

ENS IV – Tasca 02

Autor: Sergio Blanco Cuaresma

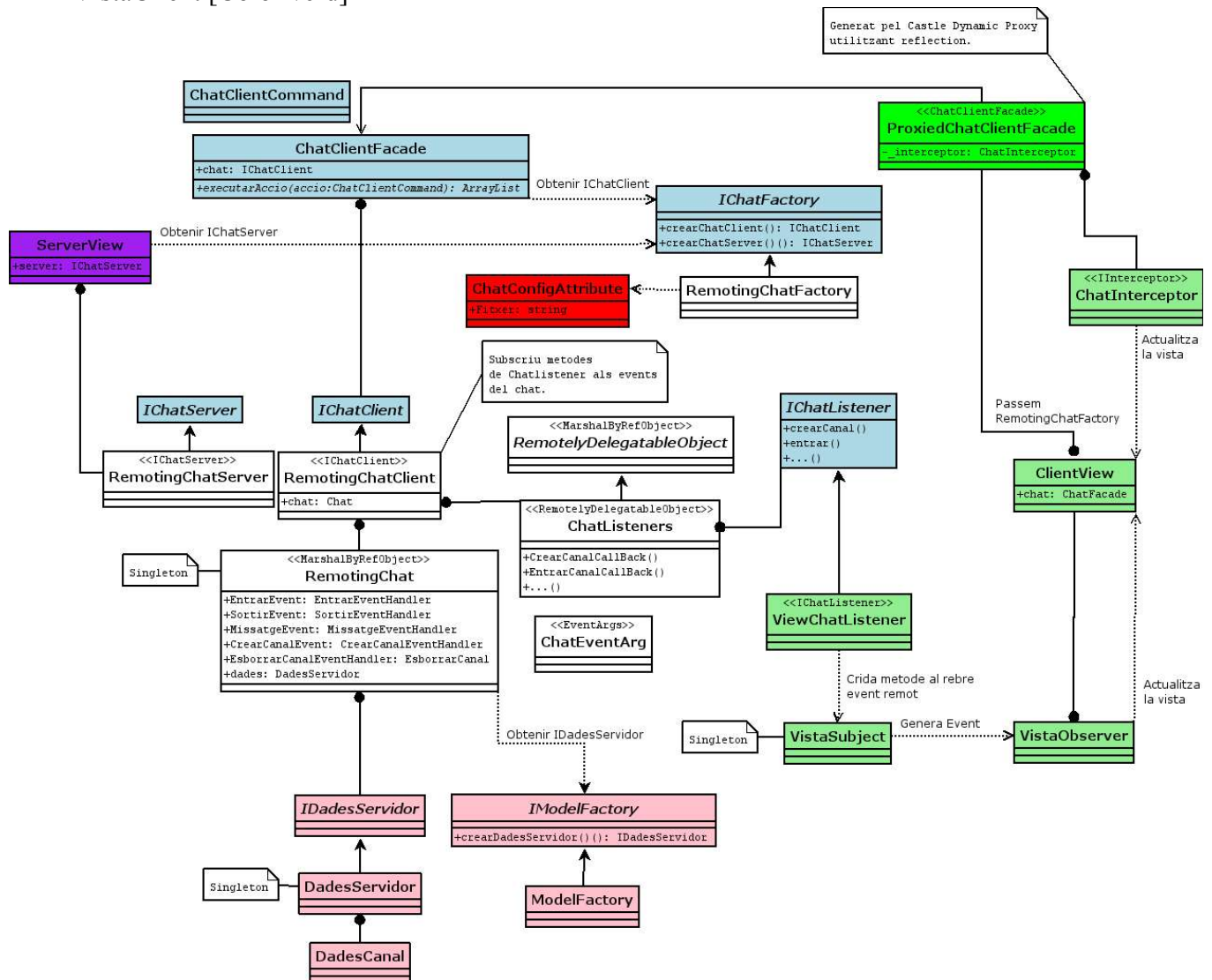
Enunciat

Modificar l'exercici 1 del Xat en .NET Remoting per tal que utilitzi:

- Custom Attributes.
- Dynamic Proxies.

Disseny

- Controlador: Interfícies d'accés al client i servidor chat. [Color Blau]
- Remoting: Implementació específica del servidor i client chat. [Color Blanc]
- Model: Classes necessàries per guardar la informació del chat a memòria. [Color Rosa]
- VistaServidor [Color Lila]
- VistaClient [Color verd]



Implementació

Custom Attributes

A la implementació específica per Remoting del Factory (`RemotingChatFactory`) que genera les interfícies d'accés pel servidor y client, s'ha afegit un atribut personalitzat anomenat "ChatConfigAttribute":

```

using System;
using System.Reflection;

namespace Chat.Remoting
{
    [AttributeUsage(AttributeTargets.Method)]
    public class ChatConfigAttribute:Attribute
    {
        private string fitxer;
        public string Fitxer
        {
            get
            {
                return fitxer;
            }
            set
            {
                this.fitxer = fitxer;
            }
        }

        public ChatConfigAttribute(String fitxer)
        {
            this.fitxer = fitxer;
        }
    }
}

```

Aquest atribut s'utilitzarà per carregar un fitxer de configuració per defecte, en cas de que no s'hagi especificat cap a la instanciació del Factory:

```

[ChatConfig("../RemotingClient.config")]
public IChatClient crearChatClient()
{
    // Obtenim atribut
    MethodInfo metode = typeof(RemotingChatFactory).GetMethod("crearChatClient");
    object [] attributes = metode.GetCustomAttributes (typeof(ChatConfigAttribute), false);
    //if (attributes != null && attributes.Length > 0)
    //{
    ChatConfigAttribute chatConfig = (ChatConfigAttribute)attributes[0];
    //}

    // Si no s'ha especificat un fitxer de configuració, agafem el fitxer per defecte de l'atribut
    if ((fitxerConfig == null) || (fitxerConfig.CompareTo("") == 0))
    {
        this.fitxerConfig = chatConfig.Fitxer;
    }
    else
    {
        chatConfig.Fitxer = this.fitxerConfig;
    }

    return new RemotingChatClient(fitxerConfig, listeners);
}

```

Dynamic Proxies

Les possibilitats per realitzar proxies en .NET son molt bones, però aquestes impliquen

modificacions directes al nostre codi (e.g. heretar de MarshallByRef o ContextBoundObject), afectant al disseny jeràrquic de la nostra aplicació. Per aquest motiu s'ha utilitzat la implementació Castle's Dynamic Proxy:

<http://www.castleproject.org/index.php/DynamicProxy>

Amb aquesta implementació es podran crear interceptors/proxies sense haver de modificar el disseny del nostre aplicatiu.

A la aplicació del chat, crearem un proxy a la VistaClienta i encapsularà les funcions del ChatClientFacade. L'objectiu serà mostrar en la finestra client, totes les comandes que la nostra interfície executa a través del ChatClientFacade (com si fos un LOG de l'aplicació).

Castle's Dynamic Proxy necessita un interceptor que s'encarregarà d'afegir funcionalitat al nostre proxy, en el nostre cas he implementat ChatInterceptor:

```
using System;
using System.Collections;
using System.Windows.Forms;
using Castle.DynamicProxy;
using Chat.Controlador;

namespace Chat.VistaClient
{
    /// <summary>
    /// Descripción breve de ChatInterceptor.
    /// </summary>
    public class ChatInterceptor : IInterceptor
    {
        private FormVistaClient window;

        public ChatInterceptor(FormVistaClient window)
        {
            this.window = window;
        }

        public object Intercept(IInvocation invocation, params object[] args)
        {
            if (invocation.Method.Name.Equals("sendCommand"))
            {
                if ((args != null) && (args.Length > 0))
                {
                    ChatClientCommand command = args[0] as
                        ChatClientCommand;
                    if (command != null)
                    {
                        switch(command.Accio)
                        {
                            [...]
                        }
                    }
                }
            }

            // Crida al objecte original
            object retValue = invocation.Proceed( args );
        }
    }
}
```

```

        if (retValue != null)
        {
            ArrayList resultat = retValue as ArrayList;
            if (resultat != null)
            {
                string log = "Retornat: ";
                foreach(string cadena in resultat){
                    log = log + cadena + " ";
                }
                window.escriureLog(log);
            }
        }
        return retValue;
    }
}

```

Al FormVistaClient he afegit un mètode per generar el proxy:

```

private ChatClientFacade obtenirInterficieChatClient(IChatFactory chatFactory)
{
    ProxyGenerator generator = new ProxyGenerator();

    object[] args = new object[1];
    args[0] = chatFactory;
    ChatInterceptor interceptor = new ChatInterceptor(this);

    ChatClientFacade proxiedChat = (ChatClientFacade)generator.CreateClassProxy(typeof(
                                                                    ChatClientFacade),interceptor, args);

    return proxiedChat;
}

```

Aquest s'utilitzarà al connectar amb el servidor:

```

// ChatSubject es singleton
vistaListeners = new VistaListeners(); // Modificara el ChatSubject
vistaObserver = new VistaObserver(this); // Consultara el ChatSubject
IChatFactory chatFactory = new RemotingChatFactory(this.textBoxRemotingConfig.Text, vistaListeners);

//chatClientFacade = new ChatClientFacade(chatFactory);
chatClientFacade = obtenirInterficieChatClient(chatFactory);

```

Castle's Dynamic Proxy generarà utilitzant reflection, un objecte que hereta de ChatClientFacade i que conté el nostre ChatInterceptor. Cal destacar, que es necessari que per capturar un mètode de ChatClientFacade, aquest cal que sigui “virtual” donat que Castle's Dynamic Proxy funciona per herència i no genera codi de forma individual per a cada crida de mètode (aconseguint així major eficiència).

S'ha afegit una nova caixa de text a l'aplicació gràfica del client per tal de visualitzar el LOG que es va generant:

