Entering data into DiVA via Puppeteer

G. Q. Maguire Jr.

This document describes my efforts to facilitate the process of entering metadata and uploading of the approved thesis into DiVA. This is one of administrative processes associated with degree projects that I have tried to automate in conjunction with several 1st cycle degree projects.

An assumption in this work is that the earlier work reported in the previous 1st cycle degree projects has been implemented and that there is a file that has been extracted that contains the information that should be reported in DiVA. This file is described in Section 1.

This work uses the Puppeteer library for NodeJS to interact with a web browser to interact with a web browser instead of requiring a user to manually make entries via the browser. Puppeteer will be described in in Section 4.

Currently, to enter metadata and uplo9ad the approved thesis into DiVA is done with the DiVA graphical user interface (GUI) running in a browser. This GUI will be described in Section 5.

Additionally, there will be some material about the implementation is given in Section 6 and overall conclusions of this work in Section 7.

# Metadata about a thesis

The metadata for a thesis is in a file in JSON format. An example of such a file is shown in Listing 1‑1. Note that the data in this example is for a completely fictitious thesis. The content is only for illustrative purposes. The fields of the structure correspond to the text areas of the DiVA data form from xxx.

Listing 1‑1: thesis\_info.json

|  |
| --- |
| {  "Author1":{  "Last name": "Maguire Jr.",  "First name": "Gerald Q.",  "Local User Id": "u1d13i2c",  "Research group": "CCS",  "E-mail": "maguire@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS"  }  },  "Author2":{  "Last name": "Noz",  "First name": "Marilyn E.",  "E-mail": "men@bogus.org",  "Other organisation": "NYU"  },  "Cooperation":{  "Partner\_name": "ABBBBA"  },  "Title":{  "Main title": "This is a long title in English",  "Subtitle": "This is an even longer subtitle in English",  "Language": "eng"  },  "Alternative title":{  "Main title": "Detta är en lång titel på svenska",  "Subtitle": "Detta är en ännu längre undertexter på svenska",  "Language": "swe"  },  "Degree":{  "Level": "Independent thesis Basic level (degree of Bachelor)",  "University credits": "15 HE credits",  "Educational program": "Bachelor of Science in Engineering - Computer Engineering",  "Subject\_course": "Communications Systems"  },  "Content category":{  },  "Other information":{  "Year": "2019",  "Number of pages": "xiii,72"  },  "Series":{  "Title of series": "TRITA-ICT-EX",  "No. in series": "2019:00"  },  "Other series":{  "Title of series": "",  "ISSN": "",  "EISSN": "",  "No. in series": ""  },  "Identifiers":{  "ISRN": "",  "DOI": "",  "DOI\_Free\_full\_text": "",  "URL": "",  "URL label": "",  "URL\_Free\_full\_text": ""  },  "National subject category":{  "L1": "Engineering and Technology",  "L2": "Electrical Engineering, Electronic Engineering, Information Engineering",  "L3": "Communication Systems"  },  "Part of project":{  },  "Part of other project":{  "Project\_name": ""  },  "Keywords1":{  "Keywords": "Fiddle,Fee,Foo,Fum",  "Language": "eng"  },  "Keywords2":{  "Keywords": "Faddle,Fåå,Fää,Fööm",  "Language": "swe"  },  "Abstract1":{  "Abstract": "<p>This is a abstract for an non existant thesis about <sup>18</sup>F<sup>-</sup></p>",  "Language": "eng"  },  "Abstract2":{  "Abstract": "<p>Detta är ett abstrakt för en icke-existerande avhandling om <sup>18</sup>F<sup>-</sup></p>",  "Language": "swe"  },  "Supervisor1":{  "Last name": "Västberg",  "First name": "Anders",  "Academic title": "universitetslektor",  "Local User Id": "u1ft3a12",  "Research group": "RS",  "ORCiD": "0000-0002-4226-9652",  "E-mail": "vastberg@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS",  "L3": "Radio Systems Laboratory (RS Lab)"}  },  "Supervisor2":{  "Last name": "Normal",  "First name": "A. B.",  "E-mail": "ABNormal@example.org",  "Other organisation": "Famous Anvils"  },  "Examiner1":{  "Last name": "Maguire Jr.",  "First name": "Gerald Q.",  "Academic title": "professor",  "Local User Id": "u1d13i2c",  "ORCiD": "0000-0002-6066-746X",  "Research group": "CCS",  "E-mail": "maguire@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS",  "L3": "Radio Systems Laboratory (RS Lab)"}  },  "Presentation":{  "Date": "2019-07-25 4:31",  "Language": "eng",  "Room": "Seminar room Grimeton at COM",  "Address": "Kistagången 16, East, Floor 4, Elevator B",  "City": "Kista"  },  "Note" :{  "Note": "<p><span style='color: red;'>A completely bogus entry for testing with Puppeteer using diva5.js</span></p>"  },  "File" :{  "Filename": "/home/maguire/Diva/z1.pdf"  }  } |

