Mobile Application Development

Higher Diploma in Science in Computer Science

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Donation-Android V6

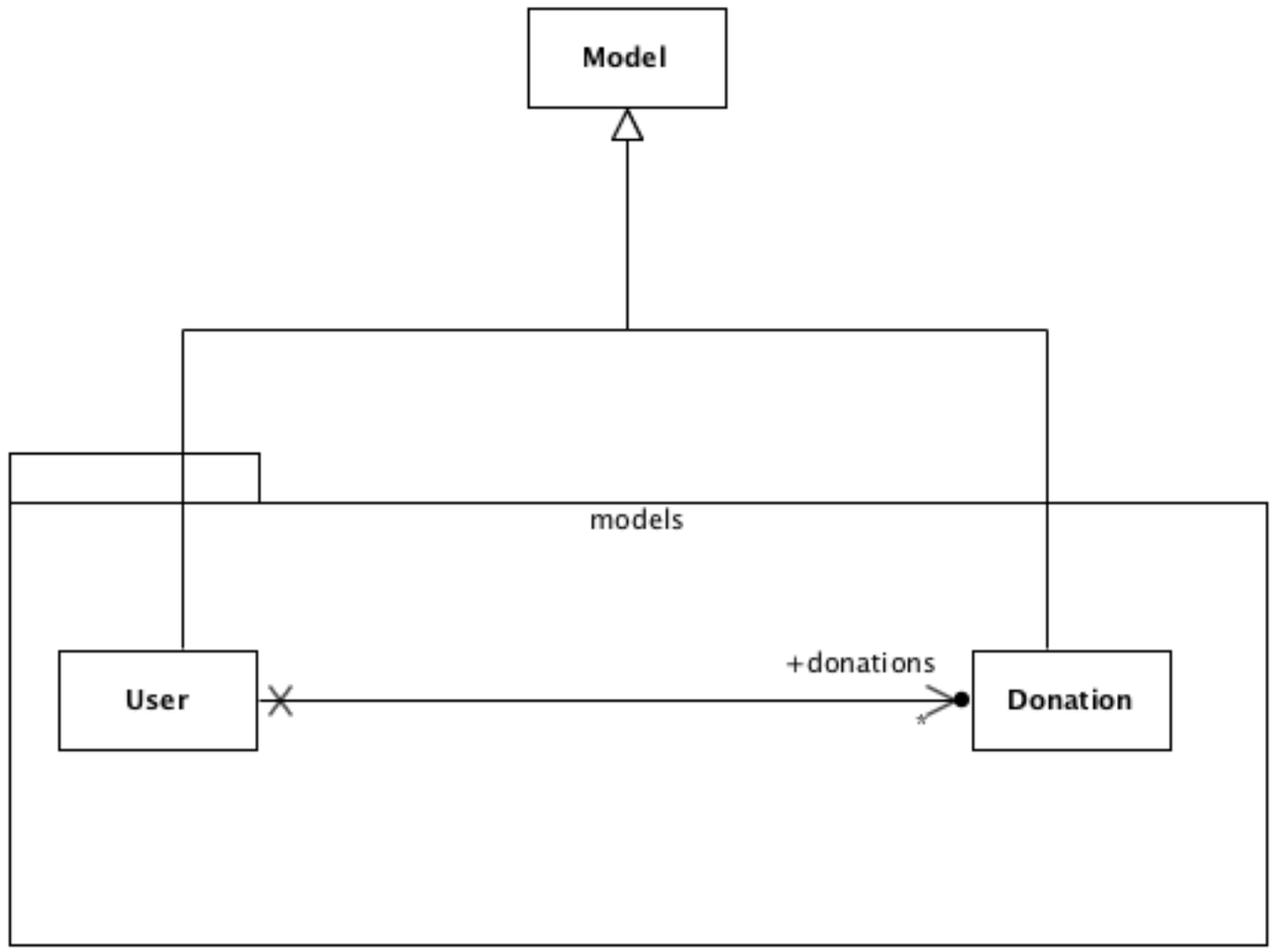
Original API

```
GET     /api/users                          DonationServiceAPI.users
GET     /api/users/{id}                     DonationServiceAPI.user
POST    /api/users                          DonationServiceAPI.createUser
DELETE  /api/users/{id}                     DonationServiceAPI.deleteUser

GET     /api/donations                      DonationServiceAPI.donations
GET     /api/donations/{id}                 DonationServiceAPI.donation
POST    /api/donations                      DonationServiceAPI.createDonation
DELETE  /api/donations/{id}                 DonationServiceAPI.deleteDonation
```

