

Spring Social GitHub Reference Manual

**Craig Walls
Keith Donald**

Spring Social GitHub Reference Manual

by Craig Walls and Keith Donald

1.0.0.M4

© SpringSource Inc., 2013

Table of Contents

| | |
|---|---|
| 1. Spring Social GitHub Overview | 1 |
| 1.1. Introduction | 1 |
| 1.2. How to get | 1 |
| 2. Configuring GitHub Connectivity | 2 |
| 3. GitHub API Binding | 4 |
| 3.1. Retrieving a GitHub user's profile | 4 |

1. Spring Social GitHub Overview

1.1 Introduction

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

Although many developers think of [GitHub](#) as Git-based source code hosting, the tagline in GitHub's logo clearly states that GitHub is about "social coding". GitHub is a social network that links developers together and with the projects they follow and/or contribute to.

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

1.2 How to get

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

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

As an extension to Spring Social, Spring Social GitHub depends on Spring Social. Spring Social's core module will be transitively resolved from the Spring Social GitHub 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 GitHub may release on a different schedule than Spring Social. Consequently, Spring Social's version may differ from that of Spring Social GitHub.

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

2. Configuring GitHub 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 GitHub provides `GitHubConnectionFactory`, a `ConnectionFactory` for creating connections with GitHub.

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

```
@Configuration
public class SocialConfig {

    @Bean
    public ConnectionFactoryLocator connectionFactoryLocator() {
        ConnectionFactoryRegistry registry = new ConnectionFactoryRegistry();
        registry.addConnectionFactory(new GitHubConnectionFactory(
            environment.getProperty("github.clientId"),
            environment.getProperty("github.clientSecret")));
        return registry;
    }
}
```

Here, a GitHub 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 `GitHubConnectionFactory`.

Because client IDs 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 client ID and secret.

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

```
<bean id="connectionFactoryLocator" class="org.springframework.social.connect.support.ConnectionFactoryRegistry">
    <property name="connectionFactories">
        <list>
            <bean class="org.springframework.social.github.connect.GitHubConnectionFactory">
                <constructor-arg value="${github.clientId}" />
                <constructor-arg value="${github.clientSecret}" />
            </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. GitHub API Binding

Spring Social GitHub's `GitHub` interface and its implementation, `GitHubTemplate`, offer an API binding to GitHub's REST API.

To obtain an instance of `GitHubTemplate`, you can instantiate it by passing an authorized access token to its constructor:

```
String accessToken = "f8FX29g..."; // access token received from GitHub after OAuth authorization
GitHub github = new GitHubTemplate(accessToken);
```

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

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

    // ... use GitHub API binding
}
```

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

With a `GitHub` in hand, there are a handful of operations it provides to interact with GitHub on behalf of the user. These will be covered in the following sections.

3.1 Retrieving a GitHub user's profile

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

```
GitHubUserProfile profile = github.getUserProfile();
```

The `GitHubUserProfile` returned from `getUserProfile()` includes several useful pieces of information about the user, including their...

- Name
- Username (ie, login name)
- Company
- Email address

- Location
- Blog URL
- Date they joined GitHub

If all you need is the user's GitHub username, you can get that by calling the `getProfileId()` method:

```
String username = github.getProfileId();
```

And if you need a URL to the user's GitHub profile page, you can use the `getProfileUrl()` method:

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