

# Mobile Application Development MyRent Tester (JUnit)

Waterford Institute of Technology

November 1, 2016

John Fitzgerald

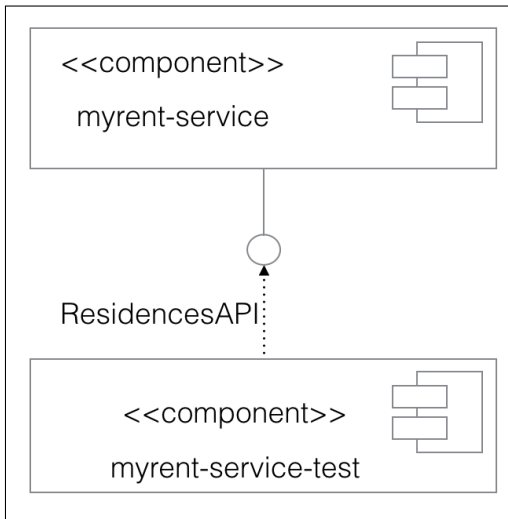
# JUnit Tester

## Learning objectives

- How to create basic JUnit test app?
- How to use Retrofit to enable tester-service communication?

# MyRent service app

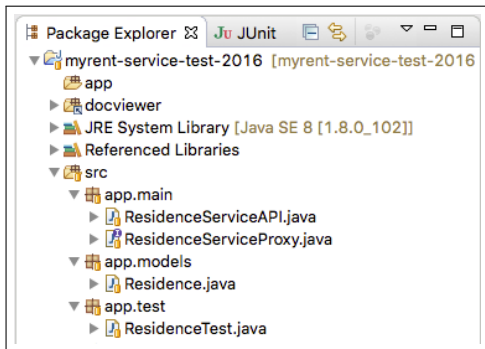
## JUnit test - service



# MyRent test app

Use to test service before client dev

- Create a Play app.
- Customize models and test classes.
- No user interface.
- No JPA relationships.
- Service uses Gson converters:
  - Json to Residence.
  - Residence to Json.



# Model

## Refactor Residence model class

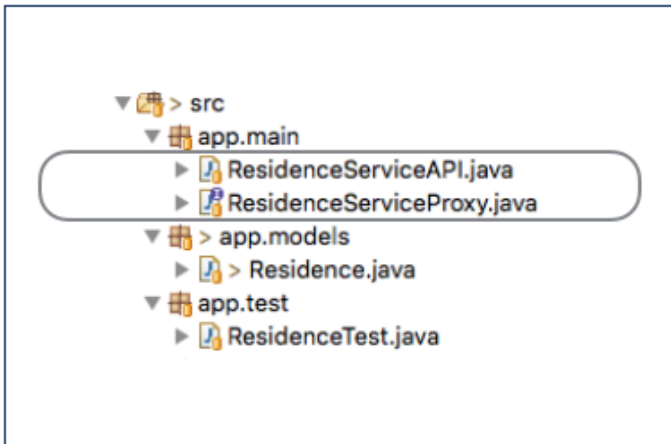
- Override Object.equals

```
@Override
public boolean equals(final Object obj) {
    if (obj instanceof Residence) {
        final Residence other = (Residence) obj;
        return Objects.equal(id, other.id)
            && Objects.equal(geolocation, other.geolocation)
            && Objects.equal(date, other.date)
            . . . ;
    }
    return false;
}
```

# HTTP Client

Retrofit integration requires API & proxy classes

- API & Proxy classes



# HTTP Client

Retrofit integration requires API & proxy classes

- API class

```
public class ResidenceServiceAPI
{
    private String service_url = "http://localhost:9000";
    private ResidenceServiceProxy service;

    public ResidenceServiceAPI() {
        Gson gson = new GsonBuilder().create();

        Retrofit retrofit = new Retrofit.Builder().baseUrl(service_url)
            .addConverterFactory(GsonConverterFactory.create(gson))
            .build();
        service = retrofit.create(ResidenceServiceProxy.class);
    }
}
```

Retrofit instance

Proxy

# HTTP Client

Retrofit integration requires API & proxy classes

- API class

```
public class ResidenceServiceAPI
{
    . . .
    public List<Residence> getResidences() throws Exception
    {
        Call<List<Residence>> call
            = (Call<List<Residence>>) service.getResidences();
        Response<List<Residence>> residences = call.execute();
        return residences.body();
    }
}
```



# HTTP Client

Retrofit integration requires API & proxy classes


- Proxy class

```
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/{id}")
    Call<Residence> getResidence(@Path("id") Long id);

    . . .
}
```



# MyRent service app

## Routes - API

Test client app communicates with service using these patterns:

# Residence		
POST	/api/residence	ResidencesAPI.createResidence
GET	/api/residences	ResidencesAPI.getResidences
DELETE	/api/residences/{id}	ResidencesAPI.deleteResidence
POST	/api/residence/update	ResidencesAPI.updateResidence

# MyRent test code

## Skeleton class

```
public class ResidenceTest {  
  
    @Before  
    public void setup() throws Exception {  
  
    }  
  
    @After  
    public void teardown() throws Exception {  
  
    }  
  
    @Test  
    public void getResidences() throws Exception {  
  
    }  
    ...  
}
```

# MyRent test code

## Sample test method

Retrieve all residence records over network & test

```
/**
 * Obtain entire collection of residences
 * @throws Exception
 */
@Test
public void getResidences() throws Exception {
    List<Residence> residences = service.getResidences();
    assertEquals(residences.size(), NUMBER_residences);
}
```

# References

Play Framework JUnit Test application

1.Play: Testing your application

<https://goo.gl/1gbz1H> [Accessed 2016-10-30]



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology  
INSTITIÚD TEICNEOLAÍOCHTA PHOIRT LÁIRGE

