"I invented clean underwear – in case he gets into an accident."

Reuso, Librerías y Frameworks

Federico Balaguer

Reuso

Ventajas

- Menor esfuerzo
- Menor tiempo de desarrollo
- Menor incertidumbre
- Mayor consistencia
- Mayor confiabilidad
    - Funcionó para n casos anteriores
- Promueve trabajo en equipo
- Adquisición tecnológica
    - Llave en mano
    - Know-how

Desventajas

- Requiere mantener configuraciones de software (versiones de dependencias)
- Riesgo de dependencias no deseadas
- Curva de aprendizaje
- Código externo
    - Dependencia tecnológica
    - Resistencia a adopción
    - Riesgo de Seguridad
- Desarrollo de código reusable
    - Requiere tiempo
    - Requiere conocimiento detallado
        - Basado en casos
        - Experto en el tema

El reuso es inesperado

Modos de reuso

Copy
&
Paste

Herencia
&
Polimorfismo

Extraer
&
Delegar

Librerías

Frameworks

Sofisticación

Reuso

Librería OO

- Se reusa código
- (OO) ⇒ Conjunto de clases
  - Funcionalidad
    - Instancias
    - Clase (Estática)
  - El sistema/módulo
    - Crea instancias
    - Invoca la funcionalidad

Framework

- Se reusa
  - Una manera de ejecutar
  - Una manera de instanciar
  - Una manera de extender
- Es incompleto. No hace nada concreto
- (OO) ⇒ Conjunto de clases
  - Framework controla la ejecución (execution thread)
    - Inversión de control
  - Cookbook: Reglas de uso
    - Instanciación: implementar aplicación
      - White-box: thread incompleto
      - Black-box: thread configurable
    - Extensión: agregar opciones
  -

Librerias & Frameworks

Definición

Un framework es una aplicación “semi-completa”, “reusable”, que puede ser especializada para producir aplicaciones a medida…

…un conjunto de clases concretas y abstractas, relacionadas para proveer una

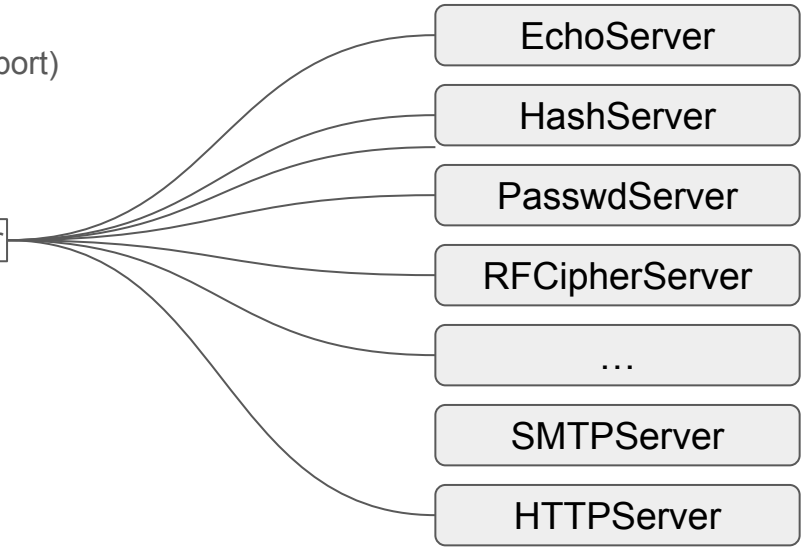
arquitectura reusable que implementa una familia de aplicaciones (relacionadas)…

Servidores TCP

- Es un programa que implementa una de las partes de la arquitectura Cliente/Servidor
  - Un Servidor escucha en un socket TCP (IP_address+port)
  - Un Cliente establece una conexión con el servidor para enviar mensajes (plain text)
  - El Servidor responde
- Ejemplos:
  - EchoServer:
    - Recibe un mensaje de texto
    - Responde el mensaje que el cliente envió
  - SMTPServer: emails
  - DayTimeServer: fecha actual
  - HTTPServer: envío y recepción de documentos web

Flujo de Control

1. Crear un socket (IP_address +port)
2. Escucha en el socket
3. Aceptar al cliente ⇒ Sesión
4. Recibir un mensaje
    a. Condición ⇒ Responder
    b. !Condición ⇒
        i. Cerrar sesión
        ii. Goto 2



- Reusar el “flujo de control”
    - El flujo de control es incompleto (generalmente no compila)
    - “Responder” es donde se puede “enganchar” la funcionalidad
    - “Responder” es un hook (obligatorio)
- El flujo de control pertenece a un framework

SimpleThreadTCPServer

Flujo de Control (incompleto)

1. Crear un socket (IP_address +port)
2. Escucha en el socket
3. Aceptar al cliente ⇒ Sesión
4. Recibir un mensaje
    a. Condición ⇒ **Responder**
    b. !Condición ⇒ Cerrar sesión

Extensión

1. No existen puntos de extensión

Ejemplo: VoidServer, EchoServer

Instanciación (Cookbook)

