

# Practical Java Programming Using Log4J

© FPT Software

1

<Lưu ý: không xóa nội dung cũ mà viết thêm lên như nhật ký>

Tài liệu đào tạo: <môn học, buổi học>

Phiên bản tài liệu:

Người cập nhật:

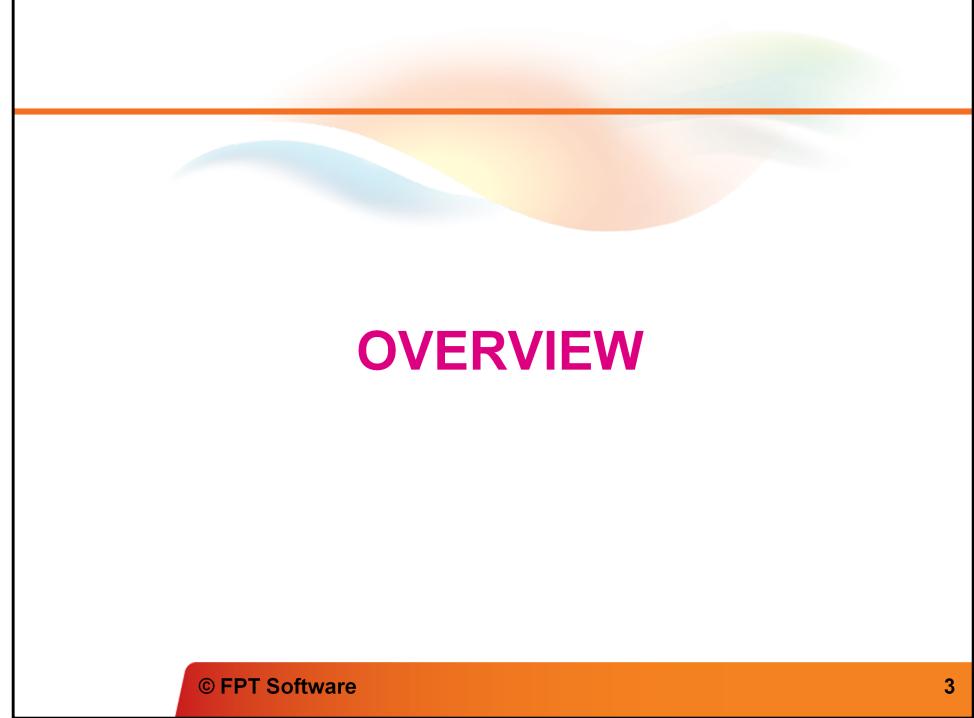
Ngày cập nhật:

Tóm tắt nội dung cập nhật chính:

Ngày ban hành sử dụng:

## Agenda

- Overview
- Add Log4J library
- Create configuration file
- Add logging initialization statements
- Add logging code
- Appender overview
- Log level overview
- Layout overview
- Filter overview



# OVERVIEW

© FPT Software

3

## Log4J Overview

- Log message in any place of code
- Control many kinds of output by appenders
- Customize the logging behavior for a specific code part
- Filter the log message by log level, keyword
- Format the output message by log layout