## Author fields

There are either one or two authors for a thesis. An example is shown in Listing 1‑2.

Listing 1‑2: Author fields

|  |
| --- |
| "Author1":{  "Last name": "Maguire Jr.",  "First name": "Gerald Q.",  "Local User Id": "u1d13i2c",  "Research group": "CCS",  "E-mail": "maguire@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS"  }  }, |

The information for this field is expected to come from a variety of source. The sources for this information will primarily be from Canvas (as shown in Table 1‑1).

Table 1‑1: Sources of data for the fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| Last name |  | | Canvas |  |
| First name |  | | Canvas |  |
| Local User Id | sis\_user\_id | | Canvas |  |
| E-mail |  | | Canvas |  |
| Research group |  | |  | Based on examiner’s affiliations? |
| organisation |  | |  | Based on examiner’s affiliations? |

Note that the “organization information has 1, 2, or 3 levels – corresponding to School, Department, and Division (in the currently university naming scheme). More information is given about these levels in Section 2

## Supervisor(s) and Examiner fields

The information about the supervisors and examiner has a similar set of fields as the author, but also adds the academic title (if applicable) and the ORCiD fields. Examples are shown in Listing 1‑3

Listing 1‑3: Supervisor and Examiner fields

|  |
| --- |
| "Supervisor1":{  "Last name": "Västberg",  "First name": "Anders",  "Academic title": "universitetslektor",  "Local User Id": "u1ft3a12",  "Research group": "RS",  "ORCiD": "0000-0002-4226-9652",  "E-mail": "vastberg@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS",  "L3": "Radio Systems Laboratory (RS Lab)"}  },  "Supervisor2":{  "Last name": "Normal",  "First name": "A. B.",  "E-mail": "ABNormal@example.org",  "Other organisation": "Famous Anvils"  },  "Examiner1":{  "Last name": "Maguire Jr.",  "First name": "Gerald Q.",  "Academic title": "professor",  "Local User Id": "u1d13i2c",  "ORCiD": "0000-0002-6066-746X",  "Research group": "CCS",  "E-mail": "maguire@kth.se",  "organisation": {"L1": "School of Information and Communication Technology (ICT)",  "L2": "Communication Systems, CoS",  "L3": "Radio Systems Laboratory (RS Lab)"}  }, |

The information for this field is expected to come from a variety of source. The sources for this information will primarily be from Canvas (as shown in Table 1‑1).

Table 1‑2: Sources of data for the fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| **Last name** |  | | Canvas |  |
| **First name** |  | | Canvas |  |
| **Local User Id** | sis\_user\_id | | Canvas |  |
| **E-mail** |  | | Canvas |  |
| **Research group** |  | |  | Based on KTH affiliation |
| **organisation** |  | |  | Based on KTH affiliation |
| **Other organisation** |  | |  | For people external to KTH |
| **Academic title** |  | | KTH API |  |
| **ORCiD** |  | | KTH API |  |

Note that the “organization information has 1, 2, or 3 levels – corresponding to School, Department, and Division (in the currently university naming scheme). More details on the organization data is given in Section 2.

## Titles, Abstracts, Keywords, and Number of pages

These fields (shown in the example in Listing 1‑4) are expected to be extracted from the thesis (either in PDF for DOCX format) together with the “Number of pages” field. Extraction of these fields was described in the thesis:

Shiva Besharat and Qi Li, Connecting Silos: Automation system for thesis processing in Canvas and DiVA. Bachelor's thesis, Stockholm, Sweden: KTH: Skolan för elektroteknik och datavetenskap (EECS): Kommunikationssystem, CoS: Radio Systems Laboratory (RS Lab), 2018, TRITA-EECS-EX-2018:164 [Online]. Available: [http://urn.kb.se/resolve?urn=urn%3Anbn%3Ase%3Akth%3Adiva-230996](http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-230996)

The year will generally come from the (year of the) date that the thesis is submitted.

