Making Retrofit Work For You

Jake Wharton

Retrofit

```
interface ApiService {
}
```

Retrofit

```
interface ApiService {
  List<User> search(
      String query,
      SortOrder order);
}
```

Retrofit

```
interface ApiService {
  List<User> search(
      String query,
      SortOrder order);
}

class HttpApiService implements ApiService {
  // ...
}
```

Retrofit

```
interface ApiService {
  List<User> search(
      String query,
      SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search")
  List<User> search(
      String query,
      SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search")
  List<User> search(
      @Query("q") String query,
      SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search")
  List<User> search(
      @Query("q") String query,
      @Query("sort") SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search")
  Call<List<User>> search(
      @Query("q") String query,
      @Query("sort") SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search/{category}")
  Call<List<User>> search(
      @Path("category") String category,
      @Query("q") String query,
      @Query("sort") SortOrder order);
}
```

Retrofit

```
interface ApiService {
  @GET("/search/{category}")
  Call<List<User>> search(
      @Path("category") String category,
      @Query("q") String query,
      @Query("sort") SortOrder order);

  @POST("/upload/image")
  Call<Void> uploadImage(
      @Body Image image);
}
```

Retrofit

```
interface ApiService {
  @GET("/search/{category}")
  Call<List<User>> search(
      @Path("category") String category,
      @Query("q") String query,
      @Query("sort") SortOrder order);

  @POST("/upload/image")
  @Headers("SomeHeader: SomeValue")
  Call<Void> uploadImage(
      @Body Image image);
}
```

Retrofit

```
interface ApiService {
  // ...
}
```

Retrofit

```
interface ApiService {
  // ...
}

Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .build();
```

Retrofit

```
interface ApiService {
  // ...
}

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

ApiService service =
    retrofit.create(ApiService.class);
```

Retrofit

```
interface ApiService {
  // ...
}

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

ApiService service =
    retrofit.create(ApiService.class);
```

HTTP Client

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();
```

HTTP Client

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();
Foo foo = retrofit.create(Foo.class);
```

HTTP Client

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();
Foo foo = retrofit.create(Foo.class);
```

```
Bar bar = retrofit.create(Bar.class);
```

HTTP Client

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .build();
Foo foo = retrofitFoo.create(Foo.class);
```

```
Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .build();
Bar bar = retrofitBar.create(Bar.class);
```

HTTP Client

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .client(new OkHttpClient())
    .build();
Foo foo = retrofitFoo.create(Foo.class);

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .client(new OkHttpClient())
    .build();
Bar bar = retrofitBar.create(Bar.class);
```

HTTP Client

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .client(new OkHttpClient())
    .build();
Foo foo = retrofitFoo.create(Foo.class);

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .client(new OkHttpClient())
    .build();
Bar bar = retrofitBar.create(Bar.class);
```

HTTP Client

```
OkHttpClient client = new OkHttpClient();

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .client(client)
    .build();
Foo foo = retrofitFoo.create(Foo.class);

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .client(client)
    .build();
Bar bar = retrofitBar.create(Bar.class);
```

HTTP Client

```
OkHttpClient client = new OkHttpClient();

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .client(client)
    .build();
Foo foo = retrofitFoo.create(Foo.class);

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .client(client)
    .build();
Bar bar = retrofitBar.create(Bar.class);
```

HTTP Client

```
OkHttpClient client = new OkHttpClient();
```

HTTP Client

```
OkHttpClient client = new OkHttpClient();

OkHttpClient clientFoo = client.newBuilder()
    .addInterceptor(new FooInterceptor())
    .build();
```

HTTP Client

```
OkHttpClient client = new OkHttpClient();

OkHttpClient clientFoo = client.newBuilder()
    .addInterceptor(new FooInterceptor())
    .build();

OkHttpClient clientBar = client.newBuilder()
    .readTimeout(30, SECONDS)
    .writeTimeout(30, SECONDS)
    .build();
```

HTTP Client

```
interface Service {
  @GET("/user")
  Call<User> user();

  @POST("/login")
  Call<User> login(@Body LoginRequest request);

  @GET("/logout")
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user") // Requires authentication.
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @GET("/logout") // Requires authentication.
  Call<Void> logout();
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();
    return chain.proceed(request);
  }
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (!request.url().encodedPath().equals("/login")) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (!request.url().encodedPath().equals("/login")) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

HTTP Client

```
interface Service {
  @GET("/user") // Requires authentication.
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @GET("/logout") // Requires authentication.
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user") // Requires authentication.
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword") // No authentication.
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout") // Requires authentication.
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user") // Requires authentication.
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword") // No authentication.
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout") // Requires authentication.
  Call<Void> logout();
}
```

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (!request.url().encodedPath().equals("/login")) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

HTTP Client

```
interface Service {
  @GET("/user") // Requires authentication.
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword") // No authentication.
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout") // Requires authentication.
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user")
  Call<User> user();

  @POST("/login") // No authentication.
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword") // No authentication.
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout")
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user")
  Call<User> user();

  @POST("/login")
  @Headers("No-Authentication: true")
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword")
  @Headers("No-Authentication: true")
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout")
  Call<Void> logout();
}
```

HTTP Client

```
interface Service {
  @GET("/user")
  Call<User> user();

  @POST("/login")
  @Headers("No-Authentication: true")
  Call<User> login(@Body LoginRequest request);

  @POST("/forgotPassword")
  @Headers("No-Authentication: true")
  Call<Void> forgotPassword(
      @Body ForgotPasswordRequest request);

  @GET("/logout")
  Call<Void> logout();
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (!request.url().encodedPath().equals("/login")) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (request.header("No-Authentication") == null) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