- V1 Routes
- No relationship between User and Donation
- Each Object accessed independently

Revised Model



User -> Donation

```java
@Entity
public class User extends Model
{
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  @OneToMany(cascade = CascadeType.ALL)
  public List<Donation> donations = new ArrayList<Donation>();

  public User()
  {}

  public User(String firstName, String lastName,
              String email,     String password)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
  }

  public static User findByEmail(String email)
  {
    return find("email", email).first();
  }

  public boolean checkPassword(String password)
  {
    return this.password.equals(password);
  }
}
```

```java
@Entity
public class Donation extends Model
{
  public int    amount;
  public String method;

  public Donation()
  {}

  public Donation (int amount, String method)
  {
    this.amount = amount;
    this.method = method;
  }

  public String toString()
  {
    return amount + ", " + method;
  }
}
```

Revised API

```
# API - Users

GET     /api/users                          UsersAPI.users
GET     /api/users/{id}                     UsersAPI.user
POST    /api/users                          UsersAPI.createUser
DELETE  /api/users/{id}                     UsersAPI.deleteUser

# API - Donations

GET     /api/users/{userId}/donations       DonationsAPI.donations
GET     /api/users/{userId}/donations/{id}  DonationsAPI.donation
POST    /api/users/{userId}/donations       DonationsAPI.createDonation
DELETE  /api/users/{userId}/donations/{id}  DonationsAPI.deleteDonation
```

- Users API Unchanged
- Donations - revised to incorporate User ID directly into URL

API Examples (1)

- **GET /users/23**
  - Get a user with ID 23
- **GET /users/23/donations**
  - Get all donations made by user with ID 23
- **GET /users/23/donation/2**
  - Get the donation user 23 made, with donation ID 2

API Examples (2)

- **POST /users**
  - Create a new user, return new user (with ID)
- **POST /users/23/donations**
  - Create a new donation, return new donation (with ID)

Users API

```
# API - Users

GET     /api/users          UsersAPI.users
GET     /api/users/{id}     UsersAPI.user
POST    /api/users          UsersAPI.createUser
DELETE  /api/users/{id}     UsersAPI.deleteUser
```

```
public class UsersAPI extends Controller
{
  public static void users()
  {
    List<User> users = User.findAll();
    renderJSON(JsonParsers.user2Json(users));
  }

  public static void user(Long id)
  {
    User user = User.findById(id);
    if (user == null)
    {
      notFound();
    }
    else
    {
      renderJSON(JsonParsers.user2Json(user));
    }
  }
  public static void createUser(JsonElement body)
  {
    User user = JsonParsers.json2User(body.toString());
    user.save();
    renderJSON(JsonParsers.user2Json(user));
  }
  public static void deleteUser(Long id)
  {
    User user = User.findById(id);
    if (user == null)
    {
      notFound();
    }
    else
    {
      user.delete();
      renderText("success");
    }
  }
  public static void deleteAllUsers()
  {
    User.deleteAll();
    renderText("success");
  }
}
```

Donations API

```
# API - Donations

GET     /api/users/{userId}/donations          DonationsAPI.donations
GET     /api/users/{userId}/donations/{id}     DonationsAPI.donation
POST    /api/users/{userId}/donations          DonationsAPI.createDonation
DELETE  /api/users/{userId}/donations/{id}     DonationsAPI.deleteDonation
```

```
public class DonationsAPI extends Controller
{
  public static void donations(Long userId)
  {
    User user = User.findById(userId);
    List<Donation> donations = user.donations;
    renderText(JsonParsers.donation2Json(donations));
  }

  public static void donation (Long userId, Long id)
  {
                                                    );
                                                    dById(id);
                                                    tion))

                                                    on2Json(donation));
  }

  public static void createDonation(Long userId, JsonElement body)
  {
    User user = User.findById(userId);
    Donation donation = JsonParsers.json2Donation(body.toString());
    Donation newDonation = new Donation (donation.amount, donation.method);
    user.donations.add(donation);
    user.save();
    renderJSON (JsonParsers.donation2Json(newDonation));
  }

  public static void deleteDonation(Long userId, Long id)
  {
    User user = User.findById(userId);
    Donation donation = Donation.findById(id);
    if (!user.donations.contains(donation))
    {
      notFound();
    }
    else
    {
      user.donations.remove(donation);
      donation.delete();
      user.save();
      ok();
    }
  }
}
```