The assumption is that the main title will be in the language of the thesis and the alternative title will be in the other language (or English and Swedish). The language of the first abstract and keywords are assumed to be the same as the language of the title. Similarly, the language of the second abstract and keywords are assumed to be the other language. For, the current situation we will assume that the language of the thesis can be established by guessing using the frequent stop words of English and Swedish.

Note that the language is specified using the three letter ISO abbreviation for the language (i.e., English is “eng” and Swedish is “swe”). These same three letter abbreviations are used in DiVA.

Listing 1‑4: Fields to be extracted from the thesis

|  |
| --- |
| "Title":{  "Main title": "This is a long title in English",  "Subtitle": "This is an even longer subtitle in English",  "Language": "eng"  },  "Alternative title":{  "Main title": "Detta är en lång titel på svenska",  "Subtitle": "Detta är en ännu längre undertexter på svenska",  "Language": "swe"  },  "Other information":{  "Year": "2019",  "Number of pages": "xiii,72"  },  "Keywords1":{  "Keywords": "Fiddle,Fee,Foo,Fum",  "Language": "eng"  },  "Keywords2":{  "Keywords": "Faddle,Fåå,Fää,Fööm",  "Language": "swe"  },  "Abstract1":{  "Abstract": "<p>This is a abstract for an non existant thesis about <sup>18</sup>F<sup>-</sup></p>",  "Language": "eng"  },  "Abstract2":{  "Abstract": "<p>Detta är ett abstrakt för en icke-existerande avhandling om <sup>18</sup>F<sup>-</sup></p>",  "Language": "swe"  }, |

## Series and numbering

The primary series identification for theses is the TRITA number. The example shown in Listing 1‑5 uses the “TRITA-ICT-EX” series. The thesis is assigned a unique number from this series, in the form YYYY:dd. In the process that I have implemented in my tests Canvas programs the series number is automatically assigned from a database record that automatically increments with use (to guarantee that the number is unique – however, there could be gaps).

The fields “DOI Free full text” and “URL Free full text” are used to set a checkbox to indicate that the DOI or URL points to a freely available full text version of the report. Currently, most of these other identifiers are ignored by the current program.

Listing 1‑5: Series information

|  |
| --- |
| "Series":{  "Title of series": "TRITA-ICT-EX",  "No. in series": "2019:00"  },  "Other series":{  "Title of series": "",  "ISSN": "",  "EISSN": "",  "No. in series": ""  },  "Identifiers":{  "ISRN": "",  "DOI": "",  "DOI\_Free\_full\_text": "",  "URL": "",  "URL label": "",  "URL\_Free\_full\_text": ""  } |

The sources for this information will primarily be from Canvas (as shown in Table 1‑3).

Table 1‑3: Sources of data for the fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| Title of series |  | |  | Statically chosen for a school and bit into the program. |
| No. in series |  | | database |  |

Note that the current assignment of TRITA numbers is done manually by administrators who record the information in a spreadsheet – which has resulted in multiple theses being assigned the same number – that later has to be corrected.

Note also that the title of the series and the Number in the series are used together with the title, subtitle, and author(s) names to generate a cover for the thesis.

## Project affiliation

There is a field to record a degree project being done as part of a a project. However, I have never used this field – so I have ignored it in the current program.

Listing 1‑6: Project information

|  |
| --- |
| "Part of project":{  },  "Part of other project":{  "Project\_name": ""  }, |

## External cooperation (industrial projects)

KTH’s external relations office is trying to track which degree projects have been done with external partners. This field is used to record the name of the external partner (if there is one). If there is not partner name specified, the check box for this field is set to false.

Listing 1‑7: External cooperation field

|  |
| --- |
| "Cooperation":{  "Partner\_name": "ABBBBA"  } |

The sources for information will be from Canvas (as shown in Table 1‑3).

Table 1‑4: Sources of data for the field

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| Partner name |  | | Canvas | Affiliation of the external supervisor (from a column of data in the gradebook) |

## National subject category

A required field in an entry for a student thesis is the national subject category.