1. Subclasificar SimpleThreadTCPServer
    a. Debe implementar Main(String[])
        i. crear una instancia
        ii. enviar método startLoop(String[])
    b. Debe implementar handleMessage(String)
    c. Métodos que podría implementar
        i. acceptAndDisplaySocket(ServerSocket)
        ii. checkArguments(String[])
        iii. displayAndExit(int)
        iv. displaySocketData(Socket)
        v. displayUsage()
        vi. displaySocketInformation()
        vii. displayWarning()

SimpleThreadTCPServer

Preguntas de examen:

1) Existen hooks?
2) Identifica algún patrón? Cual?

```
12      public final void startLoop(String[] args) {
13          checkArguments(args);
14
15          int portNumber = Integer.parseInt(args[0]);
16
17
18          try (ServerSocket serverSocket = new ServerSocket(portNumber)) {
19              displaySocketInformation(portNumber);
20              while (true) {
21                  Socket clientSocket = acceptAndDisplaySocket(serverSocket);
22                  handleClient(clientSocket);
23              }
24          } catch (IOException e) {
25              displayAndExit(portNumber);
26          }
27      }
```

1. Todos los servidores pueden redefinir:
   a. checkArguments(), displaySocketInformation() y displayAndExit()

SimpleThreadTCPServer.handleClient(Socket)

```
62      private final void handleClient(Socket clientSocket) {
63
64          try (
65                  PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
66                  BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));) {
67              String inputLine;
68              while ((inputLine = in.readLine()) != null) {
69                  System.out.println("Received message: " + inputLine + " from "
70                          + clientSocket.getInetAddress().getHostAddress() + ":" + clientSocket.getPort());
71                  handleMessage(inputLine, out);
72                  if (inputLine.equalsIgnoreCase("")) {
73                      break; // Client requested to close the connection
74                  }
75              }
76              System.out.println("Connection closed with " + clientSocket.getInetAddress().getHostAddress() + ":"
77                      + clientSocket.getPort());
78          } catch (IOException e) {
79              System.err.println("Problem with communication with client: " + e.getMessage());
80          } finally {
81              try {
82                  clientSocket.close();
83              } catch (IOException e) {
84                  System.err.println("Error closing socket: " + e.getMessage());
85              }
86          }
87      }
88  }
89
```

1. Todos los servidores debern definir:
   a. handleMessage(String, PrintWriter)

Preguntas de examen:

1) Existen hooks?
2) Identifica algún patrón? Cual?

EchoServer.java

```java
import java.io.PrintWriter;

public class EchoServer extends SingleThreadTCPServer {

    public void handleMessage(String message, PrintWriter out) {
        out.println(message);
    }

    Run | Debug
    public static void main(String[] args) {

        new EchoServer().startLoop(args);

    }
}
```

1. startLoop(args) arranca el loop de control del framework
    a. El Framework toma el control (**inversión de control**)

EchoServer.java

```java
import java.io.PrintWriter;

public class EchoServer extends SingleThreadTCPServer {

    public void handleMessage(String message, PrintWriter out) {
        out.println(message);
    }

    Run | Debug
    public static void main(String[] args) {

        new EchoServer().startLoop(args);

    }
}
```

1. startLoop(args) arranca el loop de control del framework
   a. El Framework toma el control (**inversión de control**)
   b. Ahora está completo porque se implementa handleMessage(...)
2. handleMessage(...) es invocado desde alguna parte del framework
   a. Completa el loop de control
   b. “Hollywood Principle”: no nos llames, nosotros te contactaremos
   c. EchoServer hereda el loop de control no existe encapsulamiento ⇒ es una **Caja Blanca**
   d. La ejecución del server depende de la correcta implementación de handleMessage() porque es parte del loop (incompleto) de control

Framework de Caja Blanca (white box framework)

- La instanciación hereda y completa el loop de control
- Es posible que requiera agregar métodos a clases del framework
- Demanda conocimiento del código del framework

⇒ es una Caja Blanca [1]

(1) Designing Reusable Clases. B. Foote & R. Johnson

HotSpots vs FrozenSpot

FrozenSpot: aspecto del framework que afecta a todas las instanciaciones y que no se puede modificar (marca indeleble)

HotSpot: estructura en el código que permite modificar el comportamiento del framework, para instanciar y para extender.

Ej: SingleThreadTCPServer cierra la conexión con el cliente si:

1. El cliente cierra la conexión
2. El cliente envía un mensaje vacío [1]

```
72              if (inputLine.equalsIgnoreCase("")) {
73                  break; // Client requested to close the connection
74              }
```

Ejercicio: como podría convertir ese FrozenSpot en un HotSpot?

(1) SingleThreadTCPServer.java: 72-74

Resumen de Frameworks

- Proveen una solución reusable para una familia de aplicaciones
- Las clases en el framework se relacionan (herencia, conocimiento, envío de mensajes) de manera que resuelven la mayor parte del problema en cuestión
- El código del framework controla/usa al código de la instanciación
- Tipos de Frameworks
  - Aplicación: desktop, webapps, tcpservers :-)
  - Manejo Datos: ORDB, pipelines, NRDB
  - Sistemas Distribuidos: mensajes, eventos, rpc
  - Testing: unit, web pages
- oo2.next()
  - Blackbox frameworks: instanciación por composición
  - Frameworks & Design Patterns


while (not edge) {
run();
}
do {
run();
} while (not edge);