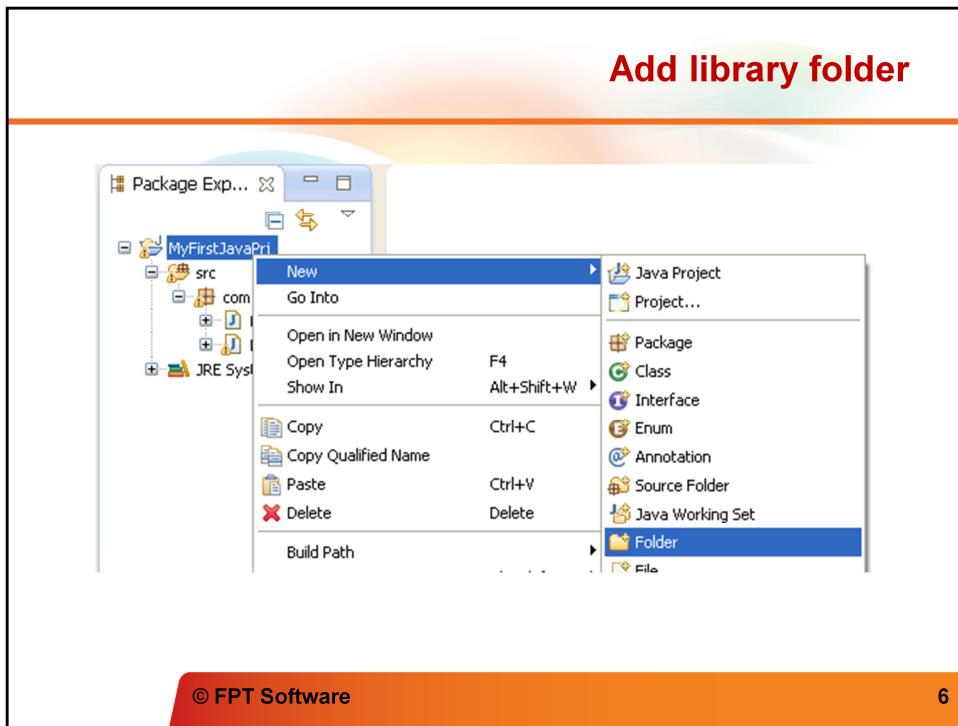


# LOG4J LIBRARY

© FPT Software

5

## Add library folder



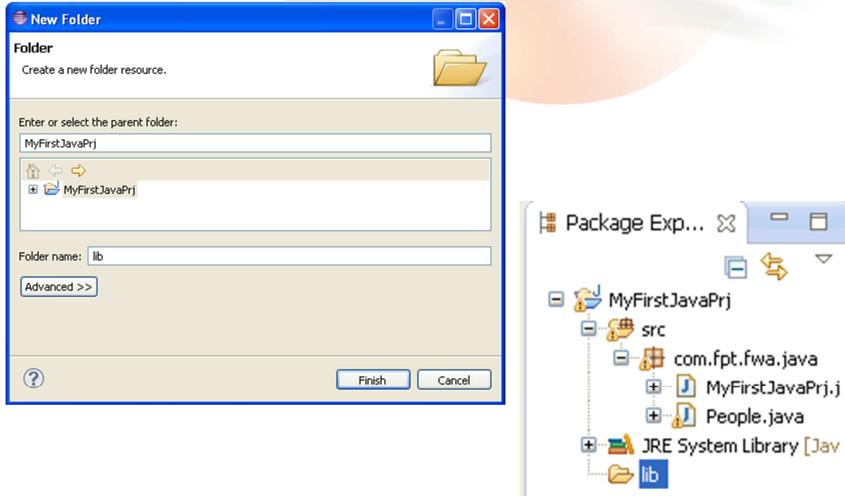
© FPT Software

6

Determine where the library stays

Then add library to project

## Add library folder dialog



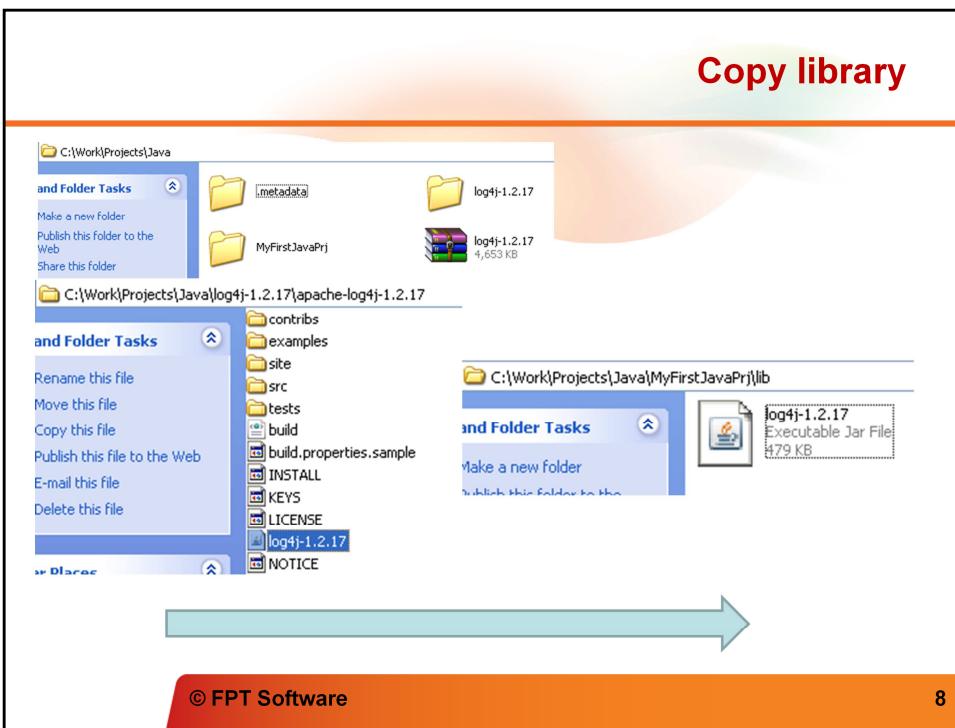
© FPT Software

7

Determine where the library stays

Then add library to project

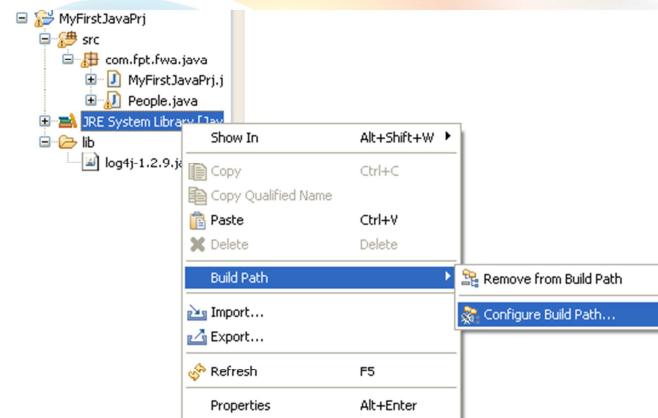
## Copy library



Determine where the library stays

Then copy library to the lib folder

## Add library



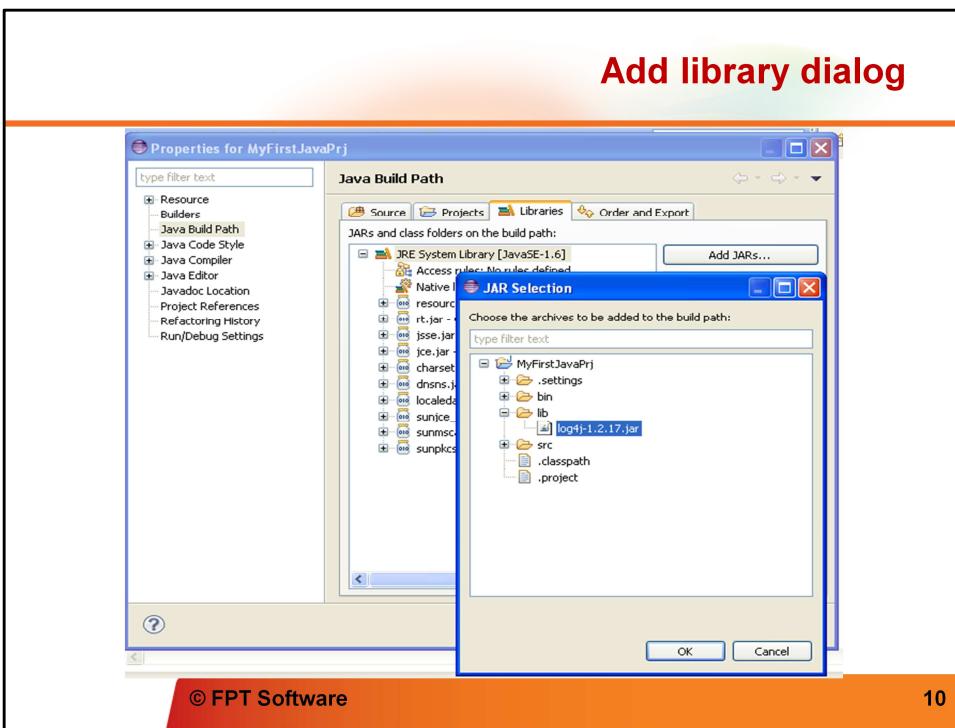
© FPT Software

9

Determine where the library stays

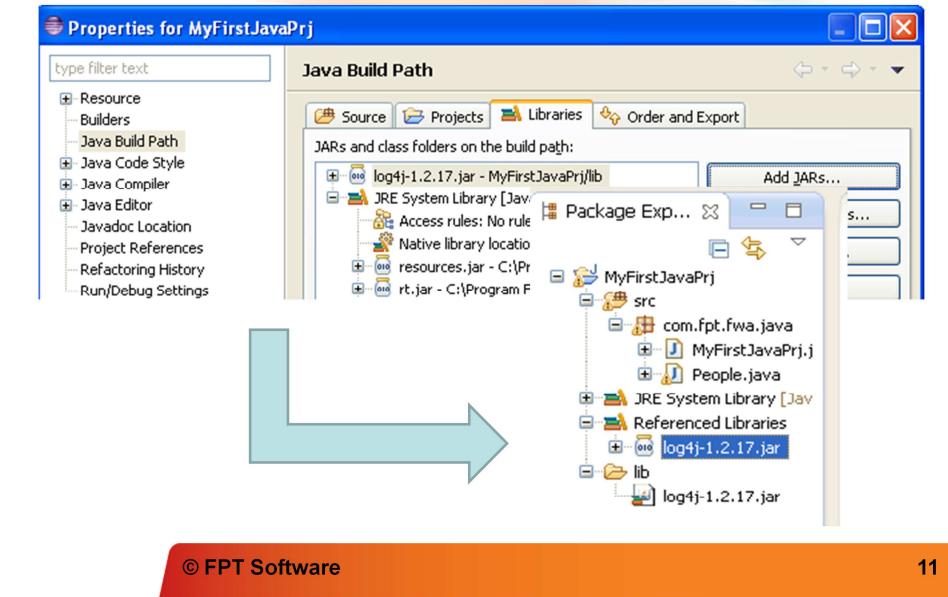
Then add library to project

## Add library dialog



10

## Add library result



© FPT Software

11



## **LOG4J CONFIGURATION FILE**

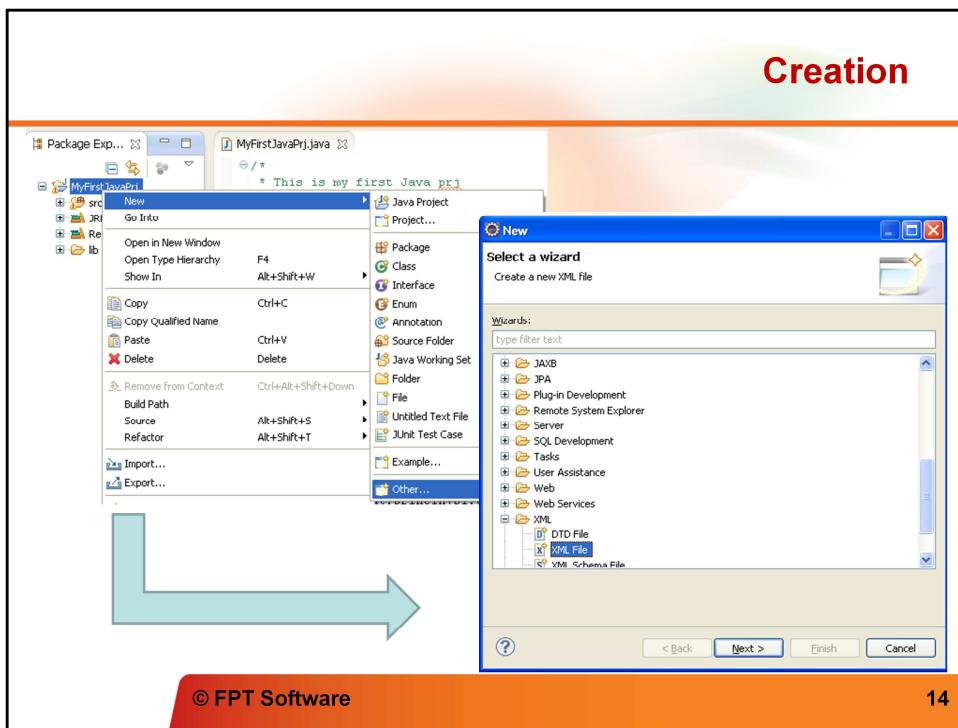
© FPT Software

12

## Content

- Add a new configuration file
- Register to Assembly
- Main structure
- Appender Section
- Logger Section

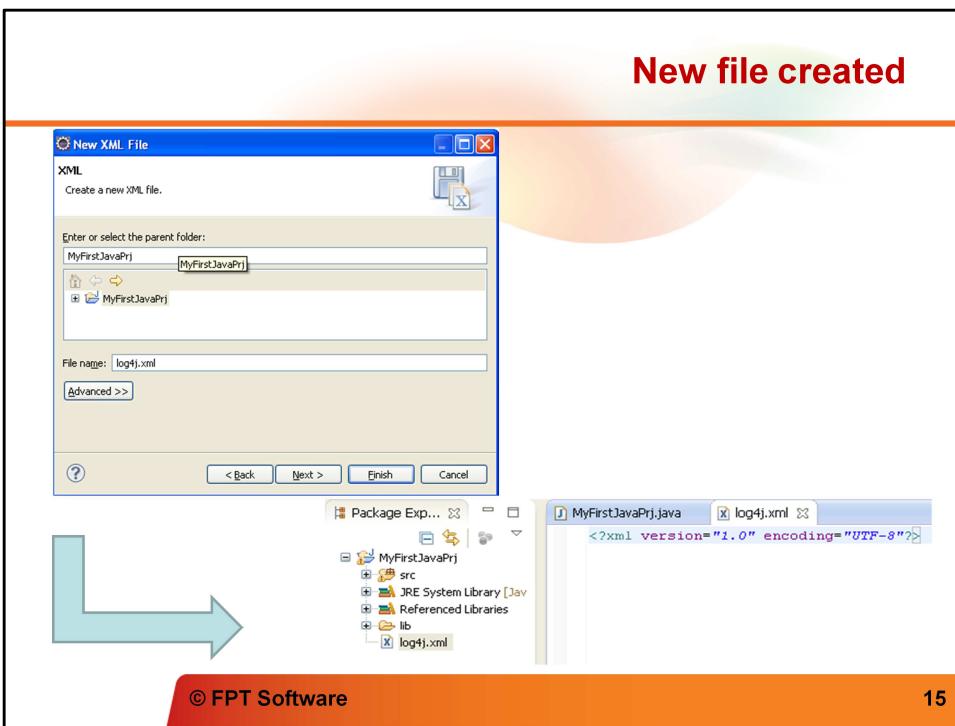
## Creation



© FPT Software

14

New file created



© FPT Software

15

## Main structure

The screenshot shows a Java IDE interface with two tabs: "MyFirstJavaPrj.java" and "log4j.xml". The "log4j.xml" tab is active, displaying the XML code for a Log4j configuration. The code includes declarations for log4j version, encoding, and namespaces, followed by sections for Appender and Logger configurations, and finally the closing tag for the log4j:configuration element.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd" >


    <!-- Appender -->

    <!-- Logger -->

</log4j:configuration>
```

© FPT Software

16