HTTP Client

```
class ServiceInterceptor implements Interceptor {
  @Override public Response intercept(Chain chain) {
    Request request = chain.request();

    if (request.header("No-Authentication") == null) {
      request = request.newBuilder()
          .addHeader("Authorization", "hunter2")
          .build();
    }

    return chain.proceed(request);
  }
}
```

Converters

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

interface Service {
  @GET("/user")
  Call<ResponseBody> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<ResponseBody> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<ResponseBody> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();
```

Converters

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();
```

Converters

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();
```

Converters

```
Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();
```

Converters

```
GsonConverterFactory gsonFactory =
    GsonConverterFactory.create();

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(gsonFactory)
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(gsonFactory)
    .build();
```

Converters

```
GsonConverterFactory gsonFactory =
    GsonConverterFactory.create();

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(gsonFactory)
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(gsonFactory)
    .build();
```

Converters

```
GsonConverterFactory gsonFactory =
    GsonConverterFactory.create();

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(gsonFactory)
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(gsonFactory)
    .build();
```

Converters

```
Gson gson = new Gson();
GsonConverterFactory gsonFactory =
    GsonConverterFactory.create(gson);

Retrofit retrofitFoo = new Retrofit.Builder()
    .baseUrl("http://foo.example.com")
    .addConverterFactory(gsonFactory)
    .build();

Retrofit retrofitBar = new Retrofit.Builder()
    .baseUrl("http://bar.example.com")
    .addConverterFactory(gsonFactory)
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(
          GsonConverterFactory.create())
      .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user(); // <-- proto

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user(); // <-- proto

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())  User ?
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())  User ✓
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create()) Friends ?
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create()) Friends ✗
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create()) Friends ✗
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create()) Friends ✗
    .addConverterFactory(
        GsonConverterFactory.create())  Friends ?
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create()) Friends ✗
    .addConverterFactory(
        GsonConverterFactory.create())  Friends ✓
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .addConverterFactory(
        ProtoConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        GsonConverterFactory.create())
    .addConverterFactory(
        ProtoConverterFactory.create()) // No!
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        ProtoConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();

  @GET("/friends")
  Call<Friends> friends();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        SimpleXmlConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user(); // <-- xml

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        SimpleXmlConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user(); // <-- xml

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}
```

Converters

```
interface Service {
  @GET("/user")
  Call<User> user(); // <-- xml

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}
```

Converters

```
interface Service {
  @GET("/user")
  Call<User> user(); // <-- xml

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}

@interface Xml {}
```

Converters

```
interface Service {
  @GET("/user")
  Call<User> user(); // <-- xml

  @GET("/friends")
  Call<Friends> friends(); // <-- json
}

@interface Xml {}
@interface Json {}
```

Converters

```
interface Service {
  @GET("/user") @Xml
  Call<User> user(); // <-- xml

  @GET("/friends") @Json
  Call<Friends> friends(); // <-- json
}

@interface Xml {}
@interface Json {}
```

Converters

```
interface Service {
  @GET("/user") @Xml
  Call<User> user(); // <-- xml

  @GET("/friends") @Json
  Call<Friends> friends(); // <-- json
}

@interface Xml {}
@interface Json {}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        SimpleXmlConverterFactory.create())
    .addConverterFactory(
        GsonConverterFactory.create())
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        new XmlOrJsonConverterFactory())
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        new XmlOrJsonConverterFactory())
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        new XmlOrJsonConverterFactory())
    .build();

class XmlOrJsonConverterFactory
    extends Converter.Factory {
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(
          new XmlOrJsonConverterFactory())
      .build();

class XmlOrJsonConverterFactory
      extends Converter.Factory {

  final Converter.Factory xml =
      SimpleXmlConverterFactory.create();

}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(
          new XmlOrJsonConverterFactory())
      .build();

class XmlOrJsonConverterFactory
      extends Converter.Factory {

  final Converter.Factory xml =
      SimpleXmlConverterFactory.create();
  final Converter.Factory json =
      GsonConverterFactory.create();

}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    for (Annotation annotation : annotations) {
    }
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    for (Annotation annotation : annotations) {
      if (annotation.getClass() == Xml.class) {
      }
    }
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    for (Annotation annotation : annotations) {
      if (annotation.getClass() == Xml.class) {
        return xml.responseBodyConverter(
            type, annotations, retrofit);
      }
    }
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations, Retrofit retrofit) {
    for (Annotation annotation : annotations) {
      if (annotation.getClass() == Xml.class) {
        return xml.responseBodyConverter(
            type, annotations, retrofit);
      }
      if (annotation.getClass() == Json.class) {
        return json.responseBodyConverter(
            type, annotations, retrofit);
      }
    }
  }
}
```

Converters

```
class XmlOrJsonConverterFactory extends Converter.Factory {
  final Converter.Factory xml = // ...
  final Converter.Factory json = // ...