Listing 1‑8: National subject category information

|  |
| --- |
| "National subject category":{  "L1": "Engineering and Technology",  "L2": "Electrical Engineering, Electronic Engineering, Information Engineering",  "L3": "Communication Systems"  } |

## Degree related information

The degree related information shown in Listing 1‑9 will be based on the student’s information (**Level**, **Educational program**, and **Subject\_course**) and from the degree project course code’s associated information (credits come from KOPPS). In earlier work I have proposed adding the student’s program of student and track to Canvas as custom information about the student. Here we will assume that this information is available via Canvas and has been added either to the student’s record or is available based on the course code for the degree project.

Listing 1‑9: Degree related information

|  |
| --- |
| "Degree":{  "Level": "Independent thesis Basic level (degree of Bachelor)",  "University credits": "15 HE credits",  "Educational program": "Bachelor of Science in Engineering - Computer Engineering",  "Subject\_course": "Communications Systems"  } |

## Content category

This field is used if the result is an artistic work (as opposed to a written report). I do **not** know where this information about be found.

Listing 1‑10: Content category

|  |
| --- |
| "Content category":{  } |

## Presentation fields

The data for the presentation shown in Listing 1‑11 is expected to come from Canvas – as this information is also expected to be used in the automatically generated announcement for the seminar. The address information should come from a table. Currently, this data is not completely correct in the Outlook Calendar.

Listing 1‑11: Presentation data

|  |
| --- |
| "Presentation":{  "Date": "2019-07-25 4:31",  "Language": "eng",  "Room": "Seminar room Grimeton at COM",  "Address": "Kistagången 16, East, Floor 4, Elevator B",  "City": "Kista"  } |

## Notes and file fields

The data for a note and file name (where the PDF with cover applied) are provided as shown in Listing 1‑12. My assumption is that the file name will be provided by the program that applies the cover to the thesis submitted via Canvas.

The note field could be used to communicate information to the DiVA administrators – for example that this entry was created by a program.

Listing 1‑12: Notes and File fields (file field for Archive only)

|  |
| --- |
| "Note" :{  "Note": "<p><span style='color: red;'>A completely bogus entry for testing with Puppeteer using diva5.js</span></p>"  },  "File" :{  "Filename": "/home/maguire/Diva/z1.pdf"  } |

In addition to simply having the file uploaded for archive purposes, it is also possible to have the full text made now (Listing 1-13) or on a specific date (Listing 1-14).

Listing 1‑13: File field for making the full text available now

|  |
| --- |
| "File" :{  "Filename": "/home/maguire/Diva/z1.pdf",  "Accept full text": "true",  } |

Listing 1‑14: File field for making the full text available on the stated date

|  |
| --- |
| "File" :{  "Filename": "/home/maguire/Diva/z1.pdf",  "Accept full text": "true",  "Available Date": "2019-08-25 01:00"  } |

**Making the full text available can be known from the Canvas gradebook entry.** Note that this entry in the gradebook should be extended with the ability to specify a date. Note that the date must include the time – otherwise the “Choose file” GIF will not appear!

# Organization levels

Currently in KTH there are up to 3 levels of organizational affiliation: School, Department, and Division. Not that in some parts of the hierarchy there are centers (‘Centres’) that can be at the Department level. We will refer to the three levels as L1, L2, and L3. In the old organization there was one case of a 4th level within a center, but it is ignored in my programs.

Unfortunately, the DiVA graphical user interface uses a tree structured interface for the organization affiliation entry and for entering the National Subject Category. This tree is dynamically fetched from the server and the tree is organized alphabetically. To make the process of going from string names for headings of L1, L2, and L3 an external file is used to store the tree and provide information to the data entry program of which parts of the tree to access to enter an affiliation. This data is stored in one of two files: kth-org-diva-old.json and kth-org-diva-new.json. The “old” version of the organization as used in the test instance of DiVA (kth.test.diva-portal.org) and the new version of the organization as used in the current production instance of DiVA (kth.diva-portal.org).

