Spring Android Reference Manual

Roy Clarkson

Spring Android Reference Manual

by Roy Clarkson

1.0.0.M1

Table of Contents

1. Spring Android Overview

1.1 Introduction

The Spring Android project supports the usage of the Spring Framework in an Android environment. This includes the ability to use RestTemplate as the REST client for your Android applications.

2. Spring Android Commons Logging Module

2.1 Introduction

Android provides its own logging API for use in Android applications. However, many Java libraries, including Spring, use a logging abstraction such as Commons Logging or Simple Logging Facade for Java (SLF4j). The Spring Android Commons Logging Module provides a implementation of the Commons Logging API that bridges onto the Android logging system. This allows Java libraries that use Common Logging to run in an Android environment.

2.2 How to get

The spring-android-commons-logging module is required to use Spring Android, since the Spring Framework itself requires Commons Logging. In general, rely on transitive dependency resolution to include this module in your classpath. We do not recommend using the Commons Logging API directly in your own Android applications; rather, use this module simply to adapt existing code onto the Android logging system. The definition of the module artifact is provided for reference below:

```
<dependency>
    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-commons-logging</artifactId>
    <version>${org.springframework.android-version}</version>
</dependency>
```

2.3 Log Level mapping

The Commons Logging API nearly maps directly onto the Android Logging API. There is one difference. The Commons Logging FATAL level maps to the ERROR Android level, since Android has no FATAL level.

3. Spring Android Rest Template Module

3.1 Introduction

Spring's RestTemplate is a robust, popular Java-based REST client. The Spring Android Rest Template Module provides a version of RestTemplate that works in an Android environment.

3.2 How to get

Add the spring-android-rest-template artifact to your classpath:

```
<dependency>
    <groupId>org.springframework.android</groupId>
    <artifactId>spring-android-rest-template</artifactId>
    <version>${org.springframework.android-version}</version>
</dependency>
```

Doing so will transitively include the spring-android-commons-logging module.

At the moment, commons-httpclient 3.x is also required for RestTemplate to work on Android without further customization:

```
<dependency>
    <groupId>commons-httpclient</groupId>
    <artifactId>commons-httpclient</artifactId>
    <version>3.1</version>
</dependency>
```

In a future release, Spring Android will provide support for Http Client 4, which is used on Android by default.

3.3 Basic usage example

Using Rest Template, it's easy to invoke RESTful APIs. Below are several usage examples.

The following example shows a query to google for the search term "Thanksgiving".

```
RestTemplate restTemplate = new RestTemplate();
restTemplate.setRequestFactory(new CommonsClientHttpRequestFactory());
String url = "https://ajax.googleapis.com/ajax/services/search/web?v=1.0&q={query}";
String result = restTemplate.getForObject(url, String.class, "Thanksgiving");
```

3.4 Usage example: reading JSON data

Alternatively, suppose you have defined a Java object you wish to populate from a RESTful web request that returns JSON content.

Define your object based on the JSON data being returned from the RESTful request:

```
public class Event {

    private Long id;

    private String title;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public String setTitle(String title) {
        this.title = title;
    }
}
```

Make the RestTemplate request:

```
RestTemplate restTemplate = new RestTemplate();
restTemplate.setRequestFactory(new CommonsClientHttpRequestFactory());
String url = "https://mypretendservice.com/events";
Event[] events = restTemplate.getForObject(url, Event[].class);
```