Mobile Application Development Using multiple Retrofit API calls containing different response body types on same compilation unit

Waterford Institute of Technology

November 19, 2016

John Fitzgerald

Managing multiple Retrofit API calls using inner classes

Retrofit

Three response body types

Consider this API: three different Call response body types:

- Residence
- String
- List<Residence>

```
public interface ResidenceServiceProxy
{
  @POST("/api/residence")
  Call<Residence> createResidence(@Body Residence residence);

  @DELETE("/api/residences/{id}")
  Call<String> deleteResidence(@Path("id") Long id);

  @GET("/api/residences")
  Call<List<Residence>> getResidences();

}
```

Retrofit

Single response body type

```
@POST("/api/residence")
Call<Residence> createResidence(@Body Residence residence);
```

```
public class ResidenceFragment implements Callback<Residence> {

  @Override
  public void onResponse(Response<Residence> response, Retrofit retrofit) {

  }

  @Override
  public void onFailure(Throwable t) {

  }
}
```

Retrofit

Two different response body types

```
// The API calls we wish to make:
@GET("/api/residences")
Call<List<Residence>> getResidences();
@DELETE("/api/residences/{id}")
Call<String> deleteResidence(@Path("id") Long id);
```

```
// This is illegal: allowed only one Callback here
public class ResidenceFragment implements Callback<List<Residence>>,
                                          Callback<String>
{
  ...
}
```

Retrofit

Two different response body types

Pattern using inner classes.

Later we demonstrate anonymous class solution.

```
/**
 * Instantiate inner class (defined in next slide)
 * Use inner class object as enqueue parameter
 */
public void retrieveResidences() {
  RetrieveResidences retrieveResidences = new RetrieveResidences();
  Call<List<Residence>> call = app.residenceService.getResidences();
  call.enqueue(retrieveResidences);
}
```

Retrofit

Two different response body types

This is one particular pattern that overcomes problem:

```
// Create inner class the implements Callback<T>. Repeat for all T
class RetrieveResidences implements Callback<List<Residence>>
{
  @Override
  public void onResponse(Response<List<Residence>> response,
                                          Retrofit retrofit) {. . .}

  @Override
  public void onFailure(Throwable t) {. . .}
}
```

```
// The associated API call
@GET("/api/residences")
Call<List<Residence>> getResidences();
```

Retrofit

Two different response body types

Suppose we wish to delete a server Residence from within same compilation unit (ResidenceFragment):

```
public void deleteResidence(Long id) {
  DeleteRemoteResidence delResidence = new DeleteRemoteResidence();
  Call<String> call = app.residenceService.deleteResidence(id);
  call.enqueue(delResidence);
}
```

Retrofit

Two different response body types

And here is DeleteRemoteResidence inner class:

```
class DeleteRemoteResidence implements Callback<String>
{
  @Override
  public void onResponse(Response<String> response, Retrofit retrofit) {
  . . .
  }

  @Override
  public void onFailure(Throwable t) { . . .}
}
```

```
@DELETE("/api/residences/{id}")
Call<String> deleteResidence(@Path("id") Long id);
```

Managing multiple Retrofit API calls using anonymous classes

Retrofit

Two different response body types

Here is the anonymous class approach to fetching list residences:

```
call.enqueue(new Callback<List<Residence>>() {

  @Override
  public void onResponse(Response<List<Residence>> response,
                                  Retrofit retrofit) {...}

  @Override
  public void onFailure(Throwable t) {...}
});
```

```
  @DELETE("/api/residences/{id}")
  Call<String> deleteResidence(@Path("id") Long id);
```

Retrofit

Two different response body types

Here is the anonymous class approach to deleting a residence:

```
call.enqueue(new Callback<String>() {
  @Override
  public void onResponse(Response<String> response, Retrofit retrofit) {. .
       .}

  @Override
  public void onFailure(Throwable t) {. . .}
});
```

```
  @DELETE("/api/residences/{id}")
  Call<String> deleteResidence(@Path("id") Long id);
```

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit