Mobile Application Development
MyRent Service

Waterford Institute of Technology

November 1, 2016

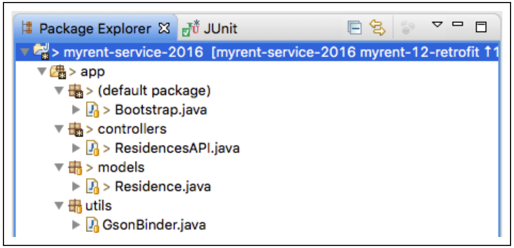
John Fitzgerald

MyRent service
Learning objectives

- Service being provided to facilitate client dev.
- Service provided for deployment to localhost.
- Service also deployed on Heroku (access provided).

MyRent service app

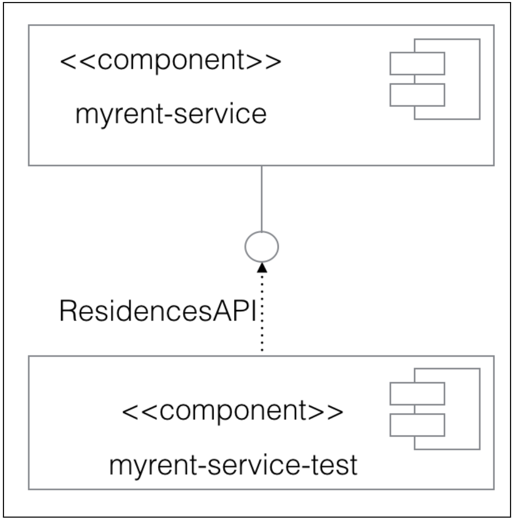
File structure



```
▼ > myrent-service-2016  [myrent-service-2016 myrent-12-retrofit ↑1
  ▼ > app
    ▼ > (default package)
      ▶ > Bootstrap.java
    ▼ > controllers
      ▶ > ResidencesAPI.java
    ▼ > models
      ▶ > Residence.java
    ▼ utils
      ▶ GsonBinder.java
```

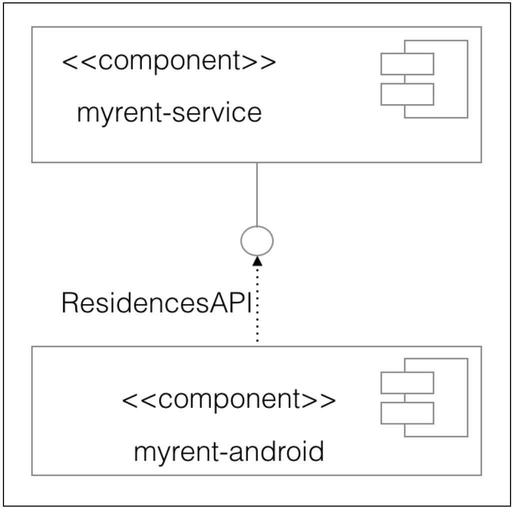
MyRent service app

Service JUnit Play tested



MyRent service app

Android client development



MyRent service app

Model

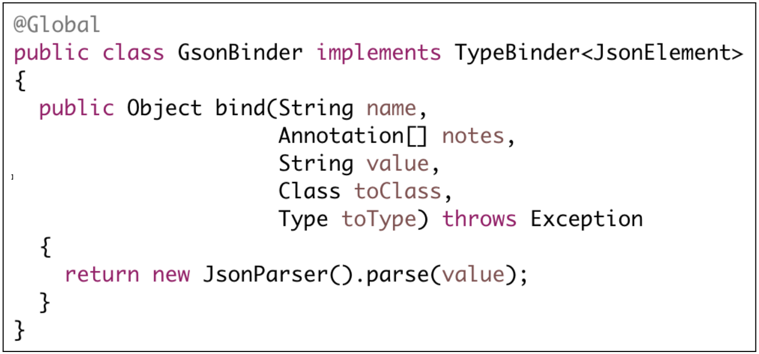
- GenericModel extended.
- Play Model an alternative.
- Generic: allows bespoke id.
- Model: auto-generates id.

```
@Entity
public class Residence extends GenericModel
{

   @Id
   public Long id;
   public String geolocation;
   public Long date;
   public boolean rented;
   public String tenant;
   public double zoom;
   public String photo;
    ..
}
```

MyRent service app

Utility class

Enables controller actions to translate to & from Json objects.

