SharkNet

September 9, 2015

Thilo Stegemann

1 GooglePlusProfile

A Google plus profile(g+ profile) is like a specified profile or an application based on the profile implementation. What a profile is and how it is used, is well explained in a previous chapter "Profile". The content which could be stored in this profile is very similar to a real g+ profile. One of the main reasons for this application is to make a/an wrapper/interface for g+ profile information. If there is a stream from Google which provides data it is possible to easily make an g+ profile object and fill it with data from the stream. At the end our object looks like the raw data of a g+ profile from Google.

1.1 GooglePlusProfileFactory

These g+ profile factory is a manager of g+ profiles. It provides functions to create g+ profiles, to get g+ profiles and to remove them. E.g. `GooglePlusProfile createGooglePlusProfile` (String name, String profileURL) throws SharkKBException; is a function to create a g+ profile. Parameter name is just the name of the profile and parameter profileURL is the location or the link where the profile is located or where it can be found. The implementation of this function is based on functionalities of "ProfileFactory". To create a GooglePlusProfileFactory a ProfileFactory is needed and to create a ProfileFactory a SharkKB is needed. At the end all information or all g+ profiles are stored in the SharkKB. `GooglePlusProfile`

`createGooglePlusProfile` (String name, String profileURL) throws SharkKBException; creates peers with a name and URL as semantic identifiers. These peers are used to be parameters of a createProfile(PeerSementicTag) function of the ProfileFactory. All functions of GooglePlusProfileFactory are like "createGooglePlusProfile(...)" because they use different functionalities of "ProfileFactory". Thats why these application is based on the profile/profileFactory implementation. The GooglePlusProfileFactory is based on ProfileFactory and the GooglePlusProfile is based on Profile.

1.1.1 Create, get and remove a GooglePlusProfile

```
public class AliceGooglePlusProfile {
    //Creating an empty in memory SharkKB
    private SharkKB kb = new InMemoSharkKB();

    //Creating a ProfileFactory object
    private ProfileFactory profileFactory;

    //Creating a GooglePlusProfileFactory object
    private GooglePlusProfileFactory googlePlusProfileFactory;

    //Initialize ProfileFactory with SharkKB and
    //GooglePlusProfileFactory with ProfileFactory
    public AliceGooglePlusProfile() throws SharkKBException {
        profileFactory = new ProfileFactoryImpl(kb);
        googlePlusProfileFactory =
            new GooglePlusProfileFactoryImpl(profileFactory);
    }

    public void createAliceGooglePlusProfile()
                throws SharkKBException {
        googlePlusProfileFactory.createGooglePlusProfile
        ("Alice", "http://www.sharksystem.net/alice.html");
    }

    public GooglePlusProfile getAliceGooglePlusProfile()
                throws SharkKBException {
        return googlePlusProfileFactory.getGooglePlusProfile
```

```
        ("Alice", "http://www.sharksystem.net/alice.html");
    }

    public void removeAliceGooglePlusProfile()
                throws SharkKBException {
        googlePlusProfileFactory.removeGooglePlusProfile
        ("Alice", "http://www.sharksystem.net/alice.html");
}
```

How does the implementation of AliceGooglePlusProfile work? First defining three private members: an empty SharkKB, a not initialized ProfileFactory and a not initialized GooglePlusProfileFactory. Building a constructor that initializes profileFactory with the empty SharkKB and then initializes googlePlusProfileFactory with profileFactory it also throws a SharkKBException. Declaring three functions to create, get and remove Alice g+ profile.

- createAliceGooglePlusProfile(): googlePlusProfileFactory.removeGooglePlusProfile (”Alice”, ”http://www.sharksystem.net/alice.html”);
- getAliceGooglePlusProfile(): googlePlusProfileFactory.getGooglePlusProfile (”Alice”, ”http://www.sharksystem.net/alice.html”);
- removeAliceGooglePlusProfile(): googlePlusProfileFactory.removeGooglePlusProfile (”Alice”, ”http://www.sharksystem.net/alice.html”);

It is important to know the signature of a g+ profile. Signature is first the name of the profile e.g. ”Alice” and second the profile URL e.g. ”http://www.sharksystem.net/alice.html”. These profile URL is like a semantic identifier of a PeerSemanticTag. It shows the link or place where the profile is located. Knowing the signature is important because it is needed to create a profile, to find or get it and to remove it.

1.2 Using and testing Alice GooglePlusProfile

```
public class AliceGooglePlusProfileTest {

    private SharkKB kb = new InMemoSharkKB();
    private ProfileFactory profileFactory;
    private GooglePlusProfileFactory googlePlusProfileFactory;
```

```
public AliceGooglePlusProfileTest() throws SharkKBException {
   profileFactory = new ProfileFactoryImpl(kb);
   googlePlusProfileFactory =
       new GooglePlusProfileFactoryImpl(profileFactory);
}

@Test
public void testCompleteAliceProfile() throws SharkKBException{
    //Create and complete Alice g+ Profile
    GooglePlusProfile myGooglePlusProfile =
        googlePlusProfileFactory.createGooglePlusProfile
        ("Alice", "http://www.sharksystem.net/alice.html");
    myGooglePlusProfile.setFirstName("Allllice");
    myGooglePlusProfile.setLastName("Alpha");
    myGooglePlusProfile.setNickname("Ali");
    myGooglePlusProfile.setFirstName("Alice");
    myGooglePlusProfile.setTagline
        ("Hi Im Alice I like to solve complex
        problems and writing good code.");
    myGooglePlusProfile.setIntroduction
        ("My story begins a long time ago,
        I was 14 when I created my first little program." +
        "At this moment I was so interested in programing
         it even becomes one of my greatest hobbies.");
    myGooglePlusProfile.setBraggingRights
        ("I wrote my own linux kernel.");
    myGooglePlusProfile.setOccupation
        ("I do my bachelors degree of applied science
        at the HTW subject: applied computing");
    myGooglePlusProfile.setSkills
        ("Java, Scala, C++, C, PHP, HTML, CSS");
    myGooglePlusProfile.addEmployment
        ("HTW", "Programmer", 2012, 2015,
         true, "I write Scala and Java tests.");
    myGooglePlusProfile.addEducation("University of Applied Sciences",
        "Applied Computing", 2012, 2015, true,
        "I learned a lot about programming and software engineering.");
```

```
myGooglePlusProfile.addPlace("Berlin", true);
myGooglePlusProfile.addPlace("London", false);
myGooglePlusProfile.setGender("Female");
myGooglePlusProfile.setLookingforFriends(true);
myGooglePlusProfile.setLookingforDating(true);
myGooglePlusProfile.setLookingForNetworking(true);
myGooglePlusProfile.setLookingForRelationship(false);
myGooglePlusProfile.setBirthday("03.12.1992");
myGooglePlusProfile.setRelationship("Im in a relation with Bob.");
myGooglePlusProfile.addOtherName("Jizy");
myGooglePlusProfile.addHomeContact("Mobile", "0151-2357823");
myGooglePlusProfile.addWorkContact("Phone", "030-28665432");
myGooglePlusProfile.addWorkContact("Address", "Rathenau Str. 42");
myGooglePlusProfile.addWorkContact("Fax", "030-28665433");
myGooglePlusProfile.addOtherProfiles
    ("BobsProfile", "https://google/googlePlus/Profiles/Bob");
myGooglePlusProfile.addContributorTo
    ("SharkNet", "http://www.sharksystem.net/apps.html");
myGooglePlusProfile.addLinks
    ("My favorite website", "http://google.com");

//Reading the Profile
assertEquals("Alice", myGooglePlusProfile.getFirstName());
assertEquals("Alpha", myGooglePlusProfile.getLastName());
assertEquals("Ali", myGooglePlusProfile.getNickname());
assertEquals("Hi Im Alice I like to solve complex problems and
        writing good code.", myGooglePlusProfile.getTagline());
assertEquals("My story begins a long time ago,
        I was 14 when I created my first little program." +
        "At this moment I was so interested in programing
        it even becomes one of my greatest hobbies.",
