Сценарії збирання Java-проєктів. Візуальний UI в Java.

Apache ant

Apache Ant (англ. *ant* — мураха і водночас акронім — «Another Neat Tool») — java-утиліта для автоматизації процесу збирання програмного продукту.

Ant — платформонезалежний аналог UNIX-утиліти make, але з використанням мови Java, він вимагає платформи Java, і краще пристосований для Java-проєктів. Найпомітніша безпосередня різниця між Ant та Make те, що Ant використовує XML для опису процесу збирання і його залежностей, тоді як Make має свій власний формат Makefile. За умовчанням XML-файл називається build.xml.

Ant був створений в рамках проєкту Jakarta, сьогодні — самостійний проєкт першого рівня Apache Software Foundation.

Apache ant

Перша версія була розроблена інженером Sun Microsystems Джеймсом Девідсоном (James Davidson), який розробляв першу референтну реалізацію J2EE.

Перші версії використовувалися для збирання Java-Web-сервера Tomcat.

Apache Ant використовує XML. Файл, зазвичай називається build.xml. Можна використовувати самостійно, встановивши і запускаючи з командної стрічки. Або інтегрувати в IDE.

В NetBeans використовується за замовчуванням файли

/nbproject/ **build-impl.xml** (Net Beans виносить сурс з build.xml)

/nbproject/ **project.xml** (задає папки з кодом)

Приклади тегів Ant

<project name="CollectionExample" default="default" basedir=".">

Таргети - завдання для збірки, розрізняються по **name**, виконуються послідовно (наслідуються) послідовність задається через **depends**, можуть мати **description** - виводиться в довідці (NetBeans не ставить)

```
 <target depends="-pre-init,-init-private" name="-init-user">
        <!-- The two properties below are usually overridden -->
                <property name="default.javac.source" value="1.4"/>
        <property name="default.javac.target" value="1.4"/>
</target>
```

Проекти Intellij Idea

Intellij Idea Має власний сценарій збірки проекту. Зберігається в файлі <НазваПроекту>.iml в форматі xml.

До проекту можна підключити файли Apache Ant додатково.

Керування структурою проекту і сценарієм збирання проекту в Intellij Idea виконується через вікно Project Structure.

Щоб зібрати Jar-файл потрібно додати артифакт в проект (Project Structure -> Artifacts -> “+” -> JAR -> З модуля залежностей -> Задати папку) потім збилдити проект через меню.

Apache Maven

«Apache Maven» — це засіб автоматизації роботи з програмними проєктами, який спочатку використовувався для Java проєктів. Використовується для управління (management) та складання (build) програм. Створений 2002 року Джейсоном ван Зилом. За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML.

XML-файл описує проєкт, його зв'язки з зовнішніми модулями і компонентами, порядок будування (build), папки та необхідні плагіни. Сервер із додатковими модулями та додатковими бібліотеками розміщується на серверах.

Раніше Maven був частиною *Jakarta Project*.

Apache Maven

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

Використовується чітка структура папок.

Процес збирання записується в форматі xml в файлі pom.xml

В IDEA можна додати підтримку Maven.

