

# Spring into Mobile Application Development

Roy Clarkson and Keith Donald



# Agenda

- Introduction
- Spring Android
- Spring Mobile
- Discussion and Questions

# Introduction

---

- **What is the purpose of the Spring Android and Spring Mobile projects?**
  - Spring Android provides support for building native Android applications utilizing Spring technologies, where applicable
  - In contrast, Spring Mobile provides support for building mobile web applications

# Spring Android

- **Define the Problem**
- **Review of REST**
- **Basic Rest Template Example**
- **Spring Android Rest Template Overview**
- **Maven Can Help**
- **Spring Android Showcase and Demos**

# What problem are we trying to solve?

---

## ■ Concerns

- REST has become a popular choice for architecting both public and private web services
- The Android runtime provides HTTP clients capable of making HTTP connections and requests, but it does not have a fully featured REST client

## ■ Spring Android Solution

- The goal of Spring Android Rest Template is to provide an easy to use, and functional REST client that supports marshaling objects from XML and JSON.

## ■ Origin

- The term Representational State Transfer was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation.

## ■ His paper suggests these four design principles:

- Use HTTP methods explicitly.
  - POST, GET, PUT, DELETE
  - CRUD operations can be mapped to these existing methods
- Be stateless.
  - State dependencies limit or restrict scalability
- Expose directory structure-like URIs.
  - URI's should be easily understood
- Transfer XML, JavaScript Object Notation (JSON), or both.
  - Use XML or JSON to represent data objects or attributes

# Basic Rest Template Example

---

## ■ Google search example

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

## ■ Multiple parameters

```
RestTemplate restTemplate = new RestTemplate();  
String url = "http://example.com/hotels/{hotel}/bookings/{booking}";  
String result = restTemplate.getForObject(url, String.class, "42", "21");
```

# Google Search Demo

# Spring Android Rest Template

---

## ■ Based on SpringFramework

- The majority of the supporting classes are pulled from SpringFramework.
- Modifications were made to support Android.

## ■ RestTemplate class is the heart the library

- Entry points for the six main HTTP methods
  - DELETE - delete(...)
  - GET - getForObject(...)
  - HEAD - headForHeaders(...)
  - OPTIONS - optionsForAllow(...)
  - POST - postForLocation(...)
  - PUT - put(...)
  - any HTTP operation - exchange(...) and execute(...)

# Spring Android Rest Template

---

## ■ Http Client

- The HttpComponents HttpClient is a native HTTP client available on the Android platform.
- HttpComponentsClientHttpRequestFactory

## ■ Message Converters

- MappingJacksonHttpMessageConverter - object to JSON marshaling supported via the Jackson JSON Processor
- SimpleXmlHttpMessageConverter - object to XML marshaling supported via the Simple XML Serializer
- SyndFeedHttpMessageConverter - RSS and Atom feeds supported via the Android ROME Feed Reader

# How can Maven help?

---

## ■ Android4Maven

- This project compiles android.jar from source and pulls out source and resource files to replicate android.jar in the SDK
- <http://sourceforge.net/projects/android4maven/>

## ■ Maven Android SDK Deployer

- If you need to use Google maps, then you have to go this route
- <https://github.com/mosabua/maven-android-sdk-deployer>

## ■ Maven Android Plugin

- Provides support for Maven dependency management within Android projects
- <http://code.google.com/p/maven-android-plugin/>

# Maven Android Plugin Configuration

```
<build>
  <sourceDirectory>src</sourceDirectory>
  <finalName>${project.artifactId}</finalName>
  <plugins>
    <plugin>
      <groupId>com.jayway.maven.plugins.android.generation2</groupId>
      <artifactId>maven-android-plugin</artifactId>
      <version>2.8.4</version>
      <configuration>
        <sdk>
          <platform>3</platform>
        </sdk>
        <emulator>
          <avd>3</avd>
        </emulator>
        <deleteConflictingFiles>true</deleteConflictingFiles>
        <undeployBeforeDeploy>true</undeployBeforeDeploy>
      </configuration>
      <extensions>true</extensions>
    </plugin>
    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.3.2</version>
    </plugin>
  </plugins>
</build>
```

## ■ Maven Integration for Android Development Tools

- An Eclipse plugin that adds support for integrating m2eclipse, Android Developer Tools, and the Maven Android Plugin
- <http://code.google.com/a/eclipselabs.org/p/m2eclipse-android-integration/>

## ■ Maven Android archetypes

- This projects provides several Maven archetypes for Android. These archetypes allow you to quickly bootstrap a Maven project to develop an android application.
- <https://github.com/akquinet/android-archetypes>

# Spring Android Showcase

---

## ■ Examples

- HTTP GET
  - JSON
  - XML
- HTTP GET with Parameters
  - JSON
  - XML
- HTTP POST
  - String
  - JSON
  - XML
  - MultiValueMap
- RSS
- ATOM

# Rest Template Demos

# Spring Social on Android

---

- **What are you talking about? Keith didn't mention this in the last presentation!**
  - That is true. Surprise! :)
- **What's new?**
  - Updated Spring Android Rest Template to Spring 3.1.0.M1 release.
  - Added a new library: Spring Android Auth
- **When will the M3 release be available?**
  - Soon.
- **Stop stalling and show us a demo!**
  - OK.

# Spring Mobile

- **Provides support for developing mobile web applications**

- Builds on Spring MVC, focuses on server-side support
- Compliments client-side mobile frameworks

- **Key Features**

- Device Detection
- Site Preference Management
- Site Switcher

# Device Detection

---

- **Useful when requests by mobile devices need to be handled differently from requests made by desktop browsers**
- **Introspects HTTP requests to determine the device that originated the request.**
  - Achieved by analyzing the User-Agent header and other request headers
  - In contrast to “Feature Detection” where client detects available features
- **Spring Mobile provides a DeviceResolver abstraction and interceptor**
- **Supported DeviceResolver implementations**
  - LiteDeviceResolver
  - WurflDeviceResolver

# Device Detection Demo

# Site Preference Management

---

- **Device detection is often used to determine which "site" will be served to the user**
  - Mobile site vs. desktop site
- **Spring Mobile also provides support for "site preference management"**
- **Allows the user to indicate whether he or she prefers the mobile site or the normal site**
- **Remembers the user's preference for their session**

# Site Preference Demo

# Site Switcher

---

- **Some applications may wish to host their "mobile site" at a different domain from their "normal site"**
  - For example, Google will switch you to m.google.com if you access google.com from your mobile phone
- **SiteSwitcherHandlerInterceptor can be used to redirect mobile users to a dedicated mobile site**
- **Supported SiteSwitchers**
  - mDot - m.example.com
  - dotMobi - example.mobi

# Site Switcher Demo

# Additional Resources

---

## ■ Project Home

- <http://www.springsource.org/spring-android>
- <http://www.springsource.org/spring-mobile>

## ■ Sample Code

- <git://git.springsource.org/spring-mobile/samples.git>
- <git://git.springsource.org/greenhouse/greenhouse.git>

## ■ Blog Posts

- <http://blog.springsource.com/author/rclarkson/>
- <http://blog.springsource.com/author/keithd/>
- <http://blog.springsource.com/author/cwalls/>

# Future Roadmap

---

- **Samples illustrating integration with popular client-side mobile web frameworks**
  - jQuery Mobile
  - Sencha Touch
  - Sprout
  
- **Samples demonstrating the native “hybrid” model**
  - PhoneGap integration