        myGooglePlusProfile.getIntroduction());
assertEquals("I wrote my own linux kernel.",
        myGooglePlusProfile.getBraggingRights());
assertEquals("I do my bachelors degree of applied science
        at the HTW subject: applied computing",
        myGooglePlusProfile.getOccupation());
assertEquals("Java, Scala, C++, C, PHP, HTML, CSS",
```

```
        myGooglePlusProfile.getSkills());
assertEquals("HTW", myGooglePlusProfile.getEmployerName("0"));
assertEquals("Programmer", myGooglePlusProfile.getJobTitle("0"));
assertEquals(2012, myGooglePlusProfile.getStartOfEmployment("0"));
assertEquals(2015, myGooglePlusProfile.getEndOfEmployment("0"));
assertEquals(true, myGooglePlusProfile.getIsEmploymentCurrent("0"));
assertEquals("I write Scala and Java tests.",
        myGooglePlusProfile.getJobDescription("0"));
assertEquals("University of Applied Sciences",
        myGooglePlusProfile.getSchoolName("0"));
assertEquals("Applied Computing", myGooglePlusProfile.getMajor("0"));
assertEquals(2012, myGooglePlusProfile.getStartOfEducation("0"));
assertEquals(2015, myGooglePlusProfile.getEndOfEducation("0"));
assertEquals(true, myGooglePlusProfile.getIsEducationCurrent("0"));
assertEquals("I learned a lot about programming and
        software engineering.",
        myGooglePlusProfile.getCourseDescription("0"));
assertEquals("Berlin", myGooglePlusProfile.getCity("0"));
assertEquals(true, myGooglePlusProfile.getIsPlaceCurrent("0"));
assertEquals("London", myGooglePlusProfile.getCity("1"));
assertEquals(false, myGooglePlusProfile.getIsPlaceCurrent("1"));
assertEquals("Female", myGooglePlusProfile.getGender());
assertEquals(true, myGooglePlusProfile.getLookingForFriends());
assertEquals(true, myGooglePlusProfile.getLookingForDating());
assertEquals(true, myGooglePlusProfile.getLookingForNetworking());
assertEquals(false, myGooglePlusProfile.getLookingForRelationship());
assertEquals("03.12.1992", myGooglePlusProfile.getBirthday());
assertEquals("Im in a relation with Bob.",
        myGooglePlusProfile.getRelationship());
assertEquals("Jizy", myGooglePlusProfile.getOtherName("0"));
assertTrue(myGooglePlusProfile.getHomeContactType(0).equals("Mobile")
        && myGooglePlusProfile.getHomeContactInfo(0)
        .equals("0151-2357823"));
assertTrue(myGooglePlusProfile.getWorkContactType(0).equals("Phone")
        && myGooglePlusProfile.getWorkContactInfo(0)
        .equals("030-28665432"));
assertTrue(myGooglePlusProfile.getWorkContactType(1).equals("Address")
        && myGooglePlusProfile.getWorkContactInfo(1)
```

```
                .equals("Rathenau Str. 42"));
        assertTrue(myGooglePlusProfile.getWorkContactType(2).equals("Fax")
                && myGooglePlusProfile.getWorkContactInfo(2)
                .equals("030-28665433"));
        assertTrue(myGooglePlusProfile.getOtherProfilesLabel(0)
                .equals("BobsProfile")
                && myGooglePlusProfile.getOtherProfilesUrl(0)
                .equals("https://google/googlePlus/Profiles/Bob"));
        assertTrue(myGooglePlusProfile.getContributorsLabel(0)
                .equals("SharkNet")
                && myGooglePlusProfile.getContributorsUrl(0)
                .equals("http://www.sharksystem.net/apps.html"));
        assertTrue(myGooglePlusProfile.getLinkLabel(0)
                .equals("My favorite website")
                && myGooglePlusProfile.getLinkUrl(0)
                .equals("http://google.com"));

    }


}
```