To simplify the process of creating this tile, there is a support program called extract-organization-info.js that programmatically extracts the entire organization tree. The output of the program is a JSON file (an example of the output file is org-data-2019-08-01T09:12:13.493Z.json) that can be easily edited to produce the JSON file used for the configuration. Currently the file contains a JSON structure of the form shown in Listing 2‑1.The file is a JSON structure with “blanks” and “kth\_org” elements. The “blanks” are the ID’s of the parts of the Author field that were used to fine the ID of the "Choose organisation »" field to invoke the organization popup. The top few lines of the resulting kth-org-diva-\*.json file are shown in Listing 2‑2. The numbering in the tree starts from zero and has the generic format school-department-division.

Listing 2‑1: Example of put from the support program

|  |
| --- |
| {"**blanks**":{"Author":{"Connect authority record »":{"ID":"addForm:authorSerie:0:j\_id676","Type":"submit"},"Get saved personal data »":{"ID":"addForm:authorSerie:0:j\_id691","Type":"submit"},"Save personal data »":{"ID":"addForm:authorSerie:0:j\_id692","Type":"submit"},"Last name":{"ID":"addForm:authorSerie:0:autFamily","Type":"text"},"Year of birth":{"ID":"addForm:authorSerie:0:autBirthYear","Type":"text"},"First name":{"ID":"addForm:authorSerie:0:autGiven","Type":"text"},"Local User Id":{"ID":"addForm:authorSerie:0:j\_id716","Type":"text"},"Choose organisation »":{"ID":"addForm:authorSerie:0:j\_id736","Type":"submit"},"Research group":{"ID":"addForm:authorSerie:0:researchGroup","Type":"text"},"E-mail":{"ID":"addForm:authorSerie:0:autEmail","Type":"text"},"Other organisation":{"ID":"addForm:authorSerie:0:j\_id753","Type":"text"}}},  "**kth\_org**":{"Centres":{"ID":"0","Nordic Institute for Theoretical Physics NORDITA":{"ID":"0-0"} … } |

Listing 2‑2: Top of the kth-org-diva-\*.json file

|  |
| --- |
| {"Centres":{  "ID":"0",  "Nordic Institute for Theoretical Physics NORDITA":{"ID":"0-0"},  "Science for Life Laboratory, SciLifeLab":{  "ID":"0-1",  "KTH Center for Applied Precision Medicine (KCAP)":{"ID":"0-1-0"}  },  "SeRC - Swedish e-Science Research Centre":{"ID":"0-2"},  "XPRES, Excellence in production research":{"ID":"0-3"}  },  "Library":{"ID":"1"},  "School of Architecture and the Built Environment (ABE)":{  "ID":"2",  "Architecture":{  "ID":"2-0",  "Architectural Design":{"ID":"2-0-0"},  "Architectural Technologies":{"ID":"2-0-1"},  "Critical Studies in Architecture":{"ID":"2-0-2"},  "History and Theory of Architecture":{"ID":"2-0-3"},  "Lightning Design":{"ID":"2-0-4"},  "Urban Design":{"ID":"2-0-5"}  }, |

The information for this field is expected to come from the KTH API for the user profiles. The sources for this information will primarily be from Canvas (as shown in Table 2‑1).

Table 2‑1: Sources of data for the organization levels

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| organisation | Uses the sis\_user\_id in Canvas to provide the argument to the call to the KTH API | | KTH API for user profiles |  |

Note that the “organization information has 1, 2, or 3 levels – corresponding to School, Department, and Division. There should be a check that the KTH API data for a user’s profile returns the names of the levels in a form that matches the DiVA names – otherwise, there needs to be some translation of them.

# National Subject Categories extractions

DiVA’s National Subject Catergories GUI also uses a tree GUI. Similar to the organsation tree a support program has been written to extract the data. An example of such an output file is national-subject-catergory-data-2019-08-01T13:20:35.101Z.json. The initial part of this file is shown in Listing 3‑1. This was used to produce a pair of files: National\_subject\_categories-old.json and National\_subject\_categories-new.json' (corresponding to the test and the production DiVA instances respectively). The first few lines of the production version’s catergories are shown in Listing 3‑2.