## Appenders

The screenshot shows an IDE interface with three tabs at the top: 'MyFirstJavaPrj.java', 'log4j.xml', and 'People.java'. The 'log4j.xml' tab is active, displaying the XML configuration for Log4J. The code lists two appenders: 'FileAppender' and 'ConsoleAppender'. The 'FileAppender' section includes parameters for the file name ('mylogfile.txt'), append mode ('true'), maximum backup index ('5'), and maximum file size ('1KB'). It also defines a 'PatternLayout' with a specific conversion pattern. The 'ConsoleAppender' section includes a 'PatternLayout' with its own conversion pattern, followed by a 'StringMatchFilter' with 'begin' and 'acceptOnMatch' parameters, and a 'DenyAllFilter'.

```
<!-- FileAppender -->
<appender name="FileAppender" class="org.apache.log4j.RollingFileAppender">
    <param name="file" value="mylogfile.txt" />
    <param name="append" value="true" />
    <param name="MaxBackupIndex" value="5" />
    <param name="MaxFileSize" value="1KB" />
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
</appender>

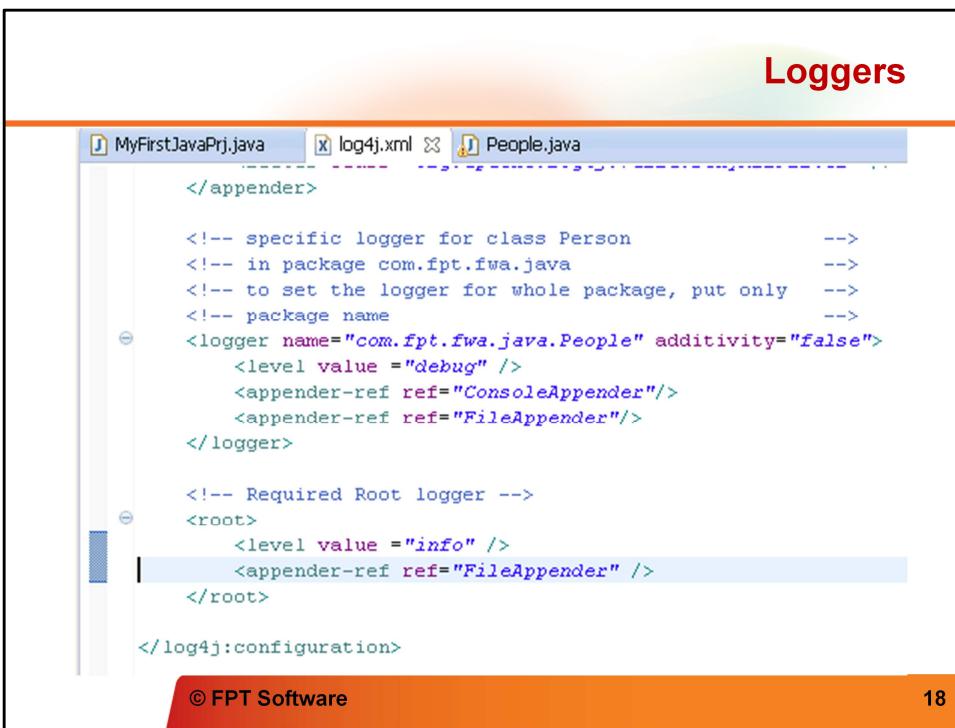
<!-- ConsoleAppender -->
<appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
    <filter class="org.apache.log4j.varia.StringMatchFilter">
        <param name="stringToMatch" value="begin" />
        <param name="acceptOnMatch" value="true" />
    </filter>
    <filter class="org.apache.log4j.varia.DenyAllFilter" />
</appender>
```

© FPT Software

17

Present detail about any element of appender, attention to filter

## Loggers



The screenshot shows a Java Integrated Development Environment (IDE) interface. At the top, there are three tabs: "MyFirstJavaPrj.java", "log4j.xml", and "People.java". The "log4j.xml" tab is currently active, displaying XML code for a Log4J configuration file. The code includes sections for appenders, a specific logger for the "com.fpt.fwa.java.People" class, and a root logger. The root logger is configured to level "info" and uses a "FileAppender". A mouse cursor is visible over the XML code, specifically pointing at the "FileAppender" reference in the root logger's configuration.

