



Spring Social LinkedIn Reference Manual

1.0.0.RC3

Craig Walls , Keith Donald

Copyright © 2011-2013

Copies of this document may be made for your own use and for distribution to others, provided that you do not charge any fee for such copies and further provided that each copy contains this Copyright Notice, whether distributed in print or electronically.

Table of Contents

| | |
|--|---|
| 1. Spring Social LinkedIn Overview | 1 |
| 1.1. Introduction | 1 |
| 1.2. How to get | 1 |
| 2. Configuring LinkedIn Connectivity | 2 |
| 3. LinkedIn API Binding | 4 |
| 3.1. Retrieving a user's LinkedIn profile data | 4 |
| 3.2. Getting a user's LinkedIn connections | 5 |

1. Spring Social LinkedIn Overview

1.1 Introduction

The Spring Social LinkedIn project is an extension to [Spring Social](#) that enables integration with LinkedIn.

[LinkedIn](#) is a social networking site geared toward professionals. It enables its users to maintain and correspond with a network of contacts they have are professionally linked to.

Spring Social LinkedIn enables integration with LinkedIn with `LinkedInConnectionFactory`, a connection factory that can be plugged into Spring Social's service provider connection framework, and with an API binding to LinkedIn's REST API.

1.2 How to get

The following Maven dependency will add Spring Social LinkedIn to your project:

```
<dependency>
  <groupId>org.springframework.social</groupId>
  <artifactId>spring-social-linkedin</artifactId>
  <version>${org.springframework.social-linkedin-version}</version>
</dependency>
```

As an extension to Spring Social, Spring Social LinkedIn depends on Spring Social. Spring Social's core module will be transitively resolved from the Spring Social LinkedIn dependency. If you'll be using Spring Social's web module, you'll need to add that dependency yourself:

```
<dependency>
  <groupId>org.springframework.social</groupId>
  <artifactId>spring-social-web</artifactId>
  <version>${org.springframework.social-version}</version>
</dependency>
```

Note that Spring Social LinkedIn may release on a different schedule than Spring Social. Consequently, Spring Social's version may differ from that of Spring Social LinkedIn.

Spring Social LinkedIn uses Spring Social's `OAuth1Template` to add OAuth 1.0a authorization headers to requests sent to LinkedIn. `OAuth1Template` uses the [Commons Codec](#) library when calculating request signatures. Therefore, you'll also need Commons Codec in your project:

```
<dependency>
  <groupId>commons-codec</groupId>
  <artifactId>commons-codec</artifactId>
  <version>1.5</version>
</dependency>
```

Consult [Spring Social's reference documentation](#) for more information on Spring Social dependencies.

2. Configuring LinkedIn Connectivity

Spring Social's `ConnectController` works with one or more provider-specific `ConnectionFactory`s to exchange authorization details with the provider and to create connections. Spring Social LinkedIn provides `LinkedInConnectionFactory`, a `ConnectionFactory` for creating connections with LinkedIn.

So that `ConnectController` can find `LinkedInConnectionFactory`, it must be registered with a `ConnectionFactoryRegistry`. The following class constructs a `ConnectionFactoryRegistry` containing a `ConnectionFactory` for LinkedIn using Spring's Java configuration style:

```
@Configuration
public class SocialConfig {

    @Bean
    public ConnectionFactoryLocator connectionFactoryLocator() {
        ConnectionFactoryRegistry registry = new ConnectionFactoryRegistry();
        registry.addConnectionFactory(new LinkedInConnectionFactory(
            environment.getProperty("linkedin.consumerKey"),
            environment.getProperty("linkedin.consumerSecret")));
        return registry;
    }
}
```

Here, a LinkedIn connection factory is registered with `ConnectionFactoryRegistry` via the `addConnectionFactory()` method. If we wanted to add support for connecting to other providers, we would simply register their connection factories here in the same way as `LinkedInConnectionFactory`.

Because consumer keys and secrets may be different across environments (e.g., test, production, etc) it is recommended that these values be externalized. As shown here, Spring 3.1's `Environment` is used to look up the application's consumer key and secret.

Optionally, you may also configure `ConnectionFactoryRegistry` and `LinkedInConnectionFactory` in XML:

```
<bean id="connectionFactoryLocator" class="org.springframework.social.connect.support.ConnectionFactoryRegistry">
    <property name="connectionFactories">
        <list>

            <bean class="org.springframework.social.linkedin.connect.LinkedinConnectionFactory">
                <constructor-arg value="${linkedin.consumerKey}" />
                <constructor-arg value="${linkedin.consumerSecret}" />
            </bean>
        </list>
    </property>
</bean>
```

This is functionally equivalent to the Java-based configuration of `ConnectionFactoryRegistry` shown before. The only casual difference is that the connection factories are injected as a list into the

`connectionFactories` property rather than with the `addConnectionFactory()` method. As in the Java-based configuration, the application's consumer key and secret are externalized (shown here as property placeholders).

Refer to [Spring Social's reference documentation](#) for complete details on configuring `ConnectController` and its dependencies.

3. LinkedIn API Binding

Spring Social LinkedIn offers integration with LinkedIn's REST API with the `LinkedIn` interface and its implementation, `LinkedInTemplate`.

To create an instance of `LinkedInTemplate`, you may pass in your application's OAuth 1 credentials, along with an access token/secret pair to the constructor:

```
String consumerKey = "..."; // The application's consumer key
String consumerSecret = "..."; // The application's consumer secret
String accessToken = "..."; // The access token granted after OAuth authorization
String accessTokenSecret = "..."; // The access token secret granted after OAuth
authorization
LinkedIn linkedin = new LinkedInTemplate(consumerKey, consumerSecret, accessToken,
    accessTokenSecret);
```

If you are using Spring Social's [service provider framework](#), you can get an instance of `LinkedIn` from a `Connection`. For example, the following snippet calls `getApi()` on a connection to retrieve a `LinkedIn`:

```
Connection<LinkedIn> connection =
    connectionRepository.findPrimaryConnection(LinkedIn.class);
if (connection != null) {
    LinkedIn linkedin = connection.getApi();

    // ... use LinkedIn API binding
}
```

Here, `ConnectionRepository` is being asked for the primary connection that the current user has with LinkedIn. If a connection to LinkedIn is found, it retrieves a `LinkedIn` instance that is configured with the connection details received when the connection was first established.

Once you have a `LinkedIn` you can use it to interact with LinkedIn on behalf of the user who the access token was granted for.

3.1 Retrieving a user's LinkedIn profile data

To retrieve the authenticated user's profile data, call the `getUserProfile()` method:

```
LinkedInProfile profile = linkedin.getUserProfile();
```

The data returned in the `LinkedInProfile` includes the user's LinkedIn ID, first and last names, their "headline", the industry they're in, and URLs for the public and standard profile pages.

If it's only the user's LinkedIn ID you need, then you can get that by calling the `getProfileId()` method:

```
String profileId = linkedin.getProfileId();
```

Or if you only need a URL for the user's public profile page, call `getProfileUrl()`:

```
String profileUrl = linkedin.getProfileUrl();
```

3.2 Getting a user's LinkedIn connections

To retrieve a list of LinkedIn users to whom the user is connected, call the `getConnections()` method:

```
List<LinkedInProfile> connections = linkedin.getConnections();
```

This will return a list of `LinkedInProfile` objects for the user's 1st-degree network (those LinkedIn users to whom the user is directly linked--not their extended network).