AliceGooglePlusProfileTest is a test class which uses every getter and setter function a GooglePlusProfile provides. In this special example every content option is set once or more often. There is a difference between setters and adding functions. A function that "sets" content can only save one content but not multiple contents at the same time. Functions with "add" like `addEmployment(...)` can add multiple contents at the same time. If setter functions are used twice they simply will overwrite the last content. But content could be arbitrary complex. Just store a large string or a text as string and set it as content. Every stored parameter or variable has its own getter. Some getters need an identifying number as Integer and other getters need an identifying number as String. These are the getters from adding functions, because "adding" functions can add multiple contents it is needed to identify these contents with an index or an identifier. To access e.g. the job title of an employment use function `getJobTitle("0")` the parameter specifies which employment is needed. To add an whole employment as e.g. HTW programmer just use addEmployment("HTW", "Programmer", 2012, 2015,

true, "I write Scala and Java tests."). Meaning of parameters (employer name, job title, start of employment, end of employment, is employment current, description of the job). To know the meaning of all parameters of every function in GooglePlusProfile just visit the GooglePlusProfile interface and see the documentation. Extracting all information out of the first employment stored:

1. myGooglePlusProfile.getEmployerName("0");
2. myGooglePlusProfile.getJobTitle("0");
3. myGooglePlusProfile.getStartOfEmployment("0");
4. myGooglePlusProfile.getEndOfEmployment("0");
5. myGooglePlusProfile.getIsEmploymentCurrent("0");
6. myGooglePlusProfile.getJobDescription("0");

The index of "adding" functions is calculated. A removing function for "adding" function is needed, because overwriting content is not possible. Just remove the added content at the index of choice and add a new one. To remove the first stored employment use `myGooglePlusProfile.removeEmployment("0");`. Every "adding" function has a remove function so feel free to use them.

1.3 Implementation of GooglePlusProfile

Core of the GooglePlusProfile is a profile object. This profile object is the only member variable in GooglePlusProfile all information is stored in this profile. To create a GooglePlusProfile use the GooglePlusProfileFactory function .createGooglePlusProfile(). At one point these function needs to create a GooglePlusProfileImpl object and to create these object it need the public constructor of GooglePlusProfileImpl. The constructor needs two parameters. First a profile object and second a boolean value. Profile is so important because every information is stored in it and the boolean value is important because it remembers if the object exists already or if it needs to create all entries. If the boolean value is true the constructor will create:

- name entry

- story entry
- work entry
- education entry
- places entry
- basic information entry
- contact information entry
- link entry

If the boolean value is false it means these entries or object already exist and don‘t need to be created.

1.3.1 Example implementation of entry contact information

```
public class GooglePlusProfileImpl implements GooglePlusProfile {

    //Profile member variable stores all information
    private Profile p;

    //Constructor which is explained above
    GooglePlusProfileImpl(Profile p, Boolean isNew) throws SharkKBException {
        this.p = p;
        if (isNew){
            createNameEntry();
            createStory();
            createWork();
            p.createProfileEntry(EDUCATION);
            p.createProfileEntry(PLACES);
            createBasicInformation();
            createContactInformation();
            createLink();
        }
    }
    ...
```

```
//Creates the skeleton or structure of contact information entry
//Is used in the constructor if second constructor parameter is true
private void createContactInformation() throws SharkKBException {
    p.createProfileEntry(CONTACTINFORMATION);
    p.addSubEntryInEntry(CONTACTINFORMATION, HOME);
    p.addSubEntryInEntry(CONTACTINFORMATION, WORK);
}

...


//################ContactInformationSection###############
public void addHomeContact(String contactType, String contactInfo)
        throws SharkKBException {
    List<Entry<?>> entryList = new ArrayList<Entry<?>>();
    entryList.add(new EntryImpl<String>(CONTACTTYPE, contactType));
    entryList.add(new EntryImpl<String>(CONTACTINFO, contactInfo));

    int count = p.getSubEntry(CONTACTINFORMATION, HOME).
        getEntryList().size();
    p.createSubEntry(CONTACTINFORMATION, HOME,
        Integer.toString(count), entryList);
}

public String getHomeContactType(int contactNumber)
        throws SharkKBException {
    List<Entry<?>> entryList =
        (List<Entry<?>>) p.getSubEntry(CONTACTINFORMATION, HOME)
        .getEntryList().get(contactNumber).getContent();
    return (String) entryList.get(0).getContent();
}

public String getHomeContactInfo(int contactNumber)
        throws SharkKBException {
    List<Entry<?>> entryList =
        (List<Entry<?>>) p.getSubEntry(CONTACTINFORMATION, HOME)
        .getEntryList().get(contactNumber).getContent();
```

```
    return (String) entryList.get(1).getContent();
}

public void removeHomeContact(String contactNumber)
        throws SharkKBException {
    p.removeEntryFromSubEntry(CONTACTINFORMATION,
        HOME, contactNumber);
}

public void addWorkContact(String contactType, String contactInfo)
        throws SharkKBException {
    List<Entry<?>> entryList = new ArrayList<Entry<?>>();
    entryList.add(new EntryImpl<String>(CONTACTTYPE, contactType));
    entryList.add(new EntryImpl<String>(CONTACTINFO, contactInfo));

    int count = p.getSubEntry(CONTACTINFORMATION, WORK)
        .getEntryList().size();
    p.createSubEntry(CONTACTINFORMATION, WORK, Integer
        .toString(count), entryList);
}

public String getWorkContactType(int contactNumber)
        throws SharkKBException {
    List<Entry<?>> entryList =
        (List<Entry<?>>) p.getSubEntry(CONTACTINFORMATION, WORK)
        .getEntryList().get(contactNumber).getContent();
    return (String) entryList.get(0).getContent();
}

public String getWorkContactInfo(int contactNumber)
        throws SharkKBException {
    List<Entry<?>> entryList =
        (List<Entry<?>>) p.getSubEntry(CONTACTINFORMATION, WORK)
        .getEntryList().get(contactNumber).getContent();
    return (String) entryList.get(1).getContent();
}

public void removeWorkContact(String contactNumber)
```

```
            throws SharkKBException {
        p.removeEntryFromSubEntry(CONTACTINFORMATION,
            WORK, contactNumber);
    }

    ...

}
```