Теги POM.XML

project – кореневий елемент файлу

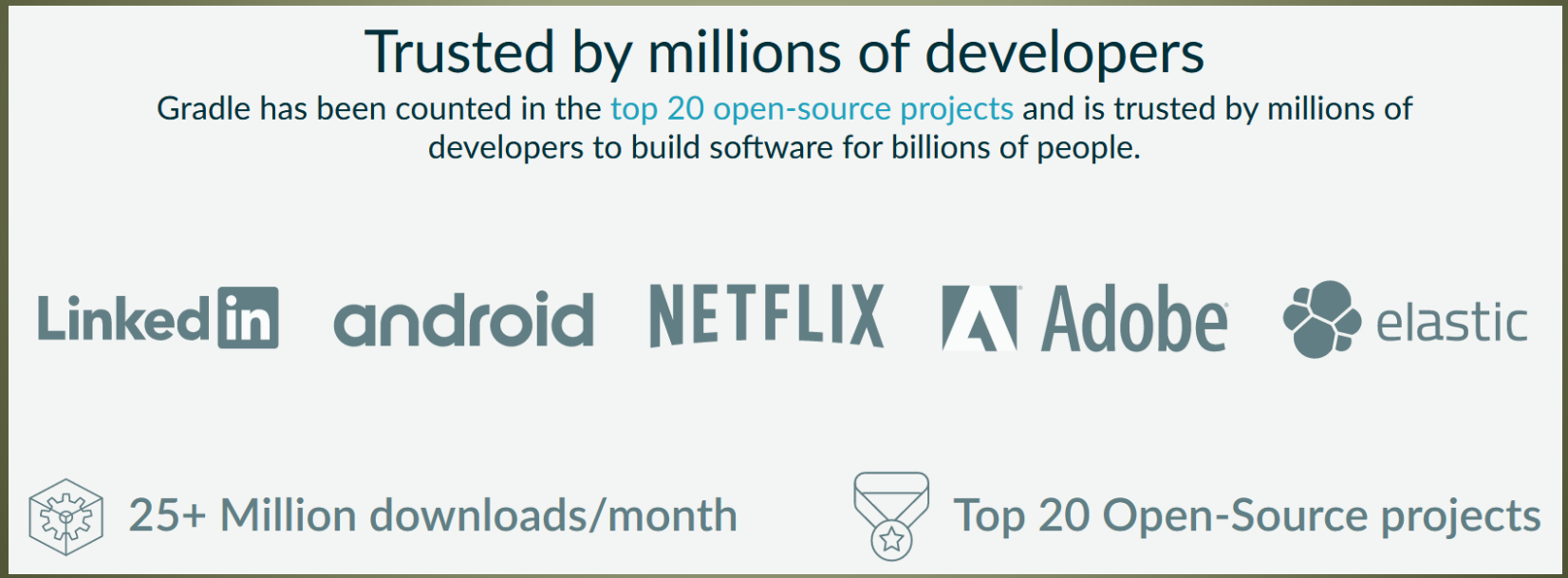
dependencies – розділ підключення сторонніх бібліотек

plugins – розділ підключених плагінів

build – розділ збирання проектів, включає розділ плагінів і команди зборки проекту

Gradle Build Tool

https://gradle.org/



Gradle Build Tool

Gradle — система автоматичного збирання, що вдосконалює принципи, закладені в Apache Ant та Apache Maven але використовує предметно-орієнтовану мову (DSL) на основі мови Groovy замість традиційної XML-подібної форми представлення конфігурації проєкту. Для визначення порядку виконання завдань Gradle використовує орієнтований ациклічний граф ("DAG").
Gradle було розроблено для побудови мультипроєктів, які можуть розростатися, і підтримує інкрементне збирання. Вона визначає, які частини було змінено, і виконує тільки ті задачі, які залежать від цих частин.

Приклади команд Gradle

```
plugins {
    id 'java’
    id 'application’
    id 'org.javamodularity.moduleplugin' version '1.8.12’
    id 'org.openjfx.javafxplugin' version '0.0.13’
    id 'org.beryx.jlink' version '2.25.0’}

application {
    mainModule =
     'com.example.fxgraddlesample'
    mainClass =
     'com.example.fxgraddlesample.HelloApplication'}
```

Java Swing

Swing — це бібліотека Java Foundation Classes [JFC] і розширення Abstract Window Toolkit [AWT]. Java Swing пропонує значно покращену функціональність порівняно з AWT, нові компоненти, розширені функції компонентів і відмінну обробку подій із підтримкою перетягування.

Відповідно до сучасних вимог графічного інтерфейсу додатків, AWT є обмеженою реалізацією, яка не зовсім здатна забезпечити компоненти, необхідні для розробки складних графічних інтерфейсів, необхідних для сучасних комерційних програм. Набір компонентів AWT має досить багато помилок і займає багато системних ресурсів порівняно з еквівалентними ресурсами Swing.

