DAI - Développement d'applications internet

Leonard Cseres | December 27, 2024

GitHub Platforme et service cloud pour le dev. de logiciels et le contrôle de version utilisant Git, permettant aux développeurs de stocker et de gérer leur code.

SSH Plus sécurisé et il n'y a pas besoin de s'authentifier à chaque fois par rapport à **HTTPS**.

Maven Composé de phases et de goals. Les phases chargent les goals, qui exécutent des tâches projet (par exemple, compiler, tester, empaqueter).

POM Project Object Model, XML qui contient des infos sur le projet et configuration de Maven.

MVNW Wrapper Maven, permet d'avoir une version de Maven spécifique tous les collaborateurs du projet et éviter des problèmes de compatibilité.

`maven-jar-plugin` Plugin Maven pour créer un fichier JAR. `maven-shade-plugin` Plugin Maven pour créer un fichier JAR **fat** (contient toutes les dépendances).

```
# Télécharge les dépendances & transitives
./mvnw dependency:go-offline
# Supprime les classes compilées
./mvnw clean
# Compile le code source
./mvnw compile
# Emballe l'application dans un fichier JAR
./mvnw package
# Plusieurs phases
./mvnw ...
```

Java Portable grâce à la JVM, orienté objet, multi-threadé, fortement typé, compilé en byte-code. (SDKMAN! utilisé pour gérer les versions)

JAR Java ARchive, contient des fichiers .class et des métadonnées.

ASCII 7 bits, 128 caractères.

Extended ASCII 8 bits, 256 caractères.

Unicode UTF-8, UTF-16, UTF-32, standard pour tous les caractères.

UTF-8 8 bits, 1 à 4 octets, compatible ASCII.

```
import java.io.*;
// Binary
InputStream; OutputStream; FileInputStream; FileOutputStream;
BufferedInputStream; BufferedOutputStream;
// Text
Reader; Writer; FileReader; FileWriter;
BufferedReader; BufferedWriter;
import java.nio.charset.StandardCharsets;

// Exception
IOException; UnsupportedEncodingException; FileNotFoundException;
```

```
// Binary Read
String filename;

try (InputStream fis = new FileInputStream(filename)) {
  int b;
  while ((b = fis.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Write
String filename;
int sizeInBytes;

try (OutputStream fos = new FileOutputStream(filename)) {
  for (int i = 0; i < sizeInBytes; i++) {
    fos.write(1);
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Buffered Read
String filename;

try (InputStream fis = new FileInputStream(filename);
    BufferedInputStream bis = new BufferedInputStream(fis)) {
  int b;
  while ((b = bis.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Binary Buffered Write
String filename, int sizeInBytes;

try (OutputStream fos = new FileOutputStream(filename);
    BufferedOutputStream bos = new BufferedOutputStream(fos)) {
  for (int i = 0; i < sizeInBytes; i++) {
    bos.write(1);
  }
  bos.flush();
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Read
String filename;

try (Reader reader = new FileReader(filename, StandardCharsets.UTF_8)) {
  int b;
  while ((b = reader.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Write
String filename, int sizeInBytes;

try (Writer writer = new FileWriter(filename, StandardCharsets.UTF_8)) {
  for (int i = 0; i < sizeInBytes; i++) {
    writer.write('a');
  }
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Buffered Read
String filename;
try (Reader reader = new FileReader(filename, StandardCharsets.UTF_8);
    BufferedReader br = new BufferedReader(reader)) {
  int b;
  while ((b = br.read()) != -1) {
    // Do nothing - simulate processing
  }
} catch (java.io.IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

```
// Text Buffered Write
String filename, int sizeInBytes;

try (Writer writer = new FileWriter(filename, StandardCharsets.UTF_8);
    BufferedWriter bw = new BufferedWriter(writer)) {
  for (int i = 0; i < sizeInBytes; i++) {
    bw.write('a');
  }
  bw.flush();
} catch (IOException e) {
  System.err.println("Error: " + e.getMessage());
}
```

Docker *Bare metal* software runs directly on hardware, *virtualization* software runs on a virtual machine, *containerization* software runs in a container.

Image read-only template for container creation

Container runnable instance of an image

Registry service storing images

DNS

- **NS:** Name server
- **CNAME:** Alias
- **A:** IPv4 addr.
- **AAA:** IPv6 addr.

Ports Unsigned 16-bit, 0-1023 reserved

TCP

