setSku(rset.getInt("sku"));
                        row.setMonto(rset.getBigDecimal("monto"));
                        …
                        rows.add(row);
                    }
                    result.setRows(rows);
                }
            }
        }
        return result;
    }
}
```

Muchas gracias

Muchas gracias