Ontology-Based Website Generation with AGAS: A Warm-up

Diana Teixeira
University of Minho
pg53766@alunos.uminho.pt

José João Almeida
University of Minho, ALGORITMI
jj@di.uminho.pt

Alberto Simões
University of Minho
alberto.simoes@checkmarx.com

Welcome to **Guide 0** of my thesis project!

Your mission: set up AGAS, run it and try out its coolest features. Don't worry, we will take it step by step and hold your hand through it all, as long as you read carefully.

Keep notes of anything tricky (bugs, confusions, surprises) as your feedback is super valuable!

1 Step 1: Setup

1.1 Prerequisites

Make sure you've got your tools:

- Python 3.11+ (3.13 is recommended):
  ```
  python --version
  ```
- Git:
  ```
  git --version
  ```
- Pip:
  ```
  sudo apt-get install python-pip
  ```
- Graphviz:
  ```
  sudo apt install python3-pydot graphviz
  ```

Do you not have them?

Just run:

```
sudo apt install python3.13-full

sudo add-apt-repository ppa:git-core/ppa
sudo apt-get update
sudo apt-get install git
```

1.2 Install AGAS

1. Grab the code:
   git clone https://github.com/dianamalheiro02/agas.git
   cd agas
   Or install directly from github:
   ```
   pip install git+https://github.com/dianamalheiro02/agas.git
   ```
2. Install it:
   ```
   pip install .
   ```

3. Test it:

   ```
   agas
   ```

If you see the help menu, congratulations! You are ready to get started.

Got stuck? Jump to the Troubleshooting (Appendix A).

2 Step 2: First Moves

1. Create your config skeleton:

   ```
   agas -s > G0.txt
   ```

2. Next, **edit the *ONTOLOGY_FILE* directory inside the G0.txt file, at least**:

   ```
   vim G0.txt

   - Or your favorite text editor. -
   ```

3. The new directory can be obtained by running these commands in the project's folder:

   ```
   cd tests/ontos/typeC/C1/ontos_final

   realpath greek_deities_ontology_complete.ttl
   ```

4. After editing, it should look something like this:

   ```
   ONTOLOGY_FILE = '/home/diana/agas/ontos/typeC/C1/ontos_final/greek_deities_
   ontology_complete.tt'
   ```

5. If you wish to edit more fields, refer to Appendix B.

6. Validate it:

   ```
   agas -c G0.txt
   ```

7. Run it:

   ```
   agas -r G0.txt
   ```

8. Open your browser to: `http://localhost:5000`

Well done! You now have your very own ontology-powered website up and running.

2.1 Explore Like an Adventurer

Once AGAS is running, explore its **browsing and reading** features.

1. Start on *Home* - check the taxonomy on the left.
2. Mark some classes and individuals as favorites.
3. Visit one of your favored individuals and explore their information (e.g., links to other classes or individuals).
4. Check if navigation feels smooth, natural and intuitive.
5. Switch to *Extra* and try out the projection editor (e.g. explore Classes, Properties, Individuals). - **Do not be alarmed if it takes a while to load or if it needs some refreshing!**
6. Visit the *RDF Graph* page and download one of the generated graphs.

2.2 Create, Delete, Edit (the Fun Part)

1. Log in:

   ```
   Username: AGAS_Admin
   Password: AGAS_Admin_Password
   ```

2. Search "**Zeus**" and click *Edit*.
3. Try these tweaks:
   - Change father from *Cronus* to *Perses*;
   - Adjust stories;
   - Change his powers;
4. Hit **Next**, then **Confirm** the changes and check the results!
5. **BONUS**: Try raw editing mode for a more "coder" vibe. **This only works if the *USER_TYPE* variable is set as *'EXP'* in G0.txt.**
6. From the *Home* page, try adding a new class(e.g. 'My_Class'), property(e.g. 'hasPropertyEX'), and individual(e.g. 'My_Individual').
7. Confirm their existence by going to the projection editor, *Full ontology* section and looking them up.
8. Then *Delete* them. Can you make it disappear completely? **If you're using the projection editor, don't forget to *Save Ontology* after!**

2.3 Versions and Downloads

1. Go to *Versions* and explore the history of your changes.
2. Locate the first add you did by checking the version's information.
3. Confirm your edits vanish and that the ontology information changes from one version to another.
4. Travel back in time to one of those edits and prune history by deleting some versions. **Never delete the version the system is using!**
5. Notice how all the changes you made return and how you can see the new adds you did.
6. Go to the *About* page and download the ZIP archive containing all application data.