  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations, Retrofit retrofit) {
    for (Annotation annotation : annotations) {
      if (annotation.getClass() == Xml.class) {
        return xml.responseBodyConverter(
            type, annotations, retrofit);
      }
      if (annotation.getClass() == Json.class) {
        return json.responseBodyConverter(
            type, annotations, retrofit);
      }
    }
    return null;
  }
}
```

Converters

```
class AnnotatedConverterFactory extends Converter.Factory {
  final Map<Class<?>, Converter.Factory> factories;

  AnnotationConverterFactory(
      Map<Class<?>, Converter.Factory> factories) {
    this.factories = new LinkedHashMap<>(factories);
  }

  @Override public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations, Retrofit retrofit) {
    for (Annotation annotation : annotations) {
      Converter.Factory factory =
          factories.get(annotation.getClass());
      if (factory != null) {
        return factory.responseBodyConverter(
            type, annotations, retrofit);
      }
    }
    return null;
  }
}
```

Converters

```
class AnnotatedConverterFactory extends Converter.Factory {
  // ...
}
```

Converters

```
class AnnotatedConverterFactory extends Converter.Factory {
  // ...

  class Builder {
    Map<Class<?>,Converter.Factory> factories = new LinkedHashMap<>()

    Builder add(Class<? extends Annotation> cls,
        Converter.Factory factory) {
      if (cls == null) throw new NullPointerException("cls");
      if (factory == null) throw new NullPointerException("factory");
      factories.add(cls, factory);
      return this;
    }

    AnnotationConverterFactory build() {
      return new AnnotatedConverterFactory(factories);
    }
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        new XmlOrJsonConverterFactory())
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(
        AnnotatedConverterFactory.builder()
            .add(Xml.class,
                SimpleXmlConverterFactory.create())
            .add(Json.class,
                GsonConverterFactory.create())
            .build())
    .build();
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
{
  "id": "1345235987",
  "name": "Shrek",
  "location": "Swamp"
}
```

Converters


```
{
  "meta": {
    "code": 200,
    "time": 287,
  },
  "notifications": {},
  "response": {
    "id": "1345235987",
    "name": "Shrek",
    "location": "Swamp"
  }
}
```

Converters


```
{
  "meta": {
    "code": 200,
    "time": 287,
  },
  "notifications": {},
  "response": {
    "id": "1345235987",
    "name": "Shrek",
    "location": "Swamp"
  }
}
```

Converters


```
{
  "meta": {
    "code": 200,
    "time": 287,
  },
  "notifications": {},
  "response": {
    "id": "1345235987",
    "name": "Shrek",
    "location": "Swamp"
  }
}
```

Converters


```
{
  "meta": {
    "code": 200,
    "time": 287,
  },
  "notifications": {},
  "response": {
    "id": "1345235987",
    "name": "Shrek",
    "location": "Swamp"
  }
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<Envelope<User>> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<Envelope<User>> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody,?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
    };
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
      }
    };
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    final Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        Envelope<?> envelope = delegate.convert(body);
      }
    };
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    final Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        Envelope<?> envelope = delegate.convert(body);
        return envelope.response;
      }
    };
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    final Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        Envelope<?> envelope = delegate.convert(body);
        return envelope.response;
      }
    };
  }
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<Envelope<User>> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(new EnvelopingConverter())
      .addConverterFactory(GsonConverterFactory.create())
      .build();

interface Service {
  @GET("/user")
  Call<Envelope<User>> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(new EnvelopingConverter())
      .addConverterFactory(GsonConverterFactory.create())
      .build();

interface Service {
  @GET("/user")
  Call<User> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addConverterFactory(new EnvelopingConverter())
    .addConverterFactory(GsonConverterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}

class Envelope<T> {
  Meta meta;
  List<Notification> notifications;
  T response;
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations, Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    final Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        Envelope<?> envelope = delegate.convert(body);
        return envelope.response;
      }
    };
  }
}
```

Converters

```
class EnvelopingConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations, Retrofit retrofit) {
    Type envelopedType =
        TypeToken.getParameterized(Envelope.class, type)
            .getType();
    final Converter<ResponseBody, Envelope<?>> delegate =
        retrofit.nextResponseBodyConverter(
            this, envelopedType, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        Envelope<?> envelope = delegate.convert(body);
        // Handle notifications, record timing information.
        return envelope.response;
      }
    };
  }
}
```

Converters

```
class EmptyToNullConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
  }
}
```

Converters

```
class EmptyToNullConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Converter<ResponseBody, ?> delegate =
        retrofit.nextResponseBodyConverter(
            this, type, annotations);
  }
}
```

Converters

```
class EmptyToNullConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Converter<ResponseBody, ?> delegate =
        retrofit.nextResponseBodyConverter(
            this, type, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
      }
    };
  }
}
```

Converters

```
class EmptyToNullConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    Converter<ResponseBody, ?> delegate =
        retrofit.nextResponseBodyConverter(
            this, type, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        if (body.contentLength() == 0) return null;
      }
    };
  }
}
```

Converters

```
class EmptyToNullConverter extends Converter.Factory {
  @Override
  public Converter<ResponseBody, ?> responseBodyConverter(
      Type type, Annotation[] annotations,
      Retrofit retrofit) {
    final Converter<ResponseBody, ?> delegate =
        retrofit.nextResponseBodyConverter(
            this, type, annotations);
    return new Converter<ResponseBody, ?>() {
      @Override public Object convert(ResponseBody body) {
        if (body.contentLength() == 0) return null;
        return delegate.convert(body);
      }
    };
  }
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(new EnvelopingConverter())
      .addConverterFactory(GsonConverterFactory.create())
      .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

```
Retrofit retrofit = new Retrofit.Builder()
      .baseUrl("http://example.com")
      .addConverterFactory(new EmptyToNullConverter())
      .addConverterFactory(new EnvelopingConverter())
      .addConverterFactory(GsonConverterFactory.create())
      .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Converters

Converters

```
interface DribbbleSearchService {
  @GET("search")
  Call<List<Shot>> search(
      @Query("q") String query,
      @Query("page") Integer page,
      @Query("per_page") Integer pageSize,
      @Query("s") @SortOrder String sort);
}
```

Converters