```java
public class TCPClient {
  public static void main(String[] args) {
    String serverAddr = "127.0.0.1";
    int port = 1234;

    try (
      Socket socket = new Socket(serverAddr, port);
      InputStreamReader isr =
          new InputStreamReader(socket.getInputStream(),
                                StandardCharsets.UTF_8);
      BufferedReader in = new BufferedReader(isr);
      OutputStreamWriter osw =
          new OutputStreamWriter(socket.getOutputStream(),
                                 StandardCharsets.UTF_8);
      BufferedWriter out = new BufferedWriter(osw)) {
      // connected
      out.write("Hello, Server!\n");
      out.flush();
      String response = in.readLine();
      if (response == null) {
        // server diconnected
      } else {
        // response
      }
    } catch (IOException e) {}
  }
}
```

```java
public class TCPServer {
  public static void main(String[] args) {
    int port = 1234;
    int nbThreads = 2;

    try (
      ServerSocket serverSocket = new ServerSocket(port);
      ExecutorService e = Executors.newFixedThreadPool(nbThreads)) {
      // server started
      while (true) {
        Socket socket = serverSocket.accept();
        e.submit(new ClientHandler(socket));
      }
    } catch (IOException e) {}
  }

  static class ClientHandler implements Runnable {
    private final Socket socket;
    public ClientHandler(Socket socket) { this.socket = socket; }

    @Override
    public void run() {
      try (
        InputStreamReader isr =
            new InputStreamReader(socket.getInputStream(),
                                  StandardCharsets.UTF_8);
        BufferedReader in = new BufferedReader(isr);
        OutputStreamWriter osw =
            new OutputStreamWriter(socket.getOutputStream(),
                                   StandardCharsets.UTF_8);
        BufferedWriter out = new BufferedWriter(osw)) {
        // client connected
        String request = in.readLine();
        if (request != null) {
          out.write("Echo: " + request + "\n");
          out.flush();
        } else {
          // client diconnected
        }
      } catch (IOException e) {}
    }
  }
}
```

UDP Unicast

```java
public class UDPClient {
  public static void main(String[] args) {
    String serverAddr = "127.0.0.1";
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket()) {
      byte[] buffer = "Hello, Server!".getBytes();
      InetAddress address = InetAddress.getByName(serverAddr);

      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length, address, port);
      socket.send(packet);

      buffer = new byte[1024];
      packet = new DatagramPacket(buffer, buffer.length);
      socket.receive(packet);

      String response = new String(
        packet.getData(), 0, packet.getLength());
      System.out.println("Server Response: " + response);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```java
public class UDPServer {
  public static void main(String[] args) {
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket(port)) {
      System.out.println("Listening on port " + port);

      while (true) {
        byte[] buffer = new byte[1024];
        DatagramPacket packet = new DatagramPacket(
          buffer, buffer.length);
        socket.receive(packet);

        String msg = new String(
          packet.getData(), 0, packet.getLength());
        System.out.println("Received: " + msg);

        InetAddress clientAddr = packet.getAddress();
        int clientPort = packet.getPort();
        buffer = ("Echo: " + msg).getBytes();

        packet = new DatagramPacket(
          buffer, buffer.length, clientAddr, clientPort);
        socket.send(packet);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

UDP Multicast

```java
public class UDPMulticastSender {
  public static void main(String[] args) {
    String multicastGroup = "230.0.0.1";
    int port = 1234;

    try (DatagramSocket socket = new DatagramSocket()) {
      byte[] buffer = "Hello, Group!".getBytes();
      InetAddress group = InetAddress.getByName(multicastGroup);

      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length, group, port);
      socket.send(packet);

      System.out.println("Message sent to group");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```java
public class UDPMulticastReceiver {
  public static void main(String[] args) {
    String multicastGroup = "230.0.0.1";
    int port = 1234;

    try (MulticastSocket socket = new MulticastSocket(port)) {
      InetAddress group = InetAddress.getByName(multicastGroup);
      socket.joinGroup(group);

      System.out.println("Joined group");
      byte[] buffer = new byte[1024];
      DatagramPacket packet = new DatagramPacket(
        buffer, buffer.length);

      while (true) {
        socket.receive(packet);
        String msg = new String(
          packet.getData(), 0, packet.getLength());
        System.out.println("Received: " + msg);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }

}
```

Concurrency

```java
@Override
public Integer call() {
```

```
  try (ExecutorService executorService =
  Executors.newFixedThreadPool(2);) {
    executorService.submit(this::emittersWorker);
    executorService.submit(this::operatorsWorker);
  } catch (Exception e) {
    System.out.println("[Receiver] Exception: " + e);
    return 1;
  }
  return 0;
}
```

Protocole Applicatif

1. **Aperçu:** Quel est le but?
2. **Protocole de transport:** Quel protocol? Quel port? Encodage? Délimiteur? Messages texte ou binaire. Qui initialise la communication?
3. **Messages:** Foncitionalité, requête et réponses.
4. **Examples:**

TODO: Dockerfile examples

TODO: SECTIONS APPLICATION PROTOCOL DESCRIPTION

TODO: MORE CONCURRENCY

TODO: SMTP