**Замена Логгирования в Spring**

По умолчанию в Spring уже используется логгирование (JCL) с использованием apache commons-logging, поэтому зависимость от commons-logging попадает в проект при импорте spring-core.

**Использование logback**

Для начала необходимо убрать зависимость от commons-logging. Для этого в секции объявления зависимости Spring-core добавим исключения дочерней зависимости.

pom.xml

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.framework.version}</version>

<exclusions>

<exclusion>

<artifactId>commons-logging</artifactId>

<groupId>commons-logging</groupId>

</exclusion>

</exclusions>

</dependency>

Spring логгирует через JCL, поэтому для начала необходимо добавить зависимости для преобразования JCL и JUL в вызовы SLF4J, для этого добавим следующие зависимости.

pom.xml

<properties>

<slf4j.version>1.7.3</slf4j.version>

</properties>

pom.xml

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jcl-over-slf4j</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>jul-to-slf4j</artifactId>

<version>${slf4j.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>${slf4j.version}</version>

</dependency>

pom.xml

<properties>

<logback.version>1.0.10</logback.version>

</properties>

pom.xml

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-core</artifactId>

<version>${logback.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>${logback.version}</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-access</artifactId>

<version>${logback.version}</version>

</dependency>

Скомпилируем и запустим приложение с настройками по умолчанию.

20:25:01.955 [main] DEBUG o.s.core.env.StandardEnvironment - Initializing new StandardEnvironment

20:25:01.979 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemProperties] PropertySource with lowest search precedence

20:25:01.983 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemEnvironment] PropertySource with lowest search precedence

20:25:01.984 [main] DEBUG o.s.core.env.StandardEnvironment - Initialized StandardEnvironment with PropertySources [systemProperties,systemEnvironment]

20:25:01.993 [main] INFO o.s.c.s.ClassPathXmlApplicationContext - Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@c126b3: startup date [Sun Nov 11 20:25:01 MSK 2012]; root of context hierarchy

20:25:02.133 [main] DEBUG o.s.core.env.StandardEnvironment - Initializing new StandardEnvironment

20:25:02.134 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemProperties] PropertySource with lowest search precedence

20:25:02.134 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemEnvironment] PropertySource with lowest search precedence

20:25:02.135 [main] DEBUG o.s.core.env.StandardEnvironment - Initialized StandardEnvironment with PropertySources [systemProperties,systemEnvironment]

20:25:02.137 [main] DEBUG o.s.core.env.StandardEnvironment - Initializing new StandardEnvironment

20:25:02.138 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemProperties] PropertySource with lowest search precedence

20:25:02.139 [main] DEBUG o.s.core.env.StandardEnvironment - Adding [systemEnvironment] PropertySource with lowest search precedence

20:25:02.140 [main] DEBUG o.s.core.env.StandardEnvironment - Initialized StandardEnvironment with PropertySources [systemProperties,systemEnvironment]

20:25:02.176 [main] INFO o.s.b.f.xml.XmlBeanDefinitionReader - Loading XML bean definitions from class path resource [app-context.xml]

20:25:02.402 [main] DEBUG o.s.b.f.xml.DefaultDocumentLoader - Using JAXP provider [com.sun.org.apache.xerces.internal.jaxp.DocumentBuilderFactoryImpl]

20:25:02.744 [main] DEBUG o.s.b.f.xml.PluggableSchemaResolver - Loading schema mappings from [META-INF/spring.schemas]

Logback сокращает название пространства имён и по умолчанию выводит все логи.