3 Bonus Challenges (Optional Quests)

- Run AGAS with a different ontology from the */tests* folder. **Never forget to change the ONTOLOGY_FILE variable in each configuration file you want to run!**
- Modify the DSL configuration files and see how the website changes.
- Experiment with larger ontologies (e.g. type B) - How does it handle it?
- Explore the editing features more freely (e.g. making changes to classes/properties/individuals).
- Try converting the ontology in the *Extra* page, or even editing it with *Protégé*.
- Play with favorites, navigation, and search across the different ontologies and data structures.

4 Final Step

Once done, please answer the questionnaire https://forms.gle/yJGaA5AUBB1PTTqw9.

Your feedback is essential to evaluate both the usability of AGAS and its effectiveness as a research tool. These questions will ask about your impressions of the DSL, navigation, editing features, and overall interaction with the platform.

And, that's it! You've completed **Guide 0**. Thanks for helping test AGAS, your journey makes this research stronger.

A Troubleshooting

A.1 Problem 1: '*pip install .*' fails

Possible causes and solutions are:

1. **'*pyproject.toml*' not found**:
   Make sure you are inside the correct folder when running the command:
   ```
   pip install .
   ```
2. **'*build*' backend not installed**:
   Install it manually:
   ```
   python3 -m pip install flit
   ```
3. **Permission denied (especially on Linux/macOS)**:
   ```
   pip install --user .
   ```
   Or try using a virtual environment to run this on.
4. **Old pip version**:
   ```
   pip install --upgrade pip
   ```
5. **error: externally-managed-environment**:
   Try and install 'pipx' and see how your computer behaves:
   ```
   sudo apt install pipx
   pipx install .
   pipx ensurepath

   - And then reopen the terminal -
   ```

A.2 Problem 2: Running '*agas -r tests/config.txt*' doesn't work

Possible causes and solutions are:

1. **ModuleNotFoundError: No module named '*agas*'**:
   Check that installation was a success with:
   ```
   pip show agas
   ```
   And if not installed, re-run:
   ```
   pip install .
   ```
2. **Wrong working directory**: Ensure you are running the command from the project root, at the same level as the '*pyproject.toml*'.
3. **Flask not installed**:
   ```
   pip install flask
   ```
4. **ModuleNotFoundError**:
   Some dependencies may be missing from pyproject.toml, please install manually and try to run again with 'pip' or 'pipx'!
   ```
   pipx uninstall agas
   pipx install .
   ```

B DSL Structure

B.1 Ontology Configuration

This section defines the ontology file path, its type (A, B, C1, or C2), and the Protégé executable location for ontology editing.

ONTOLOGY_FILE: *Points to the ontology file used for the website generation.*

PROTEGE_PATH: *Allows AGAS to launch Protégé for ontology modifications.*

ONTOLOGY_TYPE: *Determines how AGAS interprets the ontology (e.g., structured vs. complex knowledge representation).*

ONTOLOGY_IMAGES: *To specify if the ontology has images that are from your own desktop ('NONE' if it doesn't), if it does, please give the path to the folder you have them in.*

ONTOLOGY_EDIT: *Specify who can edit the ontology exactly (e.g, every single person can edit it and its information with the keyword 'ALL' or only people who 'LOGIN'.*

B.2 User Experience Customization

The DSL allows the system to tailor the website generation experience based on the user type parameter, called **USER_TYPE** and that can have two different specifications:

- **'EXP'** *(Experienced user): Provides advanced customization options.*
- **'NONEXP'** *(Non-experienced user): Offers a simpler, more guided experience.*

B.3 Template and Language Selection

Users can specify which Jinja templates should be used for rendering the website and what language they want to use.

TEMPLATES: *Specifies which Jinja templates define the page layouts.* ***LANGUAGE:*** *Define what language you want to use, 'PT' or 'EN'.*

B.4 Visualization Preferences

Users can define what elements should be displayed on generated pages.