```
</appender>

<!-- specific logger for class Person -->
<!-- in package com.fpt.fwa.java -->
<!-- to set the logger for whole package, put only -->
<!-- package name -->
<logger name="com.fpt.fwa.java.People" additivity="false">
    <level value ="debug" />
    <appender-ref ref="ConsoleAppender"/>
    <appender-ref ref="FileAppender"/>
</logger>

<!-- Required Root logger -->
<root>
    <level value ="info" />
    <appender-ref ref="FileAppender" />
</root>

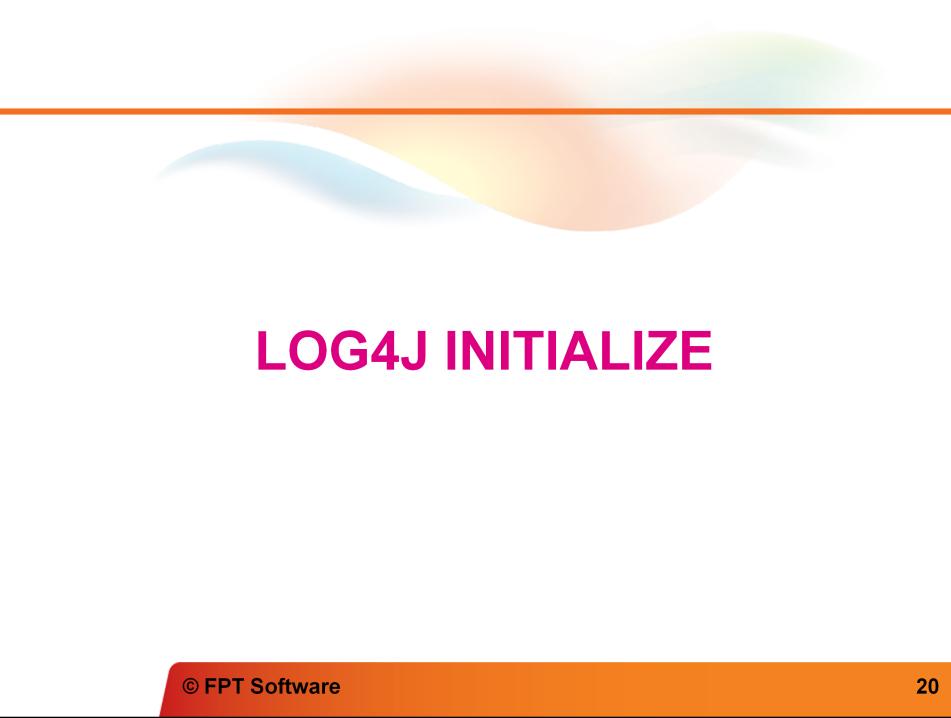
</log4j:configuration>
```

© FPT Software

18

## Summary

- Add a new configuration file: Add xml file to the root folder
- Main structure: Appenders and loggers
- Appender Section: File rolling and console
- Logger Section: Root required and specific logger is optional



## LOG4J INITIALIZE

## Initialization code

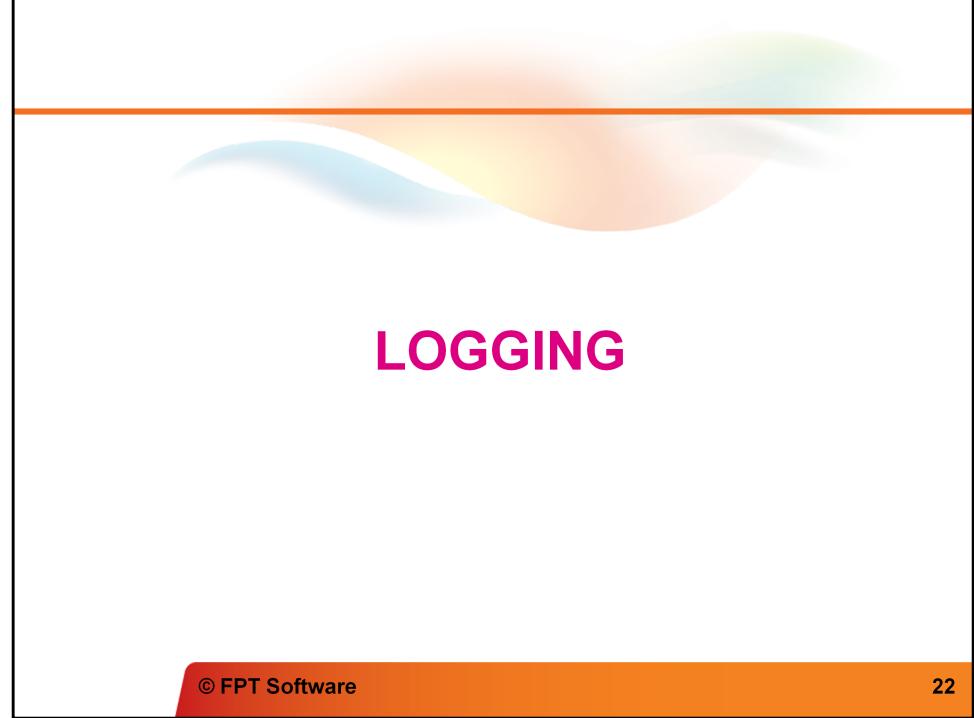
```
MyFirstJavaPrj.java ✘ log4j.xml ✘ People.java
package com.fpt.fwa.java;

// Import log4j classes.
import org.apache.log4j.Logger;
import org.apache.log4j.xml.DOMConfigurator;

/* */

// any program starts with Main method, with or without
public static void main(String[] args) {

    DOMConfigurator.configure("log4j.xml");
    logger.info("Entering application.");
}
```



# LOGGING

© FPT Software

22

## Add logging

```
MyFirstJavaPrj.java  log4j.xml  People.java
package com.fpt.fwa.java;

// Import log4j classes.
import org.apache.log4j.Logger;
import org.apache.log4j.xml.DOMConfigurator;

public class MyFirstJavaPrj {

    // Define a static logger variable so that it references the
    // Logger instance named "MyFirstJavaPrj".
    static Logger logger = Logger.getLogger(MyFirstJavaPrj.class);

    // any program starts with Main method, with or with
    public static void main(String[] args) {

        DOMConfigurator.configure("log4j.xml");
        logger.info("Entering application.");