`private void createContactInformation() throws SharkKBException;` creates entry structure. First it creates the super entry CONTACTINFORMATION: p.createProfileEntry(CONTACTINFORMATION); CONTACTINFORMATION is a static constant and is stored in GooglePlusProfile which is the interface of GooglePlusProfileImpl. Then it creates two sub entries under the super entry CONTACTINFORMATION:

- p.addSubEntryInEntry(CONTACTINFORMATION, HOME);
- p.addSubEntryInEntry(CONTACTINFORMATION, WORK);

Now there is a contact information entry which contains a home contact entry and a work contact entry. This process of creating entry structure is the same for every other entry in GooglePlusProfile. A home or work contact should contain a type of contact like is it a mobile phone, a fax, an address and so on and it should contain the actual information. If contact type is mobile phone then the contact information should be something like: 0172-1235412. Functions to add contact type and information are needed for work and home contacts. Because the implementation of home and work contact functions are nearly the same only home contact implementation will be explained in the following. `public void addHomeContact(String contactType, String contactInfo) throws SharkKBException;` contact type and information are given as parameters, so the addHomeContact(...) function can take and store them. An entry list is created. These list stores both parameters as content.

```
List<Entry<?>> entryList = new ArrayList<Entry<?>>();
entryList.add(new EntryImpl<String>(CONTACTTYPE, contactType));
entryList.add(new EntryImpl<String>(CONTACTINFO, contactInfo));
```

Multiple contacts should be stored, so an index for those is needed. These index is calculated. First entry HOME is searched especially its entry list and the size of this list: `int count = p.getSubEntry( CONTACTINFORMATION, HOME).getEntryList().size();` Finally the created entry list is stored under home contact entry with the calculated index as identifier: `p.createSubEntry (CONTACTINFORMATION, HOME, Integer.toString(count), entryList);` Next step is the implementation of getter functions. A home contact entry stores two parameters so two getters are needed. One getter for home contact type and one for home contact information. Implementing getters for information stored in entries is always a bit tricky, because entries are generic and so is there content. To extract generic content it is important to know the type of the generic content. `getHomeContactType(int contactNumber) throws SharkKBException;` first searches the entry list of home contact entry. These entry list contains all stored home contacts. Parameter contactNumber is the index and verifies which list element out of home contacts entry list should be used. If the list element is selected the content of this element should be casted to a list of unknown entries and stored in variable entryList: `List<Entry<?>> entryList = (List<Entry<?>>) p.getSubEntry ( CONTACTINFORMATION, HOME) .getEntryList() .get(contactNumber).getContent();` The first list element is selected and content should be extracted and casted to string. Finally these casted content is returned. `return (String) entryList.get(0). getContent();` The implementation of `getHomeContactInfo(int contactNumber)` is exactly the same, but one little difference: The second element out of the correct home contact entry list element content is selected and not the first one.