RDF_VIEW: *What you want to see in the Graph, which classes you want present. You can write down 'ALL' if you want them all, but if not, please list the classes you want.*

VIEW_CLASSES: *Specifies which ontology classes are important to be visualized.*

SPECIFIC_PAGES: *Where users can list which pages are more important to generate, pages of specific individuals or entities.*

MAKE_PRETTY: *Specifies which properties the user would like to 'prettify' and highlight, giving them their own designated card later on in the Individuals.html.*

SEE_PROPERTIES: *Allows for users to point out which properties they'd like to see a list of individuals that share the same information presented (e.g., instead of only seeing who your father is with the property 'hasFather', you can see the list of other identities that share the same father as you).*

GIVE_PRIORITY: *To specify the order of the cards/highlights of the Individual.html page, the user can always just appoint 'NONE' if they don't want any priority assigned and just wants to see things in alphabetic order.*

NOT_SHOW: *Allows users to pinpoint what classes, individuals or even properties they do not want present in the rendered pages.*

A star-based mechanism was introduced as well, to allow users to mark ontology **classes** and **individuals** as favorites. This system is activated through the `STARS` keyword in the DSL configuration, specifically within these parameters.

Besides this keyword, the **VIEW_CLASSES** also has a `TREE` keyword where, if specified, the user sees the taxonomy as a JQuery Tree widget.

B.5 Base SPARQL Queries

A predefined list of SPARQL queries is included in the DSL to extract essential ontology information.

BASE_QUERIES = [""" SELECT ?class WHERE ?class a owl:**Class** . """,
""" SELECT ?property WHERE ?property a owl:**ObjectProperty** . """,
""" SELECT ?property WHERE ?property a owl:**DatatypeProperty** . """,
""" SELECT ?individual WHERE ?individual rdf:type owl:**NamedIndividual** . """]

These queries allow AGAS to fetch ontology classes, object properties, datatype properties, and named individuals, forming the basis for dynamic page generation.

B.6 Metadata Focused Variables

Where users can appoint and fill in necessary metadata for visualization and rendering of the information on the ontology.

AGAS_NAME: *Which corresponds to the name they want to give the 'home' page and to the ontology itself.*
L_DISPOSITION: *Defines the layout orientation (e.g., vertical or horizontal) for list-based content.*
MODULES: *Enables users to specify which properties should be expanded and rendered directly, rather than shown as simple links. This allows greater control over the display of ontology information.*
ABOUT: *Allows users to instruct the file with the information they need and want to showcase in the 'about.html' page.*
ONTOLOGY_SOURCE: *For users to indicate where the ontology came from. And if user-made, can just fill in with the url from the AGAS platform or the prefix of the ontology.*

Example:

MODULES = 'Description': 'DescriptionText + DescriptionInGame', 'EvolvesFrom': 'EvolvesTo', 'PokemonAcquisition': 'AcquiredBy + PokeAcquiredInGame'

This metadata layer empowers AGAS to tailor page content to user-defined visualization needs.

B.7 User Information

Where users can give and fill in their own information, pertinent for support pages, such as the 'about.html' one.

USERNAME: *So users can personalize their username, that later is displayed in the 'about.html' page as well, as pertinent information for the ontology.*
USER_EMAIL: *Where users can write down their email, as to enrich their generated site and ontology information.*
USER_GITHUB: *Allows users to give their github information to, once again, enrich the website.*
USER_SOCIALS: *Which is where the users can indicate the links to their social media, professional accounts. Showcasing information and means for contact that can be pertinent when generating such a website.*

B.8 Page Orientation Option

Users can choose the manner in which to display the NONEXP Home page by filling in the field **AGAS_PAGES** with either:

- ***'BLOG'****: Enables a wiki/blog-like view.*
- ***'PAGES'****: Keeps to the default way of showcasing the information, following the normal pattern of page generation.*

As a bonus, and because background pictures are something that blogs and wikis use a lot, we also allow them to tag along a .png or .jpg image in the field **AGAS_BACKGROUNG**, to make the experience more magical and immersive, but all the while optional, of course.

- ***'NONE'****: If you do not wish to add a background.*
- ***'image/path'****: Which should be to a file in the* ***static/images*** *folder, so it can be used in the template more efficiently, since trying to move the image there automatically via the app.py was not working properly.*