Nooron FAQ

https://nooron.org/know/nooron_faq

An FAQ about Nooron, what it is and how to use it.

1. The faqs

A frequently asked question.

1.1. U00100

1.1.1. Q

What is Nooron?

1.1.2. A

I finally got really annoyed with how hard it is for people to think clearly together. So I decided to [try to] solve the problem once and for all [cue mad-scientist-music] by creating an online digital ecosystem in which knowledge, logic and presentation can all evolve in a globe-spanning, self-organizing, peer to peer system of web servers which is simultaneously the handiest, most flexible piece of software everybody[!] uses and also the software substrate for an emergent global collective intelligence. Take your pick. See How to Build a Global Brain for an expanded form of this vision. [Best keep the mad-scientist-music playing though.] The somewhat more sober Nooron White Paper covers the motivation for Nooron rather well, but is otherwise quite out of date.

1.2. U00200

1.2.1. Q

I don’t give a hoot about all this global brain stuff. I just need a ...

1.2.2. A

OK, OK, lets skip the big picture for the moment. How about if we just look at the handy stuff that Nooron does right now. it comes with some demo ’Nooron Apps’ for publishing your own ’Pattern Language’, ’Web Log’, ’FAQ’ or ’PERT Chart’ it already knows how to present the above information in HTML, DocBook and graphical forms (using Graphviz) anybody who can create HTML manually or has a bit of programming experience should be able create new Nooron Apps It is possible to merge knowledge bases, for example: your personal schedule with your workmates personal schedules (once scheduling is ontologized, that is) in short, Nooron 0.2.x is a system for publishing arbitrary knowledge through diverse multimedia XML templates [’garments’ in Nooron parlance]

1.3. U00205

1.3.1. Q

What other, concrete, features will it have by version 1.0.0?

1.3.2. A

See The Nooron Pattern Language for the whole conceptual tour, but you asked for a concrete answer. A pervasive collaborative filtering system (based on user-contributable criteria and evaluations) will provide the advantages of both peer review and evolutionary pressure on all contents of Nooron: knowledge, logic and presentation. Criteria and evaluations are a simple model which encompasses a broad range of meatspace phenomena, including: opinion polling, voting, petitioning, recommending, peer review and application of selection pressure. It will always be possible for the user to specify whose evaluations to heed and how to summarize and weight multiple evaluations. Always possible for the user to be specific about heeding particular evaluators in particular knowledge-based contexts. Automatically generated data management forms (all knowledge driven, of course) will mean that the creators of new Nooron Apps won’t have to worry about creating add and edit screens, just new ontologies. Of course, they will also be able to create custom add and edit screens if the needs of the app are so specific. Likewise, it might be that all they will prefer to merely create a custom widget for some slot and let it be automatically used by the form generation facilities, and inherited by subclasses. Nooron will be both Fully Automatic) and <a

href="/know/nooron_pattern_language/FullyAdjustable">Fully Adjustable) because it will, throughout, be sensitive to settings which will be guided by worldviews or specific criteria if not explicitly overridden by user preferences. A novel identity and preference system called Not So Basic Authentication will permit users to log onto any Nooron server without ever having to register, just by entering, instead of a userid, the URL of a knowledgebase containing pertinent identity information and preferences. [This is equivalent to issuing a capability to a user's publicly knowable information.] Nooron apps' constituents (ontologies, wardrobes, garments, data) and criteria, worldviews, reviewboards, etc will be able to propagate among Nooron servers in realtime. This can be seen as the automation (or elimination) of the typical software distribution cycle. Rich visualization of numerical slot values as well as people's opinions (their evaluations according to criteria) will be pervasive. Again, all aspects of Nooron are knowledge, so the features which work on Nooron's contents will just as readily work on most of Nooron itself. Example visualizations include: Scattercharts of frames with evaluations driving x, y and node attributes. World views versioning security/authorization system (via capabilities?) Access to 'legacy data' will be provided via PyOKBC backends for: generic PostgreSQL databases, PyOKBC-specific PostgreSQL, generic MySQL databases, remote knowledge-bases, generic python object systems (ala clos-kb in okbc-lisp) and any other PyOKBC (or OKBC) backends the community creates. There will definitely be Nooron Apps for 'Project Management', 'Scenario Planning' and 'Bug Tracking' because Nooron development will itself require them.

1.4. U00210

1.4.1. Q

Nooron is still very alpha, though, isn't it?

1.4.2. A

Oh, so alpha!

1.5. U00211

1.5.1. Q

What are some immediate next steps?

1.5.2. A

The things which will be done to Nooron ASAP are: <ul> slot constraints a web-based knowledge editing interface, initially with just a primitive security model of some sort, ultimately with capability-based security (or is this a pipe-dream!). KBs to be surfable from <code>/know/kb_name/</code> and downloadable from <code>/know/kb_name.pykb</code> Rejigger Nooron Apps so that everything which constitutes the core of an App (ontology, wardrobe and garments) will exist in one directory on disk.

1.6. U00300

1.6.1. Q

What are the 'available garments' listed over at the right?

1.6.2. A

Garments are the different appearances, disguises, what-have-you that the currently displayed frame can 'wear'. What you see at the left is a tree containing links to the different views (garments) which are available for the object (frame) being viewed. They are organized, first

1.7. U00310

1.7.1. Q

How are the 'available garments' implemented?

1.7.2. A

Each of these garments is implemented as a NooronPageTemplate or NPT. They are an adaptation of Zope Page Templates. Each different thing in Nooron (KBs, classes, individuals, slots or facets) has a different set of NPTs (or garments) available to it. Usually these are inherited from classes and superclasses up to :THING (the root of the OKBC class hierarchy). As shipped, all the attaching of NPTs to classes is going on in knowledge bases which are themselves instances of the nooron_app_wardrobe

class. Individuals can also have 'garments' associated directly with them (by making the name of the garment one of the values on the own_slot 'npt_for_self' on the individual in question.)

1.8. U00400

1.8.1. Q

Why call them 'garments' instead of 'skins'?

1.8.2. A

Well, because in Nooron there might also end up being skins. The term 'skin' is usually used to talk about a large-scale cosmetic choice that governs details such as the colors, positions and sizes of onscreen elements. There is room for Nooron to have skins too, but nothing has been done to implement that yet. The template