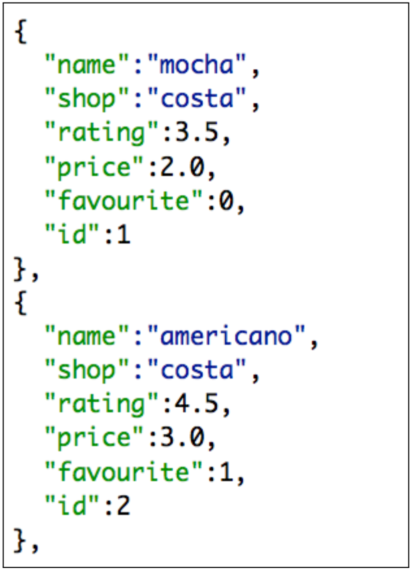


```
@Global
public class GsonBinder implements TypeBinder<JsonElement>
{
  public Object bind(String name,
                     Annotation[] notes,
                     String value,
                     Class toClass,
                     Type toType) throws Exception
  {
    return new JsonParser().parse(value);
  }
}
```

MyRent service app

JavaScript Object Notation (JSON)

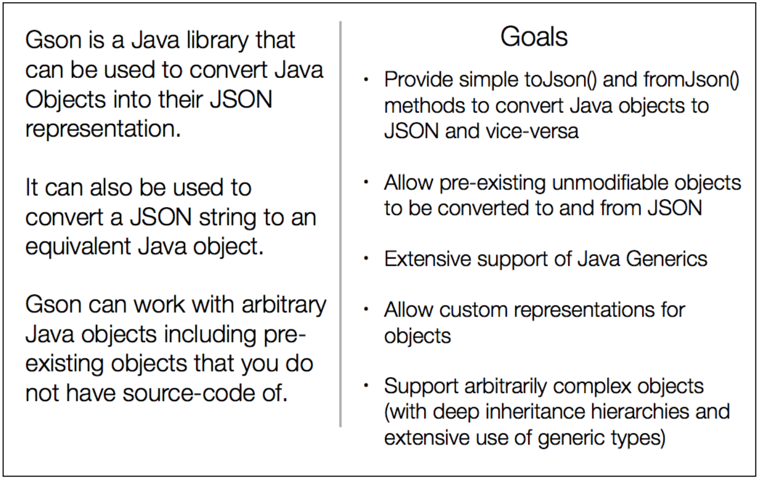
- Text-based open standard.
- Douglas Crockford originator.
- Transmit network data.
- Replacing XML.



```
{
  "name":"mocha",
  "shop":"costa",
  "rating":3.5,
  "price":2.0,
  "favourite":0,
  "id":1
},
{
  "name":"americano",
  "shop":"costa",
  "rating":4.5,
  "price":3.0,
  "favourite":1,
  "id":2
},
```

MyRent service app

Google Gson



Gson is a Java library that can be used to convert Java Objects into their JSON representation.

It can also be used to convert a JSON string to an equivalent Java object.

Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of.

Goals

- Provide simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa
- Allow pre-existing unmodifiable objects to be converted to and from JSON
- Extensive support of Java Generics
- Allow custom representations for objects
- Support arbitrarily complex objects (with deep inheritance hierarchies and extensive use of generic types)