Сократим логгирование до уровня INFO для Spring framework и DEBUG для классов приложения. Для этого создадим файл src/main/resources/logback.xml

logback.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%.-19date;%.-1level;%logger;%msg%n</pattern>

</encoder>

</appender>

<logger name="org.springframework" level="info" />

<logger name="a1s" level="debug" />

<root level="warn">

<appender-ref ref="console" />

</root>

</configuration>

Формат лога задан как %.-19date;%.-1level;%logger;%msg%n, что будет интерпретировано как:

<Дата>;<Первая буква из уровня логгирования>;<Наименование логгера>;<Сообщение>

пример:

2012-11-11 20:36:06;D;a1s.learn.App;User found

Вывод приложения серьёзно сократился и будет таким:

2012-11-11 20:36:03;I;org.springframework.context.support.ClassPathXmlApplicationContext;Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@b1054b: startup date [Sun Nov 11 20:36:02 MSK 2012]; root of context hierarchy

2012-11-11 20:36:03;I;org.springframework.beans.factory.xml.XmlBeanDefinitionReader;Loading XML bean definitions from class path resource [app-context.xml]

2012-11-11 20:36:03;I;org.springframework.jdbc.datasource.DriverManagerDataSource;Loaded JDBC driver: com.mysql.jdbc.Driver

2012-11-11 20:36:04;I;org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;Building JPA container EntityManagerFactory for persistence unit 'default'

2012-11-11 20:36:05;I;org.springframework.beans.factory.support.DefaultListableBeanFactory;Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@1595050: defining beans [dataSource,emf,userService,userDAO,org.springframework.context.annotation.internalConfigurationAnnotationProcessor,org.springframework.context.annotation.internalAutowiredAnnotationProcessor,org.springframework.context.annotation.internalRequiredAnnotationProcessor,org.springframework.context.annotation.internalCommonAnnotationProcessor,org.springframework.context.annotation.internalPersistenceAnnotationProcessor,org.springframework.aop.config.internalAutoProxyCreator,org.springframework.transaction.annotation.AnnotationTransactionAttributeSource#0,org.springframework.transaction.interceptor.TransactionInterceptor#0,org.springframework.transaction.config.internalTransactionAdvisor,transactionManager,org.springframework.context.annotation.ConfigurationClassPostProcessor.importAwareProcessor]; root of factory hierarchy

2012-11-11 20:36:06;D;a1s.learn.App;Context created

2012-11-11 20:36:06;D;a1s.learn.App;User service created

Hibernate: select user0\_.id as id1\_0\_, user0\_.name as name1\_0\_, user0\_.is\_active as is3\_1\_0\_, user0\_.is\_blocked as is4\_1\_0\_, user0\_.delete\_date as delete5\_1\_0\_, user0\_.is\_deleted as is6\_1\_0\_, user0\_.is\_hidden as is7\_1\_0\_, user0\_.is\_system as is8\_1\_0\_, user0\_.version as version1\_0\_ from users user0\_ where user0\_.id=?

User{id=1, name=Alexander Pupkin}

2012-11-11 20:36:06;D;a1s.learn.App;User found

Hibernate: select user0\_.id as id1\_0\_, user0\_.name as name1\_0\_, user0\_.is\_active as is3\_1\_0\_, user0\_.is\_blocked as is4\_1\_0\_, user0\_.delete\_date as delete5\_1\_0\_, user0\_.is\_deleted as is6\_1\_0\_, user0\_.is\_hidden as is7\_1\_0\_, user0\_.is\_system as is8\_1\_0\_, user0\_.version as version1\_0\_ from users user0\_ where user0\_.id=?

Hibernate: update users set name=?, is\_active=?, is\_blocked=?, delete\_date=?, is\_deleted=?, is\_hidden=?, is\_system=?, version=? where id=? and version=?

2012-11-11 20:36:06;D;a1s.learn.App;User changed

**Описание конфигурации logback**

По умолчанию logback ищет конфигурацию в файле logback.xml расположенный в CLASSPATH. В logback.xml описываются настройки логгеров, appender'ы и их связи.

Файл кофигурации выгялит так:

* <configuration /> - корневой элемент конфигурации
  + <logger /> - настройки логгера
  + <appender /> - настройки appender'а
  + <root /> - настройка связей между логгерами и appender'ами для корневого элемента

Рассмотрим элементы подробнее:

**<configuration /> и его атрибуты**

<configuration /> - корневой элемент конфигурации, в котором будут объявлены все дочерние элементы.