```
interface DribbbleSearchService {
  @GET("search")
  Call<List<Shot>> search(
      @Query("q") String query,
      @Query("page") Integer page,
      @Query("per_page") Integer pageSize,
      @Query("s") @SortOrder String sort);
}
```

Converters

```
/**
 * Dribbble API does not have a search endpoint
 * so we have to do gross things :(
 */
class DribbbleSearchConverter {
  // ...
}
```

Converters

```
/**
 * Dribbble API does not have a search endpoint
 * so we have to do gross things :(
 */
class DribbbleSearchConverter {
  // ...
}
```

Converters

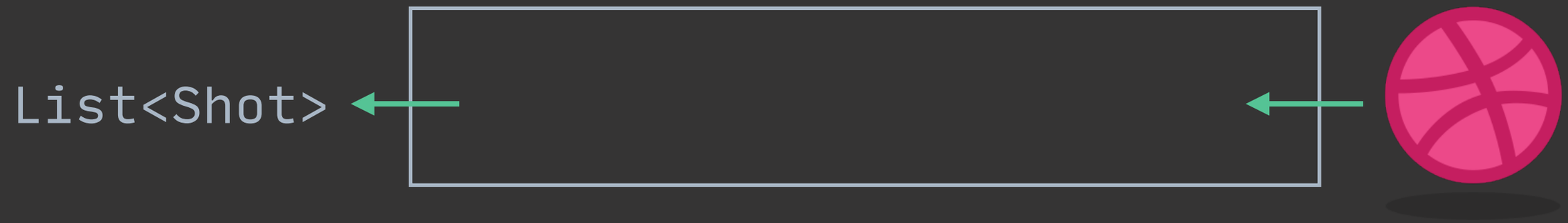
```
/**
 * Dribbble API does not have a search endpoint
 * so we have to do gross things :(
 */
class DribbbleSearchConverter {
  // ...
}
```

http://jakes.link/plaid-converter

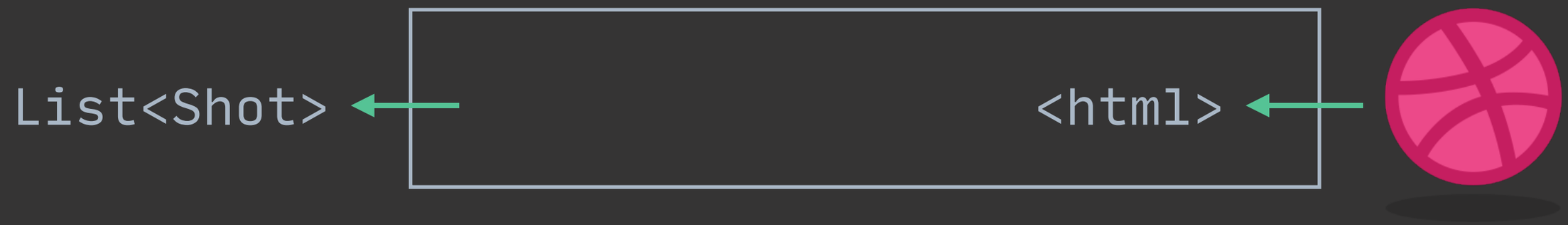
Converters

`List<Shot>`

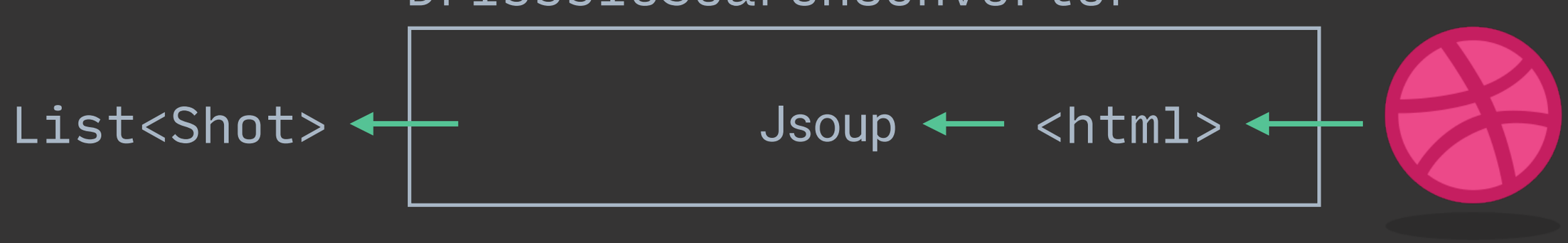
Converters



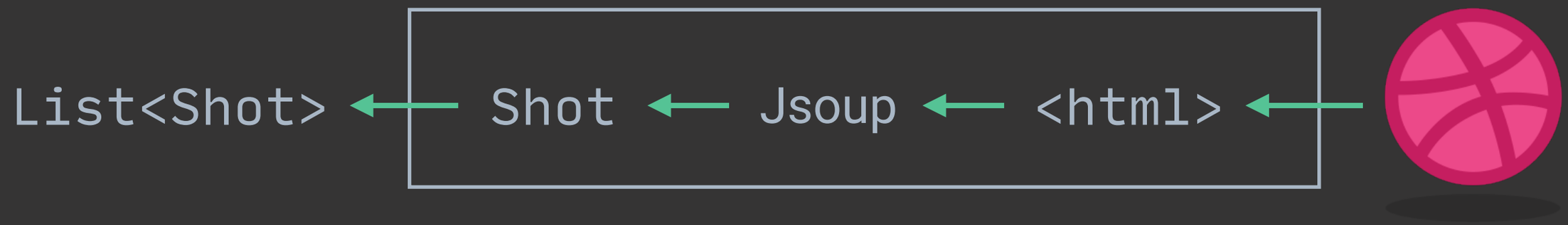
Converters



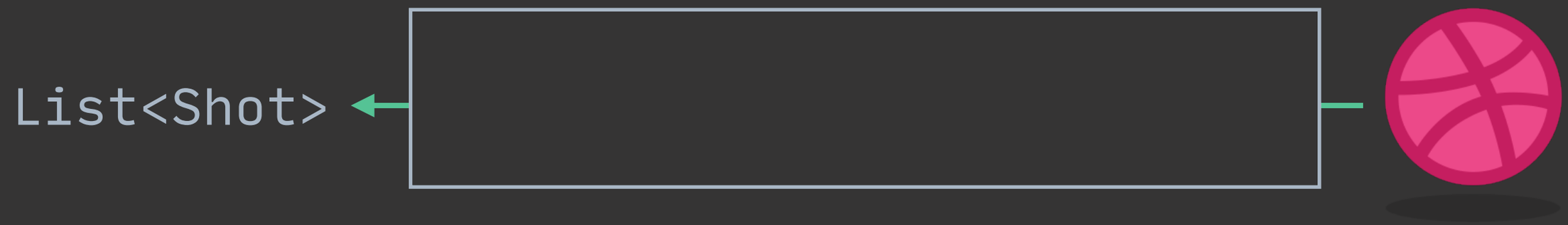
Converters



Converters



Converters



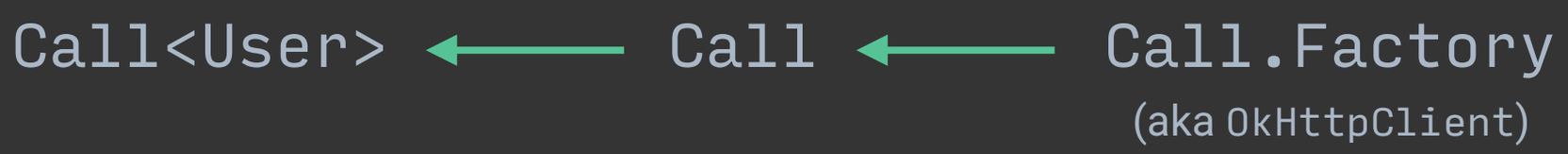
Call Adapters

Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

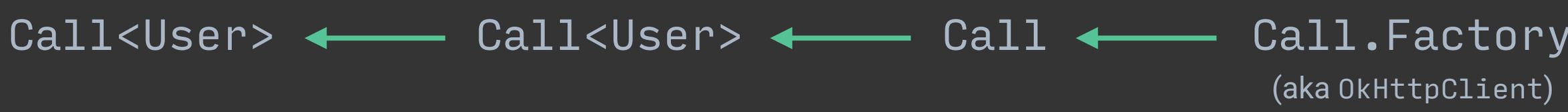
Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```



Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```



Call Adapters

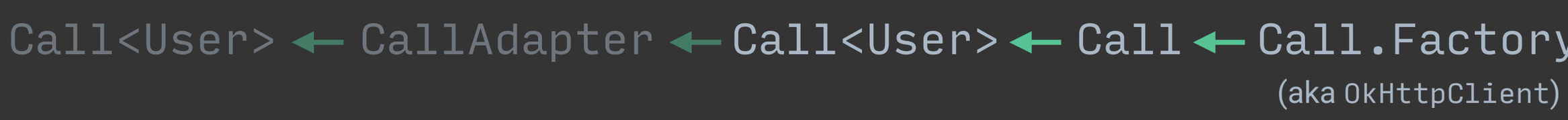
```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

