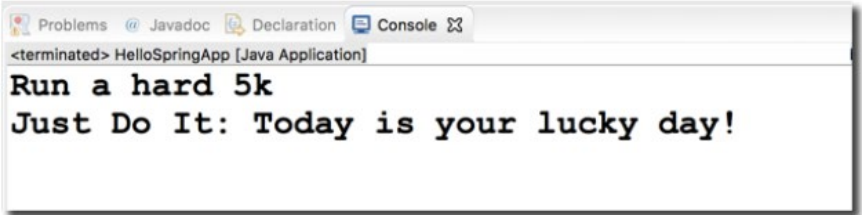
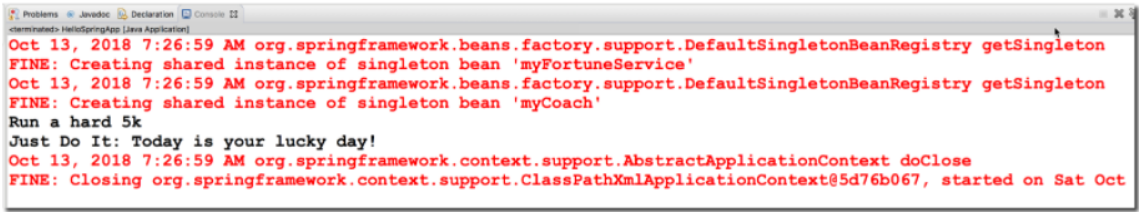
HEADS UP - Add Logging Messages in Spring 5.1

The Problem
In Spring 5.1, the Spring Development team changed the logging levels internally. As a result, by default you will no longer see the red logging messages at the INFO level. This is different than in the videos.



The Solution
If you would like to configure your app to show similar logging messages as in the video, you can make the following updates listed below. Note, you will not see the EXACT same messages, since the Spring team periodically changes the text of the internal logging messages. However, this should give you some additional logging data.



Overview of the steps
1. Create a bean to configure the parent logger and console handler

2. Configure the bean in the Spring XML config file

Detailed Steps

1. Create a bean to configure the parent logger and console handler

This class will set the parent logger level for the application context. It will also set the logging level for console handler. It sets the logger level to FINE. For more detailed logging info, you can set the logging level to level to FINEST. You can read more about the logging levels at

http://www.vogella.com/tutorials/Logging/article.html

This class also has an init method to handle the actual configuration. The init method is executed after the bean has been created and dependencies injected.

File: MyLoggerConfig.java

```
1.  package com.luv2code.springdemo;
2.
3.  import java.util.logging.ConsoleHandler;
4.  import java.util.logging.Level;
5.  import java.util.logging.Logger;
6.  import java.util.logging.SimpleFormatter;
7.
```

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MyLoggerConfig {

	private String rootLoggerLevel;
	private String printedLoggerLevel;

	public void setRootLoggerLevel(String rootLoggerLevel) {
		this.rootLoggerLevel = rootLoggerLevel;
	}

	public void setPrintedLoggerLevel(String printedLoggerLevel) {
		this.printedLoggerLevel = printedLoggerLevel;
	}

	public void initLogger() {

		// parse levels
		Level rootLevel = Level.parse(rootLoggerLevel);
		Level printedLevel = Level.parse(printedLoggerLevel);

		// get logger for app context
		Logger applicationContextLogger =
	Logger.getLogger(AnnotationConfigApplicationContext.class.getName());

		// get parent logger
		Logger loggerParent = applicationContextLogger.getParent();

		// set root logging level
		loggerParent.setLevel(rootLevel);

		// set up console handler
		ConsoleHandler consoleHandler = new ConsoleHandler();
		consoleHandler.setLevel(printedLevel);
		consoleHandler.setFormatter(new SimpleFormatter());

		// add handler to the logger
		loggerParent.addHandler(consoleHandler);
	}

}
```

2. Configure the bean in the Spring XML config file

In your XML config file, add the following bean entry. Make sure to ***list this as the first bean*** so that it is initialized first. Since the bean is initialized first, then you will get all of the logging traffic. If you move it later in the config file after the other beans, then you will miss out on some of the initial logging messages.

File: `applicationContext.xml` (snippet)

```
<!--
	Add a logger config to see logging messages.
	- For more detailed logs, set values to "FINEST"
	- For info on logging levels, see: http://www.vogella.com/tutorials/Logging/article.html
 -->
    <bean id="myLoggerConfig" class="com.luv2code.springdemo.MyLoggerConfig" init-
method="initLogger">
	<property name="rootLoggerLevel" value="FINE" />
	<property name="printedLoggerLevel" value="FINE"/>
    </bean>
```

Once you make these updates, then you will be able to see additional logging data.