Атрибуты:

* scan="true" - сканировать файл конфигураций на наличие изменений
* scanPeriod="30 seconds" - период сканирования 1 раз в 30 секунд

**Appender'ы**

Appender'ы фактически являются адаптерами логгера для вывода данных логирования, например, FileAppender или ConsoleAppender.

Appender'ы описываются с помощью элемента <appender />.

Атрибуты:

* name - наименование appender'а
* class - класс apender'а

Внутрь элемента appender могут быть помещены дополнительные элементы настройки, например, <encoder />, который отвечает за шаблон вывода строки в лог.

Пример:

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

**Вывод на консоль**

<appender name="STDERR" class="ch.qos.logback.core.ConsoleAppender">

<target>System.err</target>

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

<withJansi>true</withJansi>

</appender>

Данный код создаёт appender по имени STDERR, который реализуется с помощью класса ch.qos.logback.core.ConsoleAppender. Вывод будет осуществляться в System.err (элемент <target />). Дополнительно вывод может быть раскрашен и для этого необходимо испольовать библиотеку Jansi для Windows.

Формат вывода строк:

<Дата> [<Имя потока>] <уровень логирования, поле 5 символов> <логгер> - <строка для вывода>

**Логгирование в разные файлы с использованием logback**

Логгирование в файл осуществляется с использованием класса ``. Кроме того, существует appender sift, который инкапсулирует в себя файловый appender и способен логгировать в разные файлы.

Настроим логгирование в разные файлы с использованием значения LOGGERID из MDC ([org.slf4j.MDC](http://www.slf4j.org/api/org/slf4j/MDC.html)). logback.xml

<appender name="SIFT" class="ch.qos.logback.classic.sift.SiftingAppender">

<discriminator>

<key>LOGGERID</key>

<defaultValue>unknown</defaultValue>

</discriminator>

<sift>

<appender name="/tmp/log-${LOGGERID}" class="ch.qos.logback.core.FileAppender">

<file>/tmp/log-${LOGGERID}.log</file>

<layout class="ch.qos.logback.classic.PatternLayout">

<pattern>%d [%thread] %level %mdc %logger{35} - %msg%n</pattern>

</layout>

</appender>

</sift>

</appender>

Рассмотрим элементы подробнее:

<discriminator>

<key>LOGGERID</key>

<defaultValue>unknown</defaultValue>

</discriminator>

описывает используемое свойство из MDC и его значение по умолчанию.

<file>/tmp/log-${LOGGERID}.log</file>

настраивает имя файла как /tmp/log-<значение свойства LOGGERID>.log.

Для задания свойства LOGGERID можно воспользоваться методом MDC::putProperty().

Для начал импортируем пространство имён:

import org.slf4j.MDC;

И пример файла с установкой свойства: App.java

package a1s.learn;

import a1s.learn.domain.User;

import a1s.learn.service.UserService;

import org.jboss.logging.Logger;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.slf4j.MDC;

public class App

{

private static Logger log;

public static void main( String[] args )

{

MDC.put("LOGGERID","superapplication");

log = Logger.getLogger(App.class);

ApplicationContext ctx = (ApplicationContext)new ClassPathXmlApplicationContext("app-context.xml");

MDC.put("LOGGERID", "context");

log.debug("Context created");

UserService userService=(UserService)ctx.getBean("userService");

MDC.put("LOGGERID", "service");

log.debug("User service created");

User us = userService.findById(1);

MDC.put("LOGGERID", "entity");

log.debug("User found");

userService.change(us);

MDC.put("LOGGERID", "service");

log.debug("User changed");

}

}

После выполнения данного кода в директории /tmp/ появятся следующие файлы логов:

$ ls log\*

log-context.log log-entity.log log-service.log log-superapplication.log