`Call<User>` ← `CallAdapter` ← `Call<User>` ← `Call` ← `Call.Factory`

(aka `OkHttpClient`)

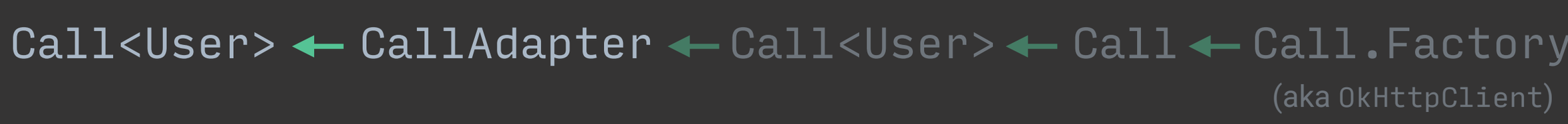
Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```



Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```



Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Observable<User> ← CallAdapter ← Call<User> ← Call ← Call.Factory

(aka OkHttpClient)

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Observable<User> ← CallAdapter ← Call<User> ← Call ← Call.Factory

(aka OkHttpClient)

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Observable<User> ← CallAdapter ← Call<User> ← Call ← Call.Factory

(aka OkHttpClient)

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ?
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ✓
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ?
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ✗
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ✗
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ✗
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict! ?
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())    ✗
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict! ✓
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        new BuiltInCallFactory()) // implict!
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        Java8CallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .addCallAdapterFactory(
        Java8CallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();

  @GET("/friends")
  Call<List<User>> friends();

  @GET("/enemies")
  CompletableFuture<List<User>> enemies();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.create())
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.createWithScheduler(io()))
    .build();
```

```
interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
    return new CallAdapter<Observable<?>>() {
      @Override public Observable<?> adapt(Call<Object> call) {
      }
    }
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
    return new CallAdapter<Observable<?>>() {
      @Override public Observable<?> adapt(Call<Object> call) {
        Observable<?> o = delegate.adapt(call);
      }
    }
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
    return new CallAdapter<Observable<?>>() {
      @Override public Observable<?> adapt(Call<Object> call) {
        Observable<?> o = delegate.adapt(call);
        return o.observeOn(mainThread());
      }
    };
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
    return new CallAdapter<Observable<?>>() {
      @Override public Observable<?> adapt(Call<Object> call) {
        Observable<?> o = delegate.adapt(call);
        return o.observeOn(mainThread());
      }
      @Override public Type responseType() {
        return delegate.responseType();
      }
    };
  }
}
```

Call Adapters