Listing 3‑1:Output of Categories

|  |
| --- |
| {"**blanks**":{"National subject category":{"Choose national subject category »":{"ID":"addForm:j\_id1039","Type":"submit"}}},"**nationalSubjectCategory**":{"Agricultural and Veterinary sciences":{"ID":"0","Agricultural Biotechnology":{"ID":"0-0","Genetics and Breeding in Agricultural Sciences":{"ID":"0-0-0"},"Plant Biotechnology":{"ID":"0-0-1"}},"Agricultural Science, Forestry and Fisheries":{"ID":"0-1","Agricultural Science":{"ID":"0-1-0"},… |

Listing 3‑2: Some of the resulting categories

|  |
| --- |
| {"Agricultural and Veterinary sciences":{  "ID":"0",  "Agricultural Biotechnology":{  "ID":"0-0",  "Genetics and Breeding in Agricultural Sciences":{"ID":"0-0-0"},  "Plant Biotechnology":{"ID":"0-0-1"}  },  "Agricultural Science, Forestry and Fisheries":{  "ID":"0-1",  "Agricultural Science":{"ID":"0-1-0"},  "Fish and Aquacultural Science":{"ID":"0-1-1"},  "Food Science":{"ID":"0-1-2"},  "Forest Science":{"ID":"0-1-3"},  "Horticulture":{"ID":"0-1-4"},  "Landscape Architecture":{"ID":"0-1-5"},  "Soil Science":{"ID":"0-1-6"},  "Wood Science":{"ID":"0-1-7"}  }, |

The information for this field is expected to come from the course code and information from KOPPS. The source for this information will the Canvas gradebook and possibly a table (as shown in Table 3‑1). Note that the National Subject Category should be knowable to at least L1 and L2 from the course code – as the degree project course needs to be associated with some L1 and L2. However, the L3 information might not be knowable from the course code, thus either L3 is left out or it needs to be specified in a custom column in the Canvas gradebook.

Table 3‑1: Sources of data for the national subject categories (L1, L2, and possibly L3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | | Data | Source | Notes |
| National subject category | Use the course code from the user’s gradebook entry to index a table of what should be the corresponding National Subject Categories | | Canvas gradebook + table |  |

# Puppeteer

The Puppeteer library is used together with NodeJS to interact with a web browser to interact with a web browser instead of requiring a user to manually make entries via the browser. Puppeteer is easily installed with:

npm install puppeteer

This also installs a version of chromium that matches the library. As a result the Puppeteer script talks to chromium. I have run it non-headless mode – so that you can watch what it is doing while the script runs. You can even manually interact with the web interface.

An important construct that I learned about later is shown in Listing 4‑1. The use of Promise.all(), enables you to wait for **all** of the elements of the vector to complete – before beginning with the next statement. This is in contrast to Promises.race(), where the wait is only to the *first to complete*.

Listing 4‑1: Example of use of Promise.all()

|  |
| --- |
| if (currentPage < pagesToScrape) {  await Promise.all([  await page.click('a.morelink'),  await page.waitForSelector('a.storylink')  ])  } |

This approach is able to eliminate a number of await page.waitFor(xxxx) statements where xxxx was a number of ms to wait.

# DiVA graphical user interface (GUI)

The DiVA GUI is based upon the ICEsoft Technologies’ ICEfaces library (see <http://www.icesoft.org/java/projects/ICEfaces/overview.jsf>). This library is used together with other JSF components to make a very interactive GUI. This library runs on top of AJAX. This means that there is communication between the client running in the browser and the server. The dynamics of this makes the process of walking and interacting with the GUI challenging. This dynamics often means that you have to wait when carrying out operations or thee is a race between the updates being made by the server and the operations that Puppeteer is performing on the page. In other cases, this means that the information that will be displayed via the GUI is not yet available at the browser – so you have to wait for it to load. Unfortunately, in some cases I did not know what selector id to wait for – so I used timed waiting (typically for one to three seconds).

One thing to keep in mind is that the DiVA GUI programs did not write the JavaScript that you will see when looking at the pages, this JavaScript was generated by the ICEfaces library. The idea seems to be that the programs can write Java programs and use the library to realize the GUI – without having to write JavaScript. One side effect of this is that the names and ids of fields have generated names and are not something that a human is likely to have written. In some cases you might be able to guess that an id is for, see for example some of the fields for an author (Table 5‑1). As we can see two of them just have generated names!

