IMiGEr architecture

Tomáš Šimandl

September 2018

1 Introduction

Interactive Multimodal Graph Explorer (IMiGEr) is application for visualization of diagrams. Diagram can be loaded from Spade JSON file. Application transform Spade JSON to own JSON format which is send to front-end and displayed to user as diagram. Loged-in user can save and load own diagrams from database.

2 Packages description

api contains classes representation of JSON which is used between front-end and back-end. Root element is class `Graph`.

graph contains representation of JSON from spade including classes for loading (package `graph.loader`) and filtering (package `graph.filter`). Root element is represented by class `GraphManager`. `GraphManager` process all filtration and return instance of `api.Graph` class.

servlets contains servlets which have some view (Upload files view, show graph view). Other servlets are in package `servlets.api`. This servlets are for example for login, removing or loading graphs.

storage contains classes which can work with file system or load files from multipart request.

user contains logic and models for accessing the database.

vo contains objects which are directly send to front-end.

configuration contains configuration which is loaded from `web.xml` file.

3 Class description

In this section will be described only the most important classes.

3.1 servlets.api.LoadGraphData

Incoming request can be processed by three different ways according to parameters and values in session.

- Session contains `demo_id` value. In that case is load and visualize diagram which is stored in server file system.
- When request contains `diagramId` attribute. It is loaded diagram from database. If diagram is private and user is not log-in unauthorized code is returned.
- If booth previous options are not fulfilled. JSON of diagram is loaded from session where was save by `UploadFiles` servlet. JSON is filtered and converted to front-end back-end JSON and send to browser.

3.2 servlets.ShowGraph

It is used only for send HTML created with `showGraph.jsp` to browser. Representing main page with diagram. When page is loaded browser send automatically request to `/api/load-graph-data` which return JSON of diagram.

3.3 servlets.UploadFiles

Except rendering of `uploadFiles.jsp` is servlet used to load JSON files from user. When file is successfully loaded it is saved to session for later processing. Servlet representing page for uploading new diagrams and opening existing diagrams.

3.4 graph.loader.GraphJSONDataLoader

This class is used for loading Spade diagram from JSON file. Loading is provided by method `LoadData` which returns object of `GraphManager` which contains loaded data.

3.5 graph.loader.JSONConfigLoader

Class is used for loading of configuration from file which is used for loading of Spade diagram. Configuration file contains default filter, vertex archetype icons and conditions for creating default groups.

3.6 graph.GraphManager

Class represents loaded Spade diagram without filtration. Contains information about vertices, edges, archetypes and their attributes, information about what kind of vertex and edge archetypes are in diagram, list of all possible attributes and list of all possible values in attributes. Method `createGraph` apply filters to data and create instance of class `api.Graph` which is returned. Information about filtration is added to method via class `JSONConfigLoader` which is used for loading configuration from configuration file.

3.7 graph.filter.GraphFilter

This class contains information about spade graph filtration. Specifically:

- What kind of vertex archetypes should be in diagram. This information is stored with `VertexArchetypeFilter` class.
- What kind of edge archetypes should be in diagram. This information is stored with `EdgeArchetypeFilter` class.
- Information about vertex and edge attribute filters. That means information what values should attributes have to be in graph. This information is stored with `AttributeFilter` class.

4 Flow diagram

4.1 Visualization of new Spade graph

When new file is uploaded to server and is successfully loaded it is saved to session and server send redirect to `/graph` URL. Servlet `ShowGraph` which handle requests of `/graph` URL, only return new HTML page. JavaScript on page starts loading graph and send request to `/api/load-graph-data` URL which handle `LoadGraphData` servlet. Servlet get graph from session and start filtering and transformation to back-end front-end JSON format. Created JSON is send back to JavaScript which visualize it to user. Flow is illustrated in figure 1.

5 Database

Database in application is used for users and diagrams. Diagram can be saved to database only by log-in user. ER diagram is illustrated in figure 2. Diagrams which are public are visible to everyone and private diagrams are only for log-in user. It is not possible to create public diagram in application GUI now.

6 Configuration

All application configuration is stored in `web.xml`. Most important is to set location of Spade configuration folder (`configLocation`), database credentials (`DbUser, DbPsw`) and database url (`DbUrl`). Name of configuration file must be `config.json`.

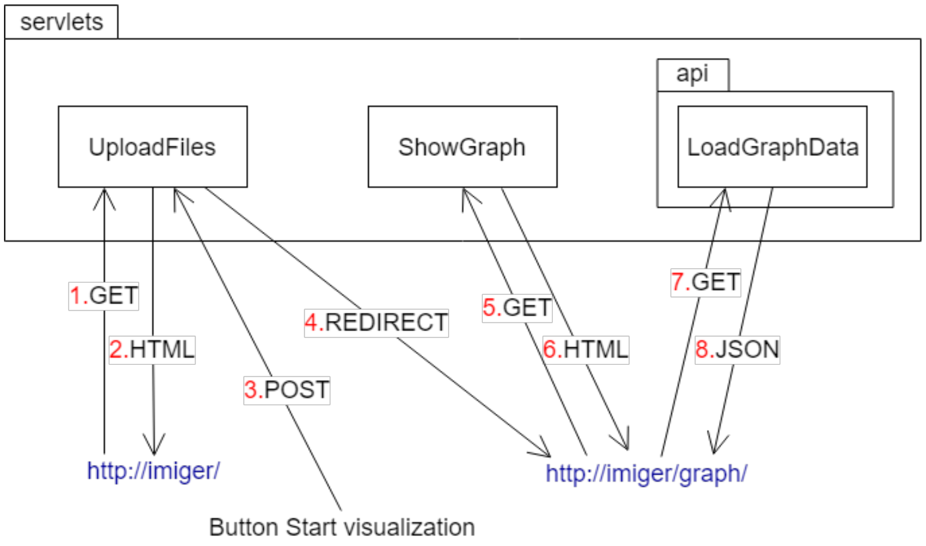

Figure 1: Flow chart of upload and visualization of new diagram. Communication between browser and servlets.

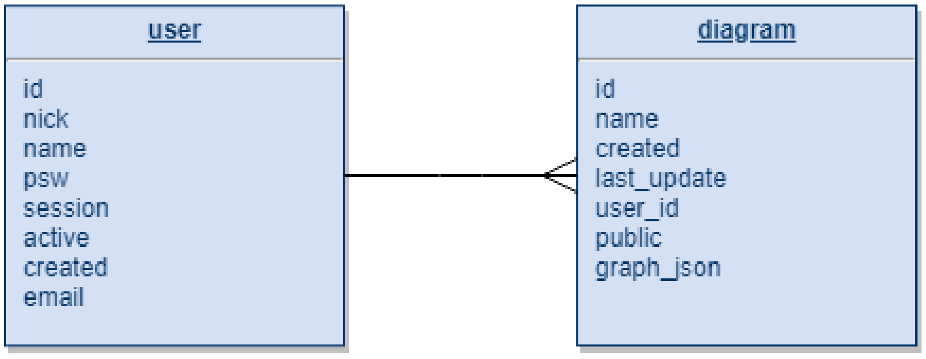

Figure 2: Entity-Relationship Model of used database.

7 Tests

On back-end are created unit tests only for filters. Rest of functionality is not tested.

Front-end tests was created for ASWI 2018 version of application and now are not valid. Front-end tests are disabled.