DEPARTAMENTO DE INFORMÁTICA
INGENIERÍA EN INFORMÁTICA.
DESARROLLO DE APLIACIONES DISTRIBUIDAS
Examen Obligatorio
20 de enero de 2009

Para la realización del examen obligatorio y optativo se dispondrá de **3 horas. NO** se podrán utilizar libros ni apuntes.
Nombre y Apellidos:
NIA:

Ejercicio 1. (1.5 puntos)

Analiza, de forma justificada, las ventajas e inconvenientes de desarrollar una aplicación distribuida en SOAP frente a una en Java RMI.

SOLUCION:

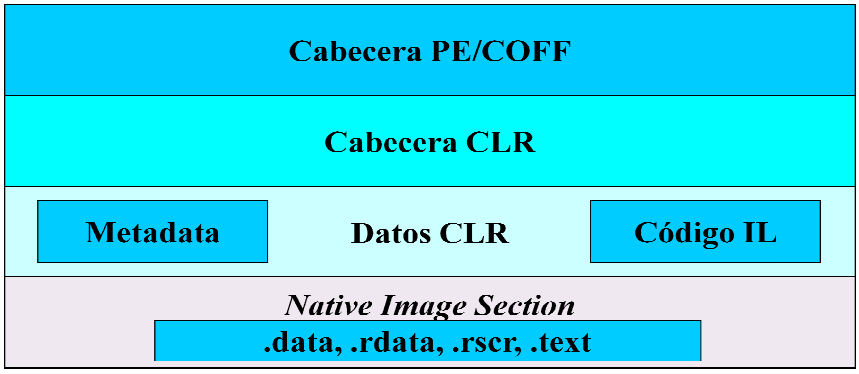
- Comparando con RMI:
  - SOAP está basado en la web (permite desarrollar aplicaciones distribuidas sobre firewalls)
  - Independencia del lenguaje.
  - Favorece la integración de servicios debido a un protocolo común.
  - El principal inconveniente es que la codificación en XML puede tener más sobrecarga (tanto de cómputo como de comunicaciones) que la empleada en Java RMI.

Ejercicio 2. (2 puntos)

Describe la estructura de un fichero *Portable Executable* (PE) en el contexto de Microsoft .Net y cómo se ejecuta.

SOLUCION:

- NET Portable Executable file:
  - Fichero de formato PE: cabecera PE/COFF estándar de Windows (*Microsoft Common Object File Format (COFF).*
  - Cabecera de CLR.
  - Datos del CLR: incluye el código IL (con la lógica de la aplicación)los metadatos (con información extra como declaración de interfaces, herencia y datos contextuales del ejecutable).
  - Sección de datos del ejecutable.
  - La diferencia en la ejecución de un PE respecto a un ejecutable (EXE) está en el programa cargador: en el caso del PE, el programa cargador envía la ejecución al entorno CLR. Esto permite ejecutar el fichero PE en cualquier plataforma con soporte a Microsoft .NET.



Ejercicio 3. (1.5 puntos)

¿Qué es la *Global Assembly Cache* (GAC) en el contexto de Microsoft .Net.?

SOLUCION:

Se trata de un repositorio de librerías que gestiona componentes y librerías desarrolladas en entorno .Net. CLR incluye mecanismos con políticas de versiones mediante el acceso a los metadatos de los assemblies. De este modo, permite gestionar distintas versiones del mismo componente y acceder al mismo como una librería dinámica.

Ejercicio 4. (3 puntos)

Dado el siguiente código de .Net Remoting se pide:

1. ¿Qué es un *delegate* en el contexto de Microsoft .Net.? ¿En qué se utilizan en el ejemplo?
2. ¿Qué implicaciones tiene la herencia de la línea L1? ¿Por qué es necesario utilizarla?
3. ¿Qué significa el atributo de la línea L2?
4. ¿Qué información contiene el fichero descrito en la línea L3?
5. Describe qué es el método *method1* descrito en la línea L4
6. Indica qué modificaciones debe realizar en el código para que el método remoto invocado por el cliente reciba dos argumentos de tipo *string*.

```
L1 public class RemotingDelegates : MarshalByRefObject{
    public delegate string RemoteAsyncDelegate();

L2 [OneWayAttribute]
   public void OurRemoteAsyncCallBack(IAsyncResult ar){
      RemoteAsyncDelegate del = (RemoteAsyncDelegate)((AsyncResult)ar).AsyncDelegate;
      Console.WriteLine("\r  del.EndInvoke(ar) );
        return;
   }

    public static void Main(string[] Args){
        RemotingDelegates HandlerInstance = new RemotingDelegates();
        HandlerInstance.Run();
    }


       public void Run(){
L3       RemotingConfiguration.Configure("SyncAsync.exe.config");
         ServiceClass obj = new ServiceClass();
         AsyncCallback method2 = new AsyncCallback(this.OurRemoteAsyncCallBack);
L4       RemoteAsyncDelegate delegate1 = new RemoteAsyncDelegate(obj.method1);
         IAsyncResult RemAr = delegate1.BeginInvoke(method2, null);

         while(!RemAr.IsCompleted){
             Thread.Sleep(1);
          }
         }
}
```

Solución

1. Los delegates son equivalentes a punteros a funciones y se emplean para poder realizar el callback asíncrono a funciones.
2. Permite ejecutar de forma remota el delegate (junto con el método remoto que tiene asociado)
3. El método remoto se ejecuta de forma no bloqueante.

4. Contiene la descripción del método remoto del servidor (nombre del servidor, número de puerto y etiqueta del método remoto).
5. Es el método remoto del servidor.
   Hay que añadir los argumentos en el método **`delegate1.BeginInvoke`** y en la declaración del delegate **`(public delegate string RemoteAsyncDelegate();)`**

Ejercicio 5. (2 puntos)

En el contexto de *Cloud Computing* se pide:

1. Indica los requisitos fundamentales que requiere una arquitectura de *Cloud Computing.*
2. Cita un servicio de *Cloud Computing* que podemos encontrar en el mercado.
3. ¿Qué características deben de tener las aplicaciones que se ejecutan en *Cloud Computing*?

Solución

1. Requiere de una infraestructura global de comunicaciones (internet), soporte a la virtualización y una infraestructura para ofrecer recursos de cómputo como servicios.
2. El EC2: Elastic Compute Cloud que oferta cómputo o el S3: Simple Storage Service que oferta almacenamiento.
3. Requiere de aplicaciones levemente acopladas (pocas comunicaciones) con acceso intensivo a los datos.