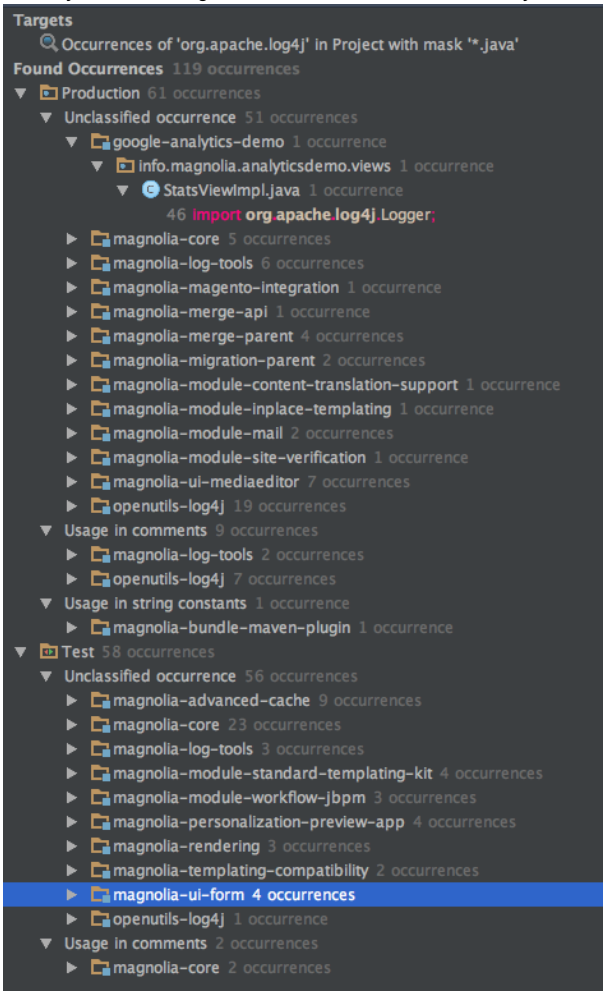
Log4j2 upgrading

Apache Log4j 2 is an upgrade to Log4j that provides significant improvements over its predecessor, Log4j 1.x, and provides many of the improvements available in Logback while fixing some inherent problems in Logback's architecture.

Information

- Ticket MAGNOLIA-6794
- Spring framework implementation
- Changes http://logging.apache.org/log4j/2.x/changes-report.html#a2.7
    - Latest version: 2.7 - 2016-10-02
- Frequently Asked Questions - https://logging.apache.org/log4j/2.0/faq.html
- Migrating from Log4j 1.x - http://logging.apache.org/log4j/2.x/manual/migration.html
- Dependencies
    - Using Log4j in your Apache Maven build - http://logging.apache.org/log4j/2.x/maven-artifacts.html
    - Log4j Runtime Dependencies - https://logging.apache.org/log4j/2.0/runtime-dependencies.html
    - Log4j 1.2 bridge - http://logging.apache.org/log4j/2.x/log4j-1.2-api/
- Configuration
    - Programmatic Configuration http://logging.apache.org/log4j/2.x/manual/customconfig.html
- Plugins - http://logging.apache.org/log4j/2.x/manual/plugins.html

Problems

1. New package
    - *org.apache.log4j* changed to *org.apache.logging.log4j*
    - We may need to change in a bunch of modules when we try to convert to the new package
    - 

2. DOM and Property Configurator was removed
    - They're not available in new version - https://issues.apache.org/jira/browse/LOG4J2-340
    - Replaced with an empty implementation if we used Log4j 1.2 bridge
3. Log4J 2 over the old fashioned properties file is not supported so far
    - Log4j 2 doesn't support log4j.properties file anymore?
    - Support configuration from version 1.x log4j.properties - https://issues.apache.org/jira/browse/LOG4J2-63
4. API for setLevel / getEffectiveLevel - http://apache-logging.6191.n7.nabble.com/API-and-setLevel-getEffectiveLevel-td36238.html
5. Unit-testing
    a. Add Appender

```
Writer loggerOut = new StringWriter();
final Layout layout = new EnhancedPatternLayout(EnhancedPatternLayout.TTCC_CONVERSION_PATTERN);
Logger.getRootLogger().setLevel(Level.INFO); // Other tests might have set this to silent
Logger.getRootLogger().addAppender(new WriterAppender(layout, loggerOut));
```