Table 5‑1: IDs of some fields for an author

|  |  |
| --- | --- |
| Field | id |
| First name | addForm:authorSerie:0:autGiven |
| Last name | addForm:authorSerie:0:autFamily |
| Year of birth | addForm:authorSerie:0:autBirthYear |
| Local User Id" | addForm:authorSerie:0:j\_id716 |
| Research group | addForm:authorSerie:0:researchGroup |
| E-mail | addForm:authorSerie:0:autEmail |
| Other organisation | addForm:authorSerie:0:j\_id753 |

It is useful to know that for the fields where one can repeat them (such as the author fields) the names of the field are the same but the number after “Serie” has been incremented, i.e., the Last name for the second author will be in a field with the id: addForm:authorSerie:**1**:autFamily.

Because so many of the fields have generated names and ids (consider the button (with an image of text in) that the user clicks to add the organization information: ,"Choose organisation »" is "addForm:authorSerie:0:j\_id736"), I found that the way to understand what the fields were and what the IDs were - involved brute force reading the JavaScript and extracting the elements. My first version a program to fill in a form was based on manually extracting the IDs and then invoking them. My later version of the program had routines to walk each of the fieldset elements to extract the IDs.

Another useful thing to know is that each of the different colored regions of the at <https://kth.test.diva-portal.org/dream/add/add2.jsf> is a textarea, hence you can find all of them with the function shown in Listing 5‑1: getTextAreaHandles. The call to page.$$ passes a selector and returns ElementHandles to all of the elements of the page that match the selector (in this case a <div class= diva2addtextarea' … > element). Using the resulting ElementHandles ou can now access each of the text areas, i.e., the groups of input fields..

Listing 5‑1: getTextAreaHandles

|  |
| --- |
| async function getTextAreaHandles(page) {  let textareaHandles;  textareaHandles = await page.$$('div.diva2addtextarea');  return textareaHandles;  } |

The next thing to look for is a <fieldset> as this is generally used to group the GUI elements together.

# Implementation details

In order to run any of the programs you need a configuration file, similar to that shown in Listing 6‑1.

Listing 6‑1: config.json

|  |
| --- |
| {  "canvas":{  "access\_token": "xxxxx",  "host": "canvas.docker",  "username": "xxxx",  "password": "xxx"  },    "diva":{  "host": "kth.test.diva-portal.org",  "username": "xxx",  "password": "xxxxx"  }    } |

## Running the programs

The programs are all run by NodeJs (in this case the program node) as shown in Table 6‑1

Table 6‑1: Command line entries

|  |  |
| --- | --- |
| Task | Command line |
| To extract the organization information | node extract-organization-info.js |
| To extract the national subject categories | node extract-national-subject-categories-info.js |
| To enter the data for a thesis information | node diva6.js thesis\_info.json |

## Using HTML id attributes

When using Puppeteer to interact with DiVA it is very convenient to use HTML id attributes to select an object. For example, in the case of the (old) organization tree:

// the + for ICT is at '[id="organisationPopupForm:tree-d-9"] a'

// the + for COS is at '[id="organisationPopupForm:tree-d-9-1"] a'

// ICT/COSRSlab is at '[id^="organisationPopupForm:tree:n-9-1-3"] a'

Note that because id values include “:” (colon) characters, you have to quote them to use them in selectors.

However, not all objects have id attributes, so sometimes you have to find them based on the class and the tag.

## Dynamics of the DOM

The DiVA interface is a dynamic document; hence its contents changes as you work. In some cases I found that I had problems with entering data directly into the DOM as it would get lost due to an update by the server of the document. In some cases I used a focus on a field, and then sent the characters to the field (as if they were being typed using the function page.keyboard.sendCharacter(text\_to\_send)). In other cases I dynamically updated the value of a field and this worked (as the local code noticed the changes to the field’s value and updated the server with the changes). An example of the latter is shown in Listing 6‑2: Example of directly modifying the value of a part of the DOM.

Listing 6‑2: Example of directly modifying the value of a part of the DOM

|  |
| --- |
| last\_name='Maguire Jr.';  //selector='[id="addForm:authorSerie:0:autFamily"]';  selector=make\_selector\_from\_id(blanks['Author1']['Last name'].ID);  await page.$eval(selector, (el, value) => el.value = value, last\_name); |

## Logging into the DiVA instance

To start the process of entering data into DiVA one has to log into the DiVA site (either kth.test.diva-portal.org or kth.diva-portal.org). The test site was used for the development of the programs described in this document. The process is shown in Listing 6‑3. The program attempts to access the page at the test or production server called '/dream/info.jsf' . As can be seen in this listing the user is re-directed to other pages to login. The first is a SAML redirect and the second is a page that brings the user to the <http://login.kth.se> server and this passes the information of which site to redirect the user to after they have successfully logged in, i.e. <https://.diva-portal.org/shibboleth>. At this point the code looks in the configuration fule to the the user name of the user who is to login in. It could easily be adapted to also pass the password, but this has not been done due to KTH’s user password policies.