```
class RxObserveOnCallAdapterFactory extends CallAdapter.Factory {
  @Override public CallAdapter<?, ?> get(Type returnType,
      Annotation[] annotations, Retrofit retrofit) {
    if (getRawType(returnType) != Observable.class) {
      return null;
    }
    CallAdapter<Object, Observable<?>> delegate =
        retrofit.nextCallAdapter(this, returnType, annotations);
    return new CallAdapter<Observable<?>>() {
      @Override public Observable<?> adapt(Call<Object> call) {
        Observable<?> o = delegate.adapt(call);
        return o.observeOn(mainThread());
      }
      @Override public Type responseType() {
        return delegate.responseType();
      }
    };
  }
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.createWithScheduler(io()))
    .build();
```

```
interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .addCallAdapterFactory(
        new RxObserveOnCallAdapterFactory())
    .addCallAdapterFactory(
        RxJavaCallAdapterFactory.createWithScheduler(io()))
    .build();

interface Service {
  @GET("/user")
  Observable<User> user();
}
```

Call Adapters

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

interface Service {
  @GET("/user")
  Call<User> user();
}
```

Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Response<User> response = call.execute();
```

Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Response<User> response = call.execute();
if (!response.isSuccessful()) {
  // handle error body?
}
```

Call Adapters

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Response<User> response = call.execute();
if (!response.isSuccessful()) {
  ResponseBody errorBody = response.errorBody();
}
```

Call Adapters

```
interface Call<R, E> {
  Response<R, E> execute();
  void enqueue(Callback<R, E> cb);
}
```

```
interface Callback<R, E> {
  void onResponse(Response<R, E> response);
  void onFailure(IOException e);
}

interface Response<R, E> {
   R body();
   E error();
}
```

Call Adapters

```
interface Call<R, E> {
  Response<R, E> execute();
  void enqueue(Callback<R, E> cb);
}
```

```
interface Callback<R, E> {
  void onResponse(Response<R, E> response);
  void onFailure(IOException e);
}

interface Response<R, E> {
   R body();
   E error();
}
```

Call Adapters

```
interface Call<R, E> {
  Response<R, E> execute();
  void enqueue(Callback<R, E> cb);
}
```

```
interface Callback<R, E> {
  void onResponse(Response<R, E> response);
  void onFailure(IOException e);
}

interface Response<R, E> {
   R body();
   E error();
}
```

Call Adapters

```
interface Call<R, E> {
  Response<R, E> execute();
  void enqueue(Callback<R, E> cb);
}
```

```
interface Callback<R, E> {
  void onSuccess(R body);
  void onError(E errorBody);
  void onFailure(IOException e);
}
```

Call Adapters

```
interface Call<R, E> {
  Response<R, E> execute();
  void enqueue(Callback<R, E> cb);
}
```

```
interface Callback<R, E> {
  void onSuccess(R body);
  void onClientError(E errorBody);
  void onServerError(String message);
  void onUnauthenticated();
  void onFailure(IOException e);
}
```

Mock Mode

Mock Mode

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

Mock Mode

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Service service = retrofit.create(Service.class);
```

Mock Mode

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Service service = retrofit.create(Service.class);

class MockService implements Service {
  @Override public Call<List<User>> user() {
    return Calls.success(Arrays.asList(
        new User("Alice"), new User("Bob")));
  }
}
```

Mock Mode

```
interface Service {
  @GET("/user")
  Call<User> user();
}
```

```
Service service = retrofit.create(Service.class);

class MockService implements Service {
  @Override public Call<List<User>> user() {
    return Calls.success(Arrays.asList(
        new User("Alice"), new User("Bob")));
  }
}

Service fakeService = new MockService();
```

Mock Mode

```
class MockService implements Service {
  @Override public Call<List<User>> user() {
    return Calls.success(Arrays.asList(
        new User("Alice"), new User("Bob")));
  }
}
```

Mock Mode

```
class MockService implements Service {
  final BehaviorDelegate<Service> delegate;

  MockService(BehaviorDelegate<Service> delegate) {
    this.delegate = delegate;
  }

  @Override public Call<List<User>> user() {
    return Calls.success(Arrays.asList(
        new User("Alice"), new User("Bob")));
  }
}
```

Mock Mode

```
class MockService implements Service {
  final BehaviorDelegate<Service> delegate;

  MockService(BehaviorDelegate<Service> delegate) {
    this.delegate = delegate;
  }

  @Override public Call<List<User>> user() {
    List<User> response = Arrays.asList(
        new User("Alice"), new User("Bob"));
    return delegate
        .returningResponse(response).user();
  }
}
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

NetworkBehavior behavior = NetworkBehavior.create();
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

NetworkBehavior behavior = NetworkBehavior.create();
MockRetrofit mock = new MockRetrofit.Builder()
    .networkBehavior(behavior)
    .build();
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

NetworkBehavior behavior = NetworkBehavior.create();
MockRetrofit mock = new MockRetrofit.Builder()
    .networkBehavior(behavior)
    .build();

BehaviorDelegate<Service> delegate =
    mock.create(Service.class);
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

NetworkBehavior behavior = NetworkBehavior.create();
MockRetrofit mock = new MockRetrofit.Builder()
    .networkBehavior(behavior)
    .build();