Java Swing

```
import javax.swing.*;
public class MyWindow {
    public static void main(String[] args) {
        JFrame frm = new JFrame("First window");
        frm.setSize(300, 200);
        frm.getContentPane()
        .add(new JLabel("Hello, Window!"));
        frm.setDefaultCloseOperation(
                JFrame.EXIT_ON_CLOSE);
        frm.setLocationRelativeTo(null);
        frm.setVisible(true);
    }
}
```

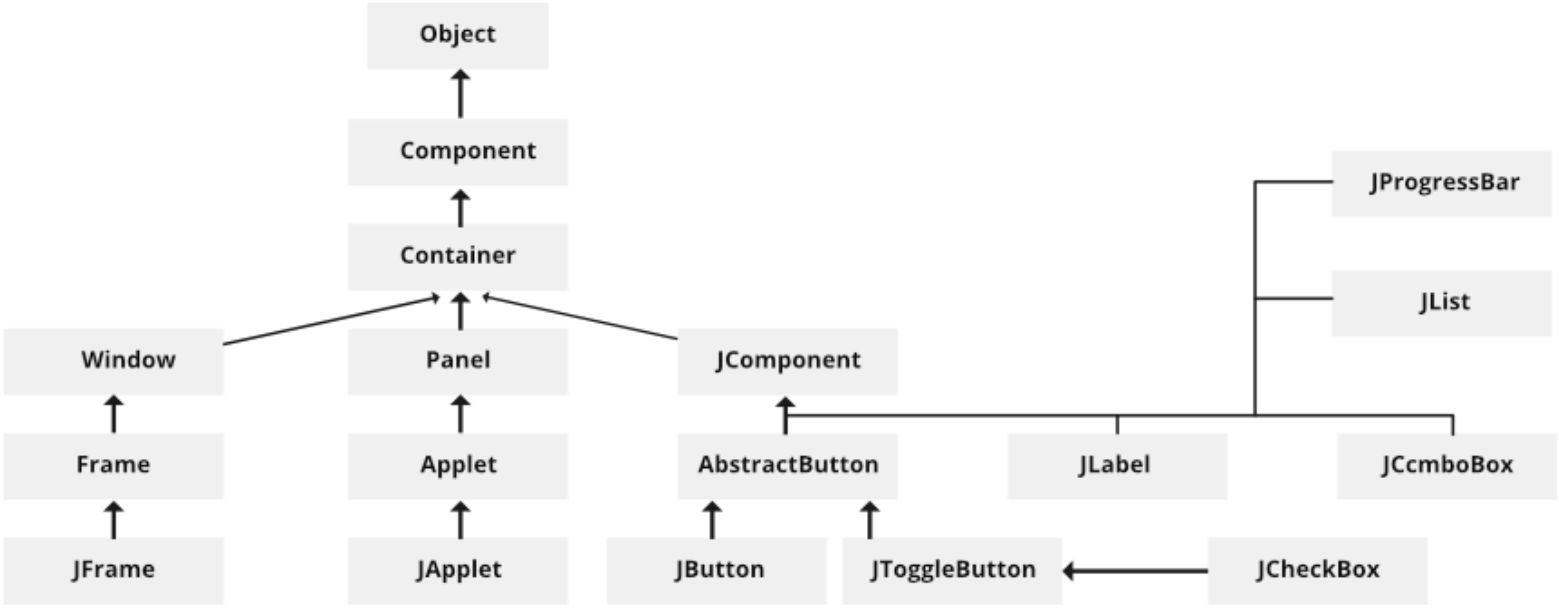
Java swing Вибір файла

```
JFileChooser jFileChooser = new JFileChooser();
int result = jFileChooser.showOpenDialog(null);
File file;
if(result==JFileChooser.APPROVE_OPTION){
    file = jFileChooser.getSelectedFile();
    System.out.println("File: " + file.getName());
    try {     ..    }
}
```

Java Swing

- Swing — це набір API (API — набір класів та інтерфейсів), що надається для розробки графічних інтерфейсів користувача
- Swing — це бібліотека розширення до AWT (Abstract Window Toolkit), включає нові та вдосконалені компоненти, які покращують зовнішній вигляд і функціональність графічних інтерфейсів користувача
- Він використовує архітектуру дизайну моделі/подання (та MVC).
- Swing повністю написаний на Java. Компоненти Java Swing не залежать від платформи, а компоненти Swing є легкими.
- Swing підтримує зовнішній вигляд Pluggable і Swing забезпечує більш потужні компоненти, наприклад, таблиці, списки, панелі прокручування, засіб вибору кольорів, панель із вкладками тощо.