JsonParsers

- Change library used to parse json from gson to ‘flexjson’
- Enables fields to be included/excluded easily
- We specifically exclude class metadata, model derived elements + collections
- This will simplify the data structures in the client apps, and prevent requests generating excessive responses

```
public class JsonParsers
{
  public static JSONSerializer userSerializer     = new JSONSerializer().exclude("class")
                                                                        .exclude("persistent")
                                                                        .exclude("entityId")
                                                                        .include("donations");
  public static JSONSerializer donationSerializer = new JSONSerializer().exclude("class")
                                                                        .exclude("persistent")
                                                                        .exclude("entityId");
  public static User json2User(String json)
  {
    return new JSONDeserializer<User>().deserialize(json, User.class);
  }

  public static String user2Json(Object obj)
  {
    return userSerializer.serialize(obj);
  }
```

Donation-Android V6

Donation-Andord Model

- Keep the model simple by excluding OneToMany relationships
- This will reduce temptation to ‘mirror’ the service side data structure in the client
- Client focus is on retrieving relevant information, not complete object graph

Revisions - User / Donation Classes

```java
public class User
{
  public Long   id;
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  public User()
  {}

  public User(String firstName, String lastName,
              String email,     String password)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
  }
}
```

```java
public class Donation
{
  public Long   id;
  public int    amount;
  public String method;

  public User  from;

  Donation()
  {}

  public Donation (int amount, String method)
  {
    this.amount = amount;
    this.method = method;
  }

  public String toString()
  {
    return amount + ", " + method;
  }
}
```

- Includes IDs, which are generated and returned by service

Revisions - DonationApp

- Maintain ‘current’ logged in user in DonationApp

```
public class DonationApp extends Application
{
  //…

  public User            currentUser;

  //…

  public boolean validUser (String email, String password)
  {
    for (User user : users)
    {
      if (user.email.equals(email) && user.password.equals(password))
      {
        currentUser = user;
        return true;
      }
    }
    return false;
  }
```

Revised Request - GetDonations

```
class GetDonations extends Request
{
  private User user;

  public GetDonations(Context context, User user, Response<Donation> callback, String message)
  {
    super(context, callback, message);
    this.user = user;
  }

  @Override
  protected List<Donation> doRequest(Object... params) throws Exception
  {
    String response = Rest.get("/api/users/" + user.id + "/donations");
    List<Donation> donationList = JsonParsers.json2Donations(response);
    return donationList;
  }
}
```

- Command now takes a user object
- This is used to provide the ID of the user in the API
- eg : GET /users/23/donations

Revised Request - CreationDonation

```
class CreateDonation extends Request
{
  private User user;

  public CreateDonation(Context context, User user, Response<Donation> callback, String message)
  {
    super(context, callback, message);
    this.user = user;
  }

  @Override
  protected Donation doRequest(Object... params) throws Exception
  {
    String response = Rest.post ("/api/users/" + user.id + "/donations", JsonParsers.donation2Json(params[0]));
    return JsonParsers.json2Donation(response);
  }
}
```

- Command now also takes a user object
- This is used to provide the ID of the user in the API
- eg : POST /users/23/donations

Create Donation API Call

```
public void donateButtonPressed (View view)
{
  String method = paymentMethod.getCheckedRadioButtonId() == R.id.PayPal ? "PayPal" : "Direct";
  int donatedAmount =  amountPicker.getValue();
  if (donatedAmount == 0)
  {
    String text = amountText.getText().toString();
    if (!text.equals(""))
      donatedAmount = Integer.parseInt(text);
  }
  if (donatedAmount > 0)
  {
    DonationServiceAPI.createDonation(this, app.currentUser, this,
                                      "Registering new donation...", new Donation(donatedAmount, method));
  }
}
```

- Pass the current user to the donation API