        - *EnhancedPatternLayout* is removed. Maybe use *PatternLayout* directly
        - Need an another way to setLevel for logger - http://stackoverflow.com/questions/23434252/programmatically-change-log-level-in-log4j2
        - *Logger.getRootLogger().addAppender* can not be used anymore. Should use a Log4j2 plugin to add an appender (http://stackoverflow.com/questions/24205093/how-to-create-a-custom-appender-in-log4j2)
        - *WriterAppender* is removed also
    b. Custom Appender

```
public static class TestAppender extends AppenderSkeleton {
    public List<LoggingEvent> events = new ArrayList<>();

    @Override
    public void close() {
    }

    @Override
    public boolean requiresLayout() {
        return false;
    }

    @Override
    protected void append(LoggingEvent event) {
        events.add(event);
    }
}
```

    Need to rewrite as a plugin.
    c. Custom Level

```
public class LoggingLevel extends Level {

    public static final LoggingLevel AUDIT_TRAIL = new LoggingLevel(99, "AUDIT_TRAIL", 0);

    protected LoggingLevel(int level, String levelStr, int syslogEquivalent) {
        super(level, levelStr, syslogEquivalent);
    }

    public static Level toLevel(String sArg) {
        return AUDIT_TRAIL;
    }

    public static Level toLevel(int val) {
        return AUDIT_TRAIL;
    }

}
```

6. Log Tools can not start due to it relies on Log4j

```
private AbstractBeanContainer populateContainer() {
    BeanItemContainer<LogLevelBean> container = new BeanItemContainer<>(LogLevelBean.class);
    Enumeration<Logger> loggers = logManager.getCurrentLoggers();

    while (loggers.hasMoreElements()) {
        Logger logger = loggers.nextElement();
        LogLevelBean logLevelBean = new LogLevelBean(
                logger.getName(),
                logger.getEffectiveLevel(),
                // if the level is null, then that means the effectiveLevel is inherited from the parent,
                // rather than copied from the level itself. therefore, we want to show the inherited
tick
                // (inherited = true)
                logger.getLevel() == null
        );

        container.addBean(logLevelBean);
    }

    container.setItemSorter(new DefaultItemSorter() {
        @Override
        public int compare(Object o1, Object o2) {
            return ((LogLevelBean) o1).getName().compareTo(((LogLevelBean) o2).getName());
        }
    });

    return container;
}
```

7.

Code Snippets

```
public static void updateLoggers(final Appender appender, final Configuration config) {
    final Level level = null;
    final Filter filter = null;
    for (final LoggerConfig loggerConfig : config.getLoggers().values()) {
        loggerConfig.addAppender(appender, level, filter);
    }
    config.getRootLogger().addAppender(appender, level, filter);
}
```

```
/**
 * http://stackoverflow.com/questions/23434252/programmatically-change-log-level-in-log4j2
 */
public static void setLevel(Logger logger, Level level) {
    final LoggerContext ctx = (LoggerContext) LogManager.getContext(false);
    final Configuration config = ctx.getConfiguration();

    LoggerConfig loggerConfig = config.getLoggerConfig(logger.getName());
    LoggerConfig specificConfig = loggerConfig;

    // We need a specific configuration for this logger,
    // otherwise we would change the level of all other loggers
    // having the original configuration as parent as well

    if (!loggerConfig.getName().equals(logger.getName())) {
        specificConfig = new LoggerConfig(logger.getName(), level, true);
        specificConfig.setParent(loggerConfig);
        config.addLogger(logger.getName(), specificConfig);
    }
    specificConfig.setLevel(level);
    ctx.updateLoggers();
}
```