Компоненти Java Swing



Компоненти Java Swing

Component — абстрактний базовий клас для елементів керування інтерфейсом користувача Java SWING, не пов’язаних із меню.

Container— це компонент, який може містити компоненти Java SWING

JComponent — це базовий клас для всіх компонентів інтерфейсу користувача swing. Щоб використовувати компонент swing, який успадковує JComponent, цей компонент має бути в ієрархії вмісту, коренем якої є контейнер Java Swing верхнього рівня.

JLabel — це об’єктний компонент для розміщення тексту в контейнері.

JButton - клас створює кнопку з міткою.

JColorChooser - надає панель елементів керування, призначену для того, щоб користувач міг керувати кольором і вибирати його.

Компоненти Java Swing

JCheckBox — це графічний (GUI) компонент, який може перебувати у стані on-(true) або off-(false).

JRadioButton — це графічний (GUI) компонент, який може перебувати у стані on-(true) або off-(false). в групі

JList – представляє контейнер зі списком текстових елементів, що прокручується.

JComboBox надає користувачеві випадне меню вибору.

JTextField — це текстовий компонент, який дозволяє редагувати один рядок тексту.

JPasswordField — це текстовий компонент, призначений для введення пароля.

JTextArea — це текстовий компонент, який дозволяє редагувати кілька рядків тексту.

JScrollbar – представляє компонент смуги прокручування, який дозволяє користувачам вибирати зі значень діапазону.

JOptionPane – надає набір стандартних діалогових вікон, які пропонують користувачам ввести значення або щось.

Події в Java Swing

У Java подія - це об'єкт, який представляє зміну стану програми. Це спосіб для однієї частини програми спілкуватися з іншою частиною, часто у відповідь на введення користувача або інші зовнішні стимули. Події зазвичай використовуються в графічних інтерфейсах користувача (GUI) для обробки взаємодії користувача, наприклад натискання кнопок, натискання клавіш і рухи миші.

Коли відбувається подія, вона зазвичай передається до прослуховувача подій, який є об’єктом, зареєстрованим для отримання та обробки певного типу події. Слухач обробляє подію та може виконувати будь-які необхідні дії, такі як оновлення інтерфейсу користувача або виконання обчислень.

Прослуховування подій

У програмуванні, керованому подіями Java, є два основні компоненти: слухач подій і джерело подій. Джерело події - це об'єкт, який генерує подію, тоді як слухач події - це об'єкт, який отримує та обробляє подію.

Щоб створити прослуховувач подій, ви повинні реалізувати відповідний інтерфейс для події, яку ви хочете обробити. Наприклад, для обробки подій дій (таких як натискання кнопок) ви повинні реалізувати інтерфейс `ActionListener`:

Прослуховування подій

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MyActionListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("Button clicked!");
    }
}
```

```java
import javax.swing.JButton;

public class Main {
    public static void main(String[] args) {
        JButton button = new JButton("Click me!");
        button.addActionListener(new MyActionListener());
    }
}
```

Створення власних подій

```
import java.util.EventObject;

public class MessageEvent extends EventObject {
   private final String message;

   public MessageEvent(Object source, String message) {
      super(source);
      this.message = message;
   }

   public String getMessage() {
      return message;
   }
}
```

Створення власних подій

```
public class MessageSource {
   private final List<MessageListener> listeners = new ArrayList<>();
   public void addMessageListener(MessageListener listener) {
      listeners.add(listener);
   }
   public void removeMessageListener(MessageListener listener) {
      listeners.remove(listener);
   }
   public void sendMessage(String message) {
      MessageEvent event = new MessageEvent(this, message);
      for (MessageListener listener : listeners) {
         listener.messageReceived(event);
      }
   }
}
```

Створення власних подій

```
interface MessageListener extends EventListener {
   void messageReceived(MessageEvent event);
}

public class Main {
   public static void main(String[] args) {
      MessageSource source = new MessageSource();
      source.addMessageListener(new MessageListener() {
         @Override
         public void messageReceived(MessageEvent event) {
            System.out.println("Отримано повідомлення: "
                  + event.getMessage());
         }
      });
      source.sendMessage("Привіт із Java!");
   }
}
```