BehaviorDelegate<Service> delegate =
    mock.create(Service.class);

Service mockService = new MockService(delegate);
```

Mock Mode

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://example.com")
    .build();

NetworkBehavior behavior = NetworkBehavior.create();
MockRetrofit mock = new MockRetrofit.Builder()
    .networkBehavior(behavior)
    .build();

BehaviorDelegate<Service> delegate =
    mock.create(Service.class);

Service mockService = new MockService(delegate);
```

Mock Mode

```
NetworkBehavior behavior = NetworkBehavior.create();
```

Mock Mode

```
NetworkBehavior behavior = NetworkBehavior.create();
behavior.setDelay(2, SECONDS);
```

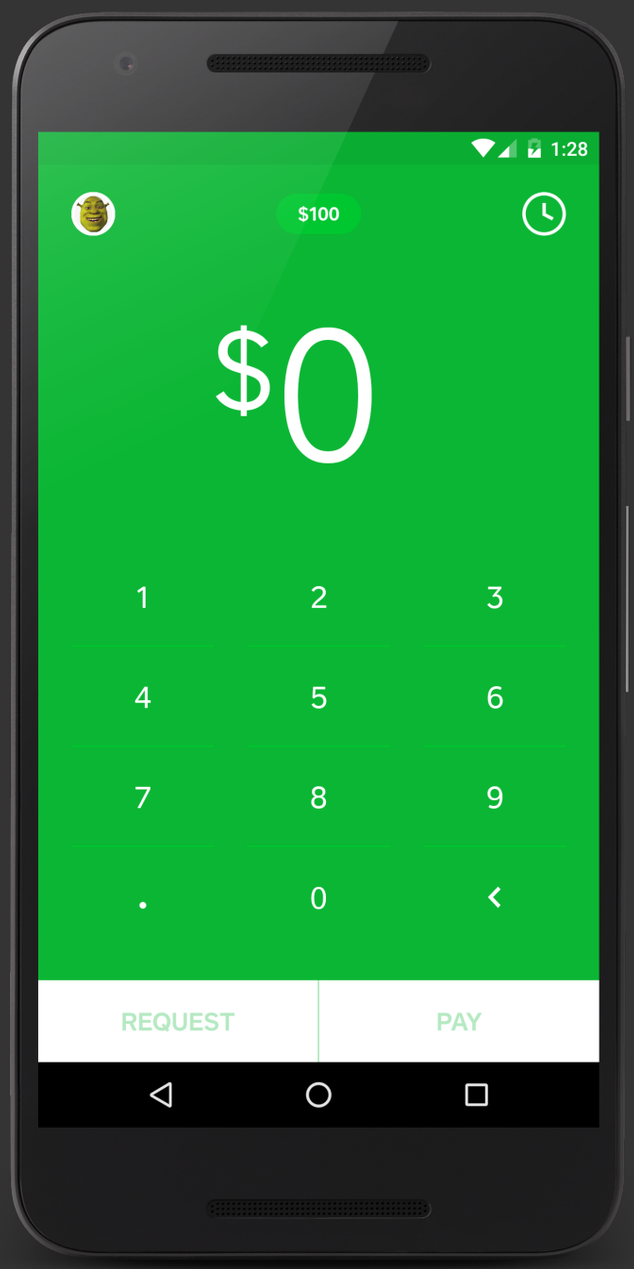
Mock Mode

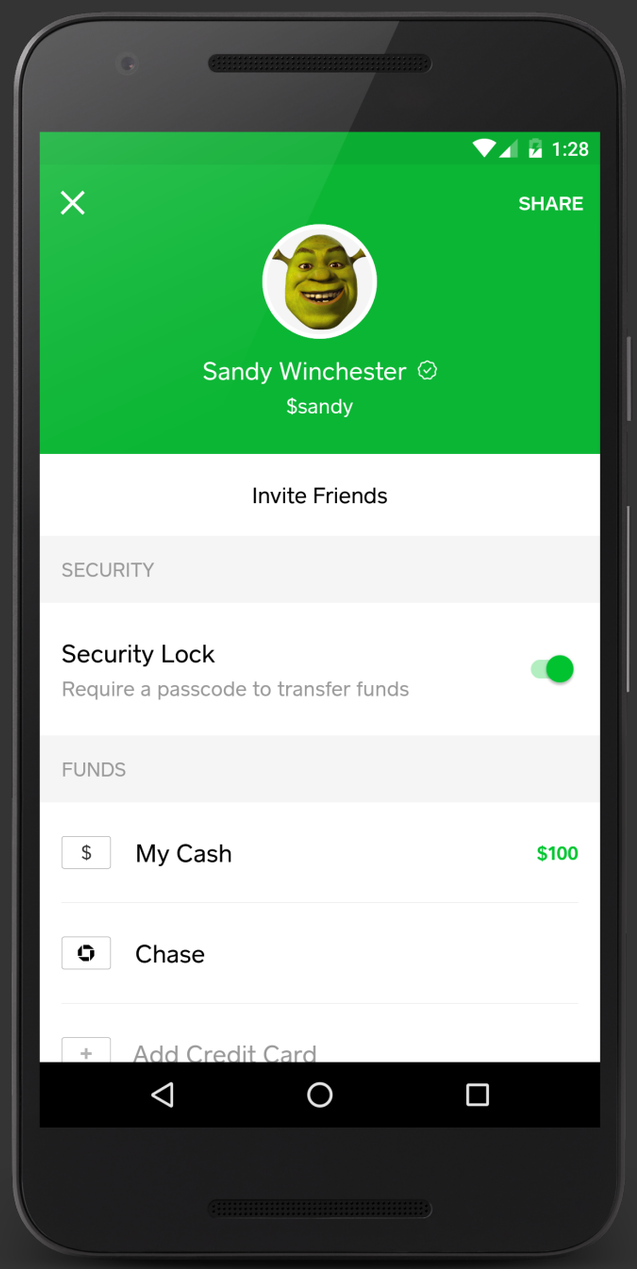
```
NetworkBehavior behavior = NetworkBehavior.create();
behavior.setDelay(2, SECONDS);
behavior.setVariancePercent(40);
```