        People p1 = new People("FPT 03", "03.03.2013");
        System.out.println(p1.toString());

        logger.info("Exiting application.");
    }
}
```

© FPT Software

23

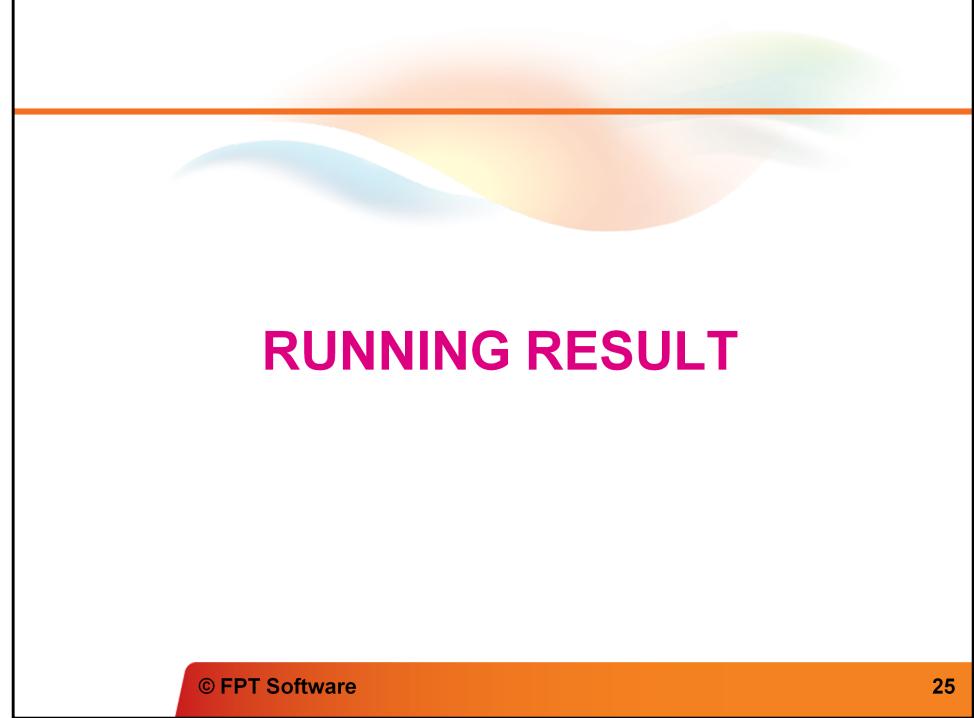
## Multi level logging

```
MyFirstJavaPrj.java    log4j.xml    People.java
}
public String toString(){
    logger.debug("Person toString begin.");
    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
    if (!isValid) return "Is not valide object";
    logger.info("Another message with the \"begin\" word.");
    logger.debug("Person toString end.");
    return name + df.format(dayOfBirth);
}
```

© FPT Software

24

Last log event is not catch cause of Filter

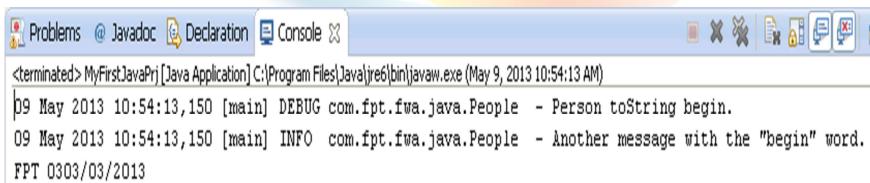


## RUNNING RESULT

© FPT Software

25

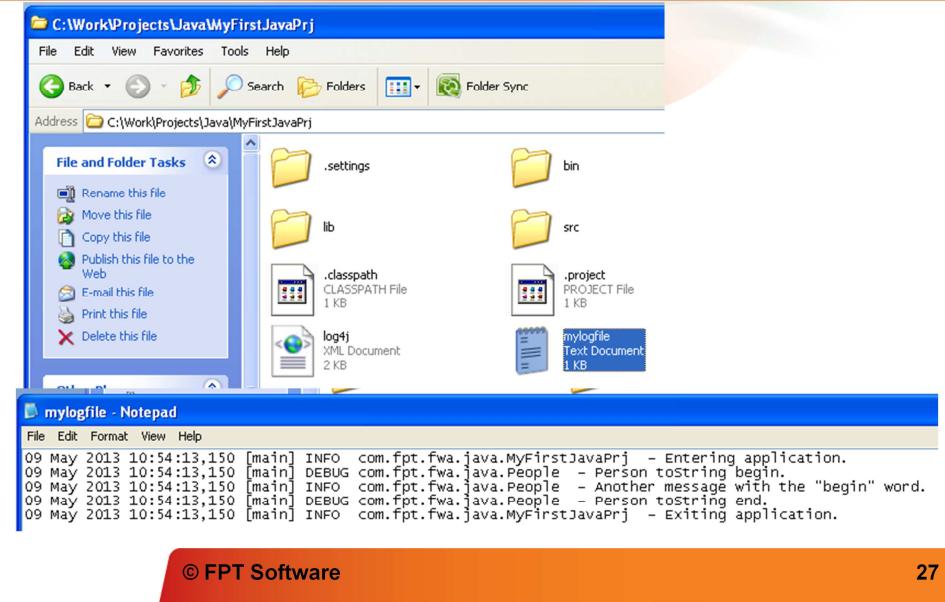
## Console Output



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected in the top bar. The console window displays the following output:

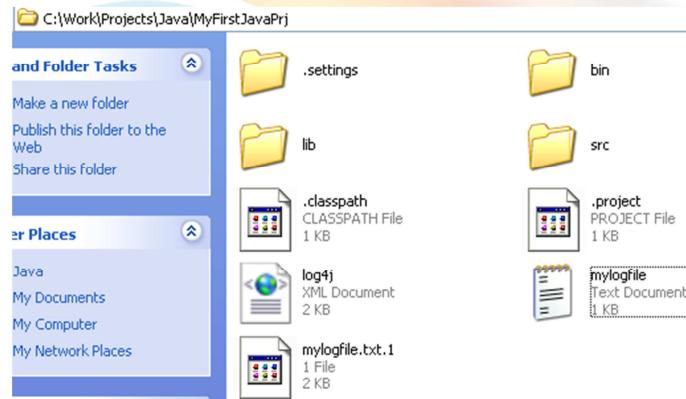
```
<terminated> MyFirstJavaProj [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (May 9, 2013 10:54:13 AM)
09 May 2013 10:54:13,150 [main] DEBUG com.fpt.fwa.java.People - Person toString begin.
09 May 2013 10:54:13,150 [main] INFO com.fpt.fwa.java.People - Another message with the "begin" word.
FPT 0303/03/2013
```

## File Output



Point out the folder and file name

## File Output: rolling



© FPT Software

28

The last line is not sent out cause of filter

## File Output: rolling file

```
mylogfile - Notepad
File Edit Format View Help
09 May 2013 10:57:05,542 [main] DEBUG com.fpt.fwa.java.People - Person toString begin.
09 May 2013 10:57:05,557 [main] INFO com.fpt.fwa.java.People - Another message with the "begin" word.
09 May 2013 10:57:05,557 [main] DEBUG com.fpt.fwa.java.People - Person toString end.
09 May 2013 10:57:05,557 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Exiting application.
```

```
mylogfile.txt - Notepad
File Edit Format View Help
09 May 2013 10:54:13,150 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Entering application.
09 May 2013 10:54:13,150 [main] DEBUG com.fpt.fwa.java.People - Person toString begin.
09 May 2013 10:54:13,150 [main] INFO com.fpt.fwa.java.People - Another message with the "begin" word.
09 May 2013 10:54:13,150 [main] DEBUG com.fpt.fwa.java.People - Person toString end.
09 May 2013 10:54:13,150 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Exiting application.
09 May 2013 10:56:52,620 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Entering application.
09 May 2013 10:56:52,635 [main] DEBUG com.fpt.fwa.java.People - Person toString begin.
09 May 2013 10:56:52,635 [main] INFO com.fpt.fwa.java.People - Another message with the "begin" word.
09 May 2013 10:56:52,635 [main] DEBUG com.fpt.fwa.java.People - Person toString end.
09 May 2013 10:56:52,635 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Exiting application.
09 May 2013 10:57:05,542 [main] INFO com.fpt.fwa.java.MyFirstJavaPrj - Entering application.
```

© FPT Software

29

The last line is not sent out cause of filter



## APPENDERS

© FPT Software

30

## Appender using

```
<!-- FileAppender -->
<appender name="FileAppender" class="org.apache.log4j.RollingFileAppender">
    <param name="file" value="mylogfile.txt" />
    <param name="append" value="true" />
    <param name="MaxBackupIndex" value="5" />
    <param name="MaxFileSize" value="1KB" />
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
</appender>

<!-- ConsoleAppender -->
<appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
    <filter class="org.apache.log4j.varia.StringMatchFilter">
        <param name="stringToMatch" value="begin" />
        <param name="acceptOnMatch" value="true" />
    </filter>
    <filter class="org.apache.log4j.varia.DenyAllFilter" />
</appender>
```

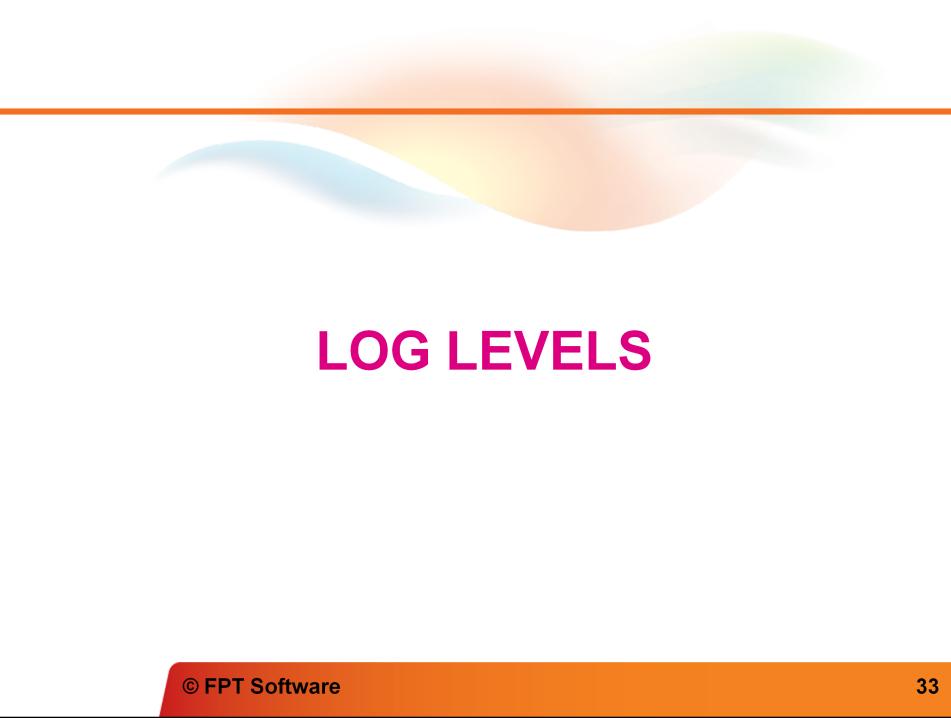
© FPT Software      31

Present detail about any element of appender, attention to filter

## Appenders

- File rolling
- Console
- File
- Daily File rolling
- JDBC – to Database
- SMTP – by email
- Other...

All == Debug

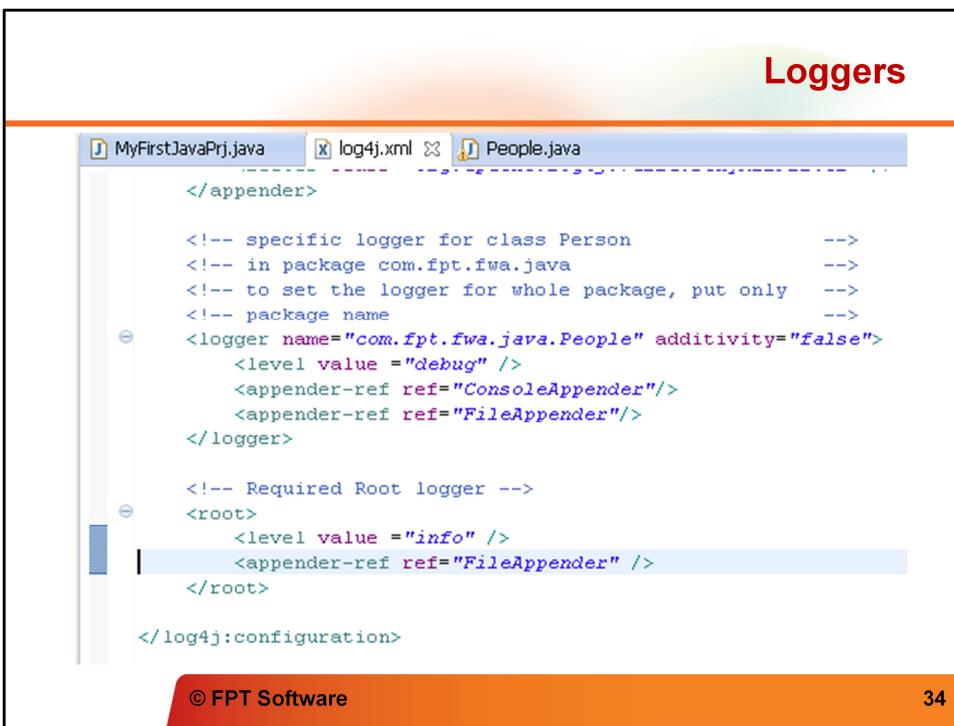


## LOG LEVELS

© FPT Software

33

## Loggers



The screenshot shows a Java IDE interface with three tabs at the top: "MyFirstJavaPrj.java", "log4j.xml", and "People.java". The "log4j.xml" tab is active, displaying XML code for a logger configuration. The code includes sections for a specific logger for the package com.fpt.fwa.java and a root logger. The root logger is configured to level "info" and append to a "FileAppender". The XML code is as follows:

```
</appender>

<!-- specific logger for class Person -->
<!-- in package com.fpt.fwa.java -->
<!-- to set the logger for whole package, put only -->
<!-- package name -->
<logger name="com.fpt.fwa.java.People" additivity="false">
    <level value ="debug" />
    <appender-ref ref="ConsoleAppender"/>
    <appender-ref ref="FileAppender"/>
</logger>

<!-- Required Root logger -->
<root>
    <level value ="info" />
    <appender-ref ref="FileAppender" />
</root>

</log4j:configuration>
```

© FPT Software

34

## Log level using



The screenshot shows a Java code editor with three tabs at the top: "MyFirstJavaPrj.java", "log4j.xml", and "People.java". The "People.java" tab is active, displaying the following code:

```
    }
    public String toString(){
        logger.debug("Person toString begin.");
        SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy");
        if (!isValid) return "Is not valide object";
        logger.info("Another message with the \"begin\" word.");
        logger.debug("Person toString end.");
        return name + df.format(dayOfBirth);
    }
}
```

The code uses the `logger` object to log messages at different levels: `debug`, `info`, and `debug` again.

© FPT Software

35

All == Debug

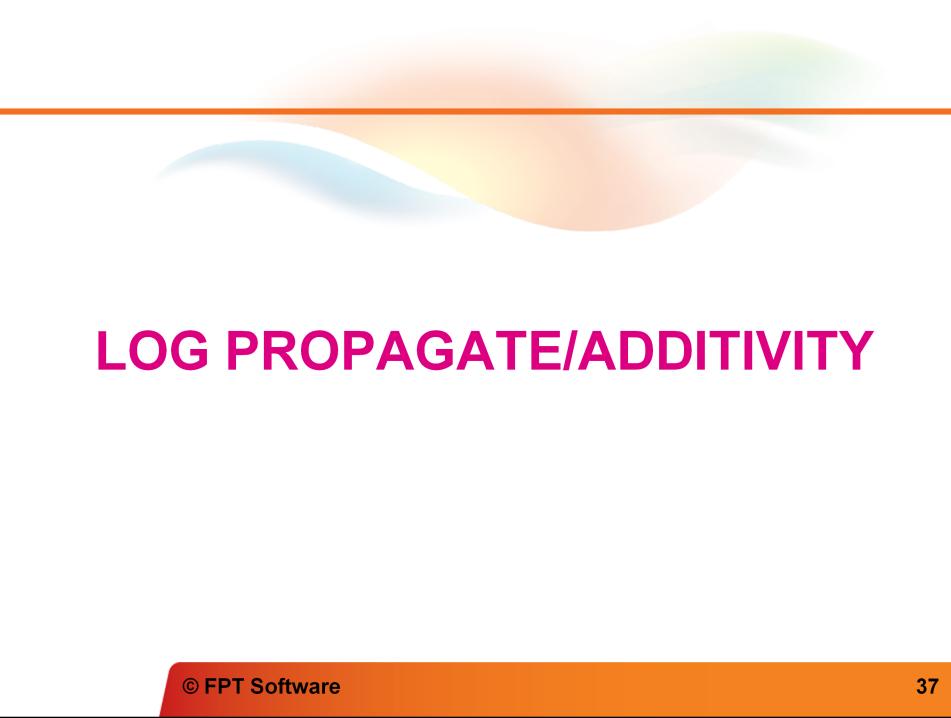
## Log levels

- TRACE
- DEBUG
- INFO
- WARN
- ERROR
- FATAL

Order from Trace to Fatal

Logger with level Trace will display msg for all log level events

Logger with level Debug will display msg for all log level event from Debug to Fatal  
and so on

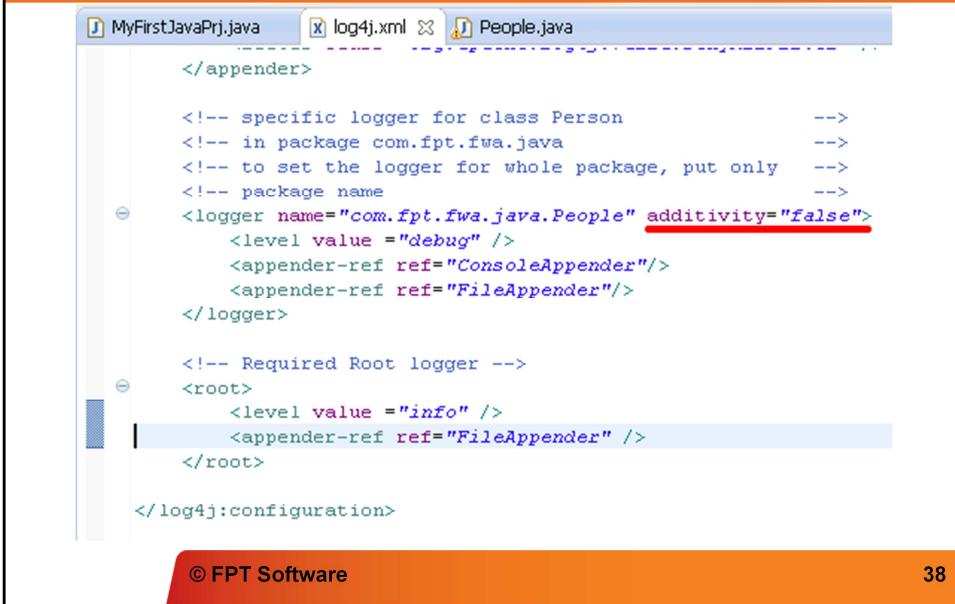


## **LOG PROPAGATE/ADDITIVITY**

© FPT Software

37

## Log propagate/Additivity



The screenshot shows an IDE interface with three tabs at the top: MyFirstJavaPrj.java, log4j.xml, and People.java. The log4j.xml tab is active and displays the following XML configuration:

```
</appender>

<!-- specific logger for class Person -->
<!-- in package com.fpt.fwa.java -->
<!-- to set the logger for whole package, put only -->
<!-- package name -->
<logger name="com.fpt.fwa.java.People" additivity="false">
    <level value ="debug" />
    <appender-ref ref="ConsoleAppender"/>
    <appender-ref ref="FileAppender"/>
</logger>

<!-- Required Root logger -->
<root>
    <level value ="info" />
    <appender-ref ref="FileAppender" />
</root>

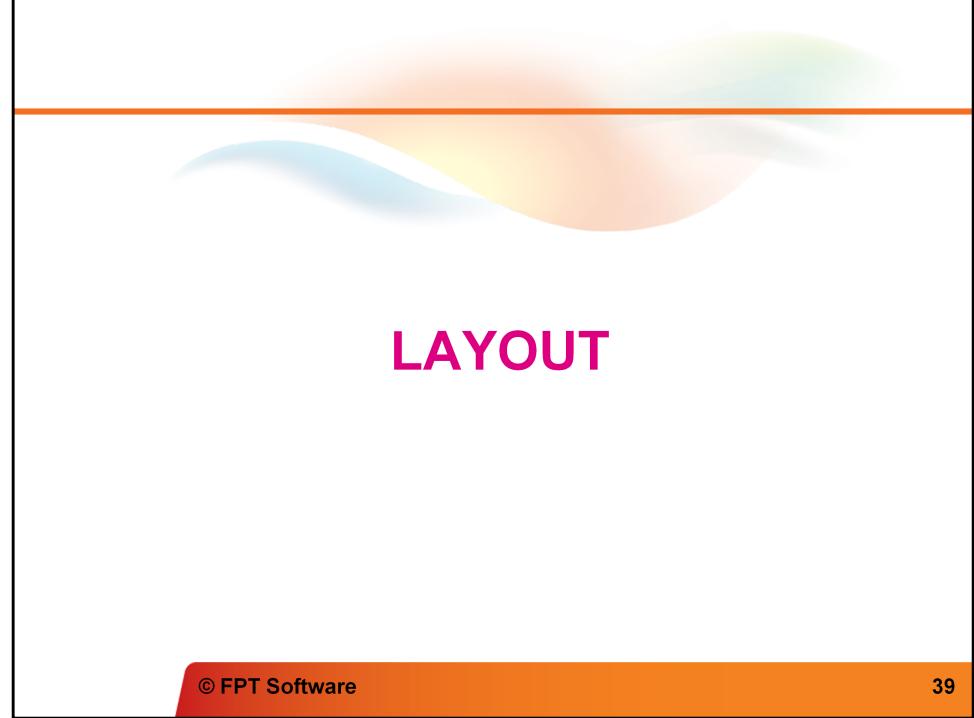
</log4j:configuration>
```

A red underline highlights the attribute `additivity="false"` under the `logger` element. A blue selection bar highlights the `<level value="info" />` and `<appender-ref ref="FileAppender" />` lines under the `<root>` element.

© FPT Software

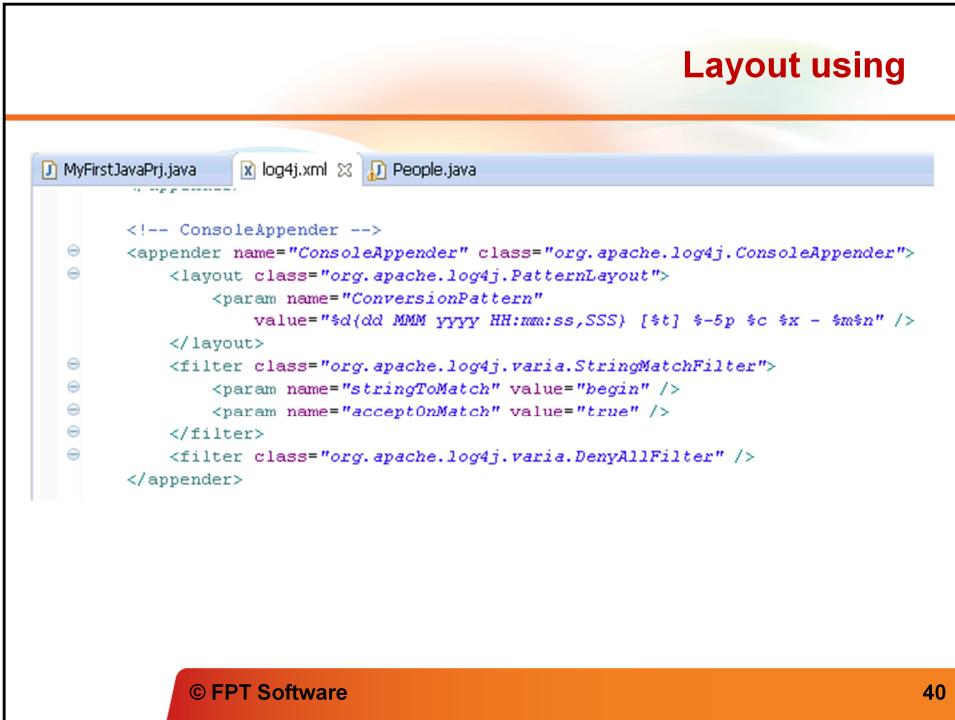
38

All == Debug



# LAYOUT

## Layout using



The screenshot shows a Java development environment with three tabs open: MyFirstJavaPrj.java, log4j.xml, and People.java. The log4j.xml tab is active and displays the following XML configuration:

```
<!-- ConsoleAppender -->
<appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
    <filter class="org.apache.log4j.varia.StringMatchFilter">
        <param name="stringToMatch" value="begin" />
        <param name="acceptOnMatch" value="true" />
    </filter>
    <filter class="org.apache.log4j.varia.DenyAllFilter" />
</appender>
```

© FPT Software

40

All == Debug

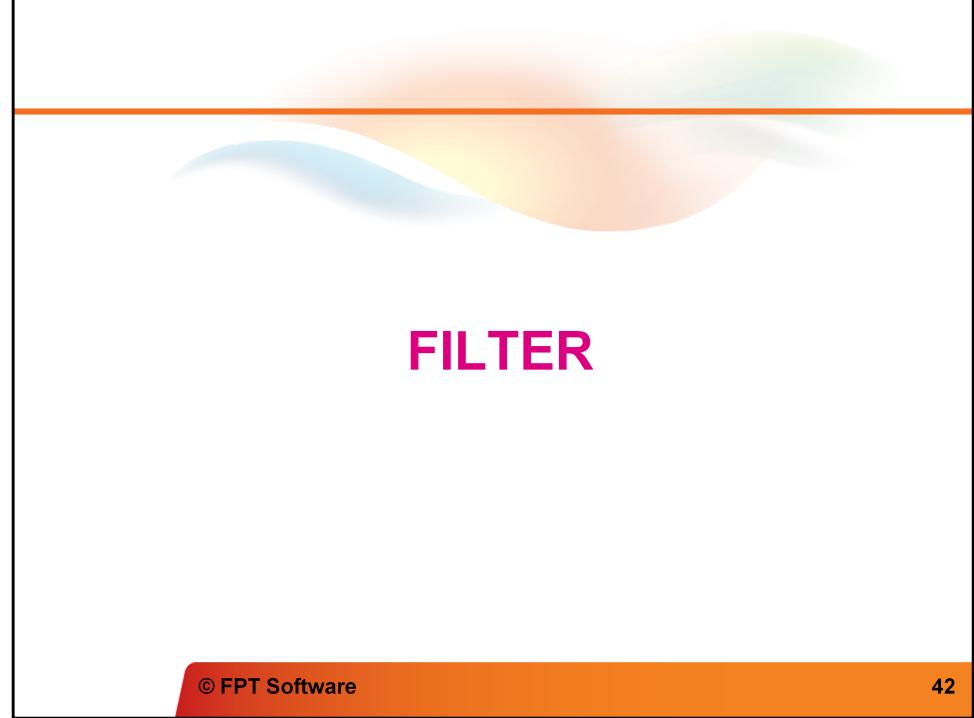
## Layouts

- PatternLayout
- HTMLayout
- SimpleLayout

PatternLayout Formats the logging event according to a flexible set of formatting flags.

HTMLayout                           output in HTML table

SimpleLayout Formats the logging event very simply: [level] - [message]

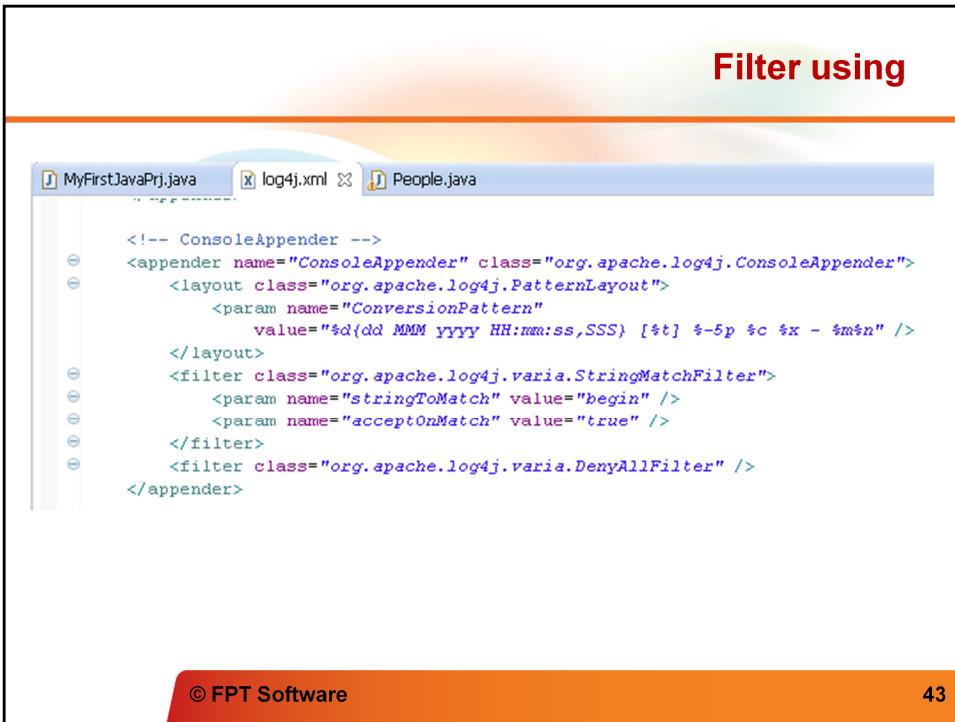


# FILTER

© FPT Software

42

## Filter using



The screenshot shows a Java IDE interface with three tabs at the top: "MyFirstJavaPrj.java", "log4j.xml", and "People.java". The "log4j.xml" tab is active, displaying XML code for a Log4J configuration file. The code defines a "ConsoleAppender" with a "PatternLayout" that includes a "StringMatchFilter". This filter is configured to accept logs where the string "begin" is found at the start of the log message. A "DenyAllFilter" is also present to handle any remaining logs.

```
<!-- ConsoleAppender -->
<appender name="ConsoleAppender" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern"
              value="%d{dd MMM yyyy HH:mm:ss,SSS} [%t] %-5p %c %x - %m%n" />
    </layout>
    <filter class="org.apache.log4j.varia.StringMatchFilter">
        <param name="stringToMatch" value="begin" />
        <param name="acceptOnMatch" value="true" />
    </filter>
    <filter class="org.apache.log4j.varia.DenyAllFilter" />
</appender>
```

© FPT Software

43

All == Debug

## Filters

- DenyAllFilter
- LevelMatchFilter
- LevelRangeFilter
- StringMatchFilter

### Attention to filter order

DenyAllFilter	Drops all “other” logging events.
LevelMatchFilter	An exact match to the event's level.
LevelRangeFilter	Matches against a range of levels.
StringMatchFilter	Matches a substring from the event's message.

## Lesson summary

- Add Log4J library: add reference in build path
- Create configuration file: xml file
- Add logging initialization statements: in main java file
- Add logging code: in each module
- Appender overview: File rolling, console, SQL
- Log level overview: 6 options
- Layout overview: output format
- Filter overview: output restriction



© FPT Software

46