Listing 6‑3: code to get logged in and acc the page to add a document

|  |
| --- |
| //diva1\_url='http://'+host\_name+'/dream/info.jsf'  diva1\_url='http://'+host\_name+'/dream/add/add1.jsf'  console.log("diva1\_url is ", diva1\_url);  await page.goto(diva1\_url, {waitUntil: 'load'});  console.log('FOUND!', page.url());  // This will produce: FOUND! https://saml-5.sys.kth.se/idp/profile/SAML2/Redirect/SSO?execution=e1s1  // since the user is not yet logged in    await page.waitForNavigation();  diva2\_url='http://'+host\_name+'/dream/add/add1.jsf'  console.log("diva2\_url is ", diva2\_url);  await page.goto(diva2\_url, {waitUntil: 'load'});  console.log('FOUND!', page.url());  // This will produce: https://login.kth.se/login?service=https%3A%2F%2Fsaml-5.sys.kth.se%2Fidp%2FAuthn%2FExtCas%3Fconversation%3De2s1&entityId=https%3A%2F%2Fwww.diva-portal.org%2Fshibboleth  if (page.url().includes('https://login.kth.se/login')) {  page.type('#username', config.diva.username);  }  await page.waitForNavigation(); |

As shown in Listing 6‑4, the program now attempts to acces the same two sites again and this time the user will be able to access the page, since they are logged in.

Listing 6‑4: code to get logged in and access the page to add a document

|  |
| --- |
| // 2nd try  //diva1\_url='http://'+host\_name+'/dream/info.jsf'  diva1\_url='http://'+host\_name+'/dream/add/add1.jsf'  console.log("diva1\_url is ", diva1\_url);  await page.goto(diva1\_url, {waitUntil: 'load'});  console.log('FOUND!', page.url());  // This will produce something line: FOUND! https://saml-5.sys.kth.se/idp/profile/SAML2/Redirect/SSO?SAMLRequest=xxxxxxx  await page.waitForNavigation();  // go to the page to add a publication  diva\_add\_url='http://'+host\_name+'/dream/add/add1.jsf'  console.log("diva\_add\_url is ", diva\_add\_url);  await page.goto(diva\_add\_url, {waitUntil: 'load'});  console.log('FOUND!', page.url());  //This will produce: FOUND! https://kth.test.diva-portal.org/dream/add/add1.jsf  // since the user is now logged in |

The page '/dream/add/add1.jsf' as a <select> tag on it so that the user can select which type of document that they want to enter, by passing in the chosen option value. Using the code shown in Listing 6‑5 we can examine the current value and then modify this value (‘studentThesis’) and read its value back to see that it has actually changed. In addition, we trigger a refresh of the page so that the user can visually see that the selected value has actually changed.

Listing 6‑5: Selecting the type of document to add

|  |
| --- |
| // <select class="iceSelOneMnu" id="j\_id15:publicationType" name="j\_id15:publicationType" onblur="setFocus('');" onchange="setF  let pubtype;  pubtype = await page.evaluate(() => {  const t1=document.querySelector("div.diva2addtextplus div.diva2addlistdown2 select.iceSelOneMnu").value  return t1;  });  console.info("pubtype is ", pubtype);  // set the type of document  await page.select('div.diva2addtextplus div.diva2addlistdown2 select.iceSelOneMnu', 'studentThesis');  await page.reload();  await page.waitFor(1000);  //page.waitForNavigation({ waitUntil: 'networkidle0' })  pubtype = await page.evaluate(() => {  const t1=document.querySelector("div.diva2addtextplus div.diva2addlistdown2 select.iceSelOneMnu").value  return t1;});  console.info("pubtype is now ", pubtype);  await page.waitFor(1000); |

# Conclusions

As someone who has written very little JavaScript previously, it took about a week to get used to using Puppeteer to interact with DiVA. It took another week to gain some understanding of promises and how to use Promise.all and Promise.race.