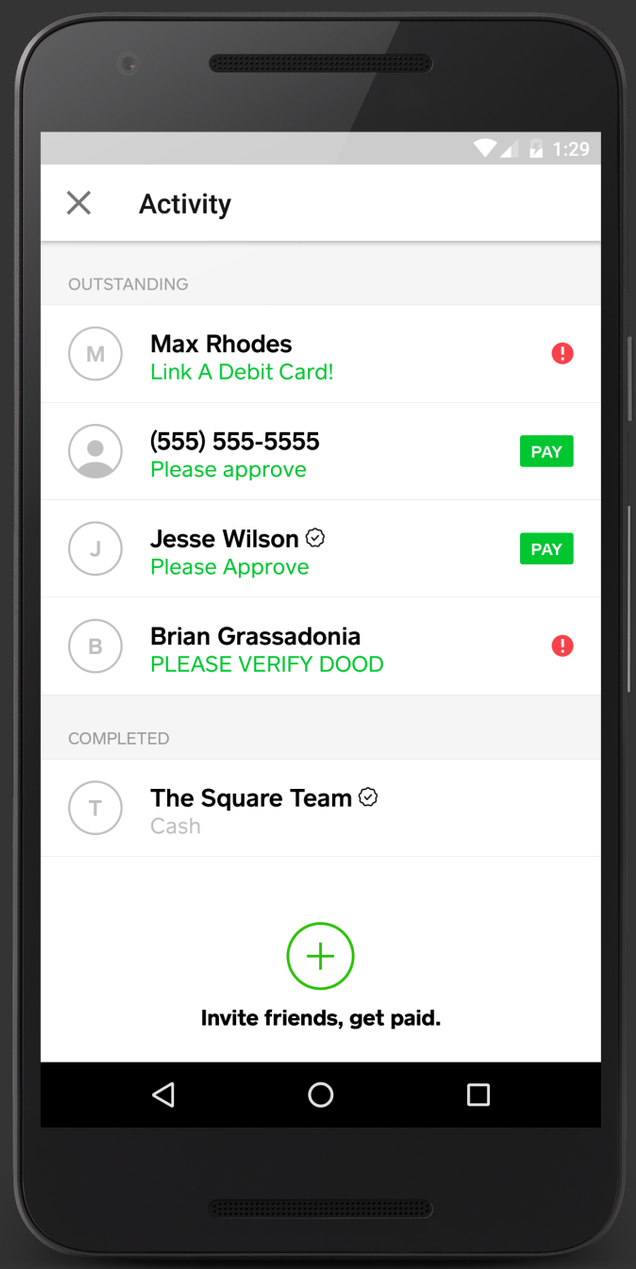
Mock Mode

```
NetworkBehavior behavior = NetworkBehavior.create();
behavior.setDelay(2, SECONDS);
behavior.setVariancePercent(40);
behavior.setFailurePercent(2);
```

Mock Mode


1:28
$100
$0
1
2
3
4
5
6
7
8
9
.
0
REQUEST
PAY


1:28
SHARE
Sandy Winchester
$sandy
Invite Friends
SECURITY
Security Lock
Require a passcode to transfer funds
FUNDS
$
My Cash
$100
Chase
Add Credit Card


1:29
Activity
OUTSTANDING
M
Max Rhodes
Link A Debit Card!
(555) 555-5555
Please approve
PAY
J
Jesse Wilson
Please Approve
PAY
B
Brian Grassadonia
PLEASE VERIFY DOOD
COMPLETED
T
The Square Team
Cash
Invite friends, get paid.

Mock Mode


9:45
Trending today
DNC-tensorflow
Mostafa-Samir — A TensorFlow implementation of DeepMind's Differential Neural Computers (DNC)
112 17
saythanks.io
kennethreitz — Spreading Thankfulness in Open Source.
52 5
hypervision
mafintosh — P2P Television
43 4
curl-tap-sh
awalGarg — tap curl in your editor before it gets to sh it
31 1
flags
sailormoon — Simple, extensible, header-only C++17 argument parser.
28 0
plug_attack
9:45
Trending today
picasso
square — A powerful image downloading and caching library for Android
5279 1513
retrofit
square — Type-safe REST client for Android and Java by Square, Inc.
4583 775
okhttp
square — An HTTP+SPDY client for Android and Java applications.
3663 828
butterknife
JakeWharton — View "injection" library for Android.
3136 626
dagger
square — A fast dependency injector for Android and Java.
3085 574
u2020

Retrofit

Retrofit

- HTTP client sends the bits across the wire.

Retrofit

- HTTP client sends the bits across the wire.
- Converters manipulate request/response data.

Retrofit

- HTTP client sends the bits across the wire.
- Converters manipulate request/response data.
- Call adapters change execution mechanism.

Retrofit

- HTTP client sends the bits across the wire.
- Converters manipulate request/response data.
- Call adapters change execution mechanism.
- Mock mode creates deterministic, fake data for testing.

Making Retrofit Work For You

twitter.com/ jakewharton
google.com/+ jakewharton
jakewharton .com