MyRent service app

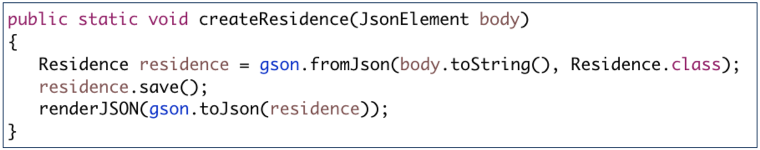
Controller: ResidencesAPI

- Actions do not render views.
- Use renderJson to return data.
- HTTP response codes: OK, notFound.

```
public class ResidencesAPI extends Controller {

  static Gson gson = new GsonBuilder().create();

  public static void createResidence(JsonElement body)  {
    Residence residence = gson.fromJson(body.toString(), Residence.class);
    residence.save();
    renderJSON(gson.toJson(residence));
  }

  public static void updateResidence(JsonElement body)  {
    Residence modifiedResidence = gson.fromJson(body.toString(), Residence.class);
    Residence residence = Residence.findById(modifiedResidence.id);
      . . .
  }

  public static void getResidences()  {
    List<Residence> residences = Residence.findAll();
    renderJSON(gson.toJson(residences));
  }

  public static void deleteResidence(Long id)  {
    Residence residence = Residence.findById(id);
      . . .
  }
}
```

MyRent service app

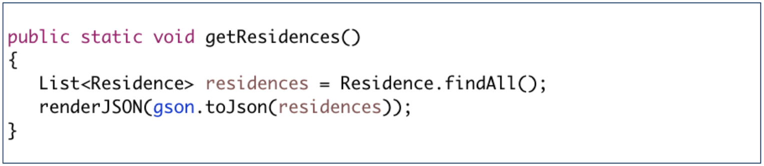
Controller: ResidencesAPI



```java
public static void createResidence(JsonElement body)
{
   Residence residence = gson.fromJson(body.toString(), Residence.class);
   residence.save();
   renderJSON(gson.toJson(residence));
}
```

MyRent service app

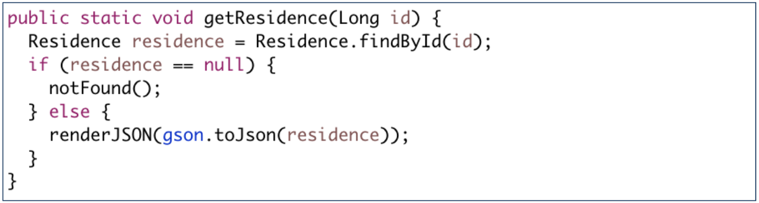
Controller: ResidencesAPI



```
public static void getResidences()
{
   List<Residence> residences = Residence.findAll();
   renderJSON(gson.toJson(residences));
}
```

MyRent service app

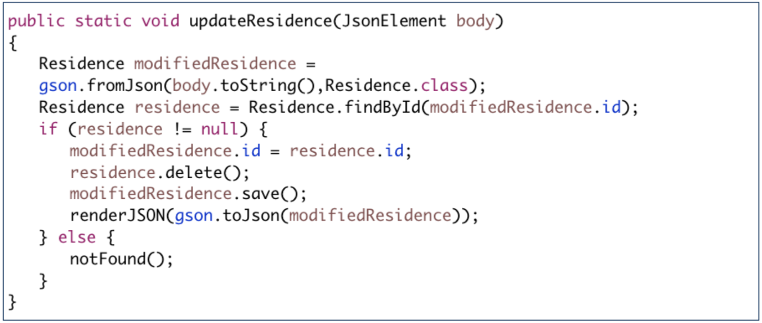
Controller: ResidencesAPI



```java
public static void getResidence(Long id) {
  Residence residence = Residence.findById(id);
  if (residence == null) {
    notFound();
  } else {
    renderJSON(gson.toJson(residence));
  }
}
```

MyRent service app

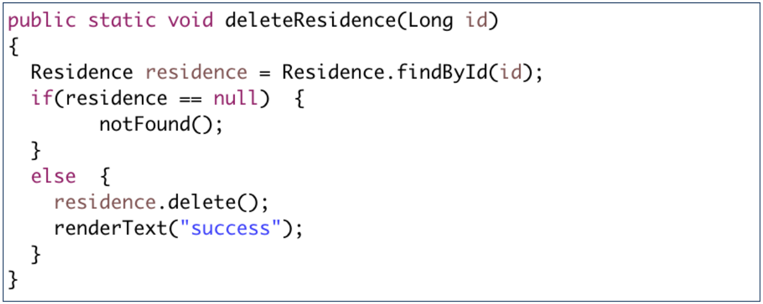
Controller: ResidencesAPI



```java
public static void updateResidence(JsonElement body)
{
   Residence modifiedResidence =
   gson.fromJson(body.toString(),Residence.class);
   Residence residence = Residence.findById(modifiedResidence.id);
   if (residence != null) {
      modifiedResidence.id = residence.id;
      residence.delete();
      modifiedResidence.save();
      renderJSON(gson.toJson(modifiedResidence));
   } else {
      notFound();
   }
}
```

MyRent service app

Controller: ResidencesAPI



```
public static void deleteResidence(Long id)
{
  Residence residence = Residence.findById(id);
  if(residence == null)  {
        notFound();
  }
  else  {
    residence.delete();
    renderText("success");
  }
}
```

MyRent service app

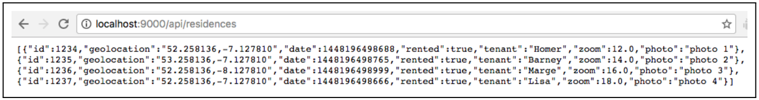
Routes - API

Client application uses these patterns to communicate with service.

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

MyRent service app

Routes - API



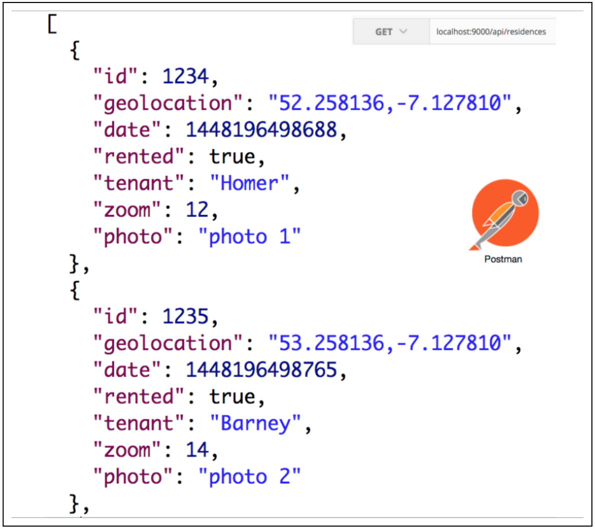
localhost:9000/api/residences

```
[{"id":1234,"geolocation":"52.258136,-7.127810","date":1448196498688,"rented":true,"tenant":"Homer","zoom":12.0,"photo":"photo 1"},
{"id":1235,"geolocation":"53.258136,-7.127810","date":1448196498765,"rented":true,"tenant":"Barney","zoom":14.0,"photo":"photo 2"},
{"id":1236,"geolocation":"52.258136,-8.127810","date":1448196498999,"rented":true,"tenant":"Marge","zoom":16.0,"photo":"photo 3"},
{"id":1237,"geolocation":"52.258136,-7.127810","date":1448196498666,"rented":true,"tenant":"Lisa","zoom":18.0,"photo":"photo 4"}]
```

MyRent service app

Test your API with Postman

GET localhost:9000/api/residences



```
[
  {
    "id": 1234,
    "geolocation": "52.258136,-7.127810",
    "date": 1448196498688,
    "rented": true,
    "tenant": "Homer",
    "zoom": 12,
    "photo": "photo 1"
  },
  {
    "id": 1235,
    "geolocation": "53.258136,-7.127810",
    "date": 1448196498765,
    "rented": true,
    "tenant": "Barney",
    "zoom": 14,
    "photo": "photo 2"
  },
```

MyRent service app

Preload sample data

Not compatible with unit testing.

```
// Bootstrap.java
@OnApplicationStart
public class Bootstrap extends Job {
  public void doJob() {
    if (Residence.count() == 0) {
     Fixtures.deleteDatabase();
     Fixtures.loadModels("data.yml");
    }
  }
}
```

MyRent service app

Preload sample data

Not compatible with unit testing.

```
// data.yml
Residence(residence_1):
  id: 1234
  geolocation: "52.258136,−7.127810"
  date: 1448196498688
  rented: true
  tenant: Homer
  zoom: 12.0
  photo: "photo 1"
```

References

Retrofit from Square Open Source

1.Gson User Guide

https://goo.gl/ZPXTsI [Accessed 2016-10-30]

3. Postman (Chrome Web Store)

https://goo.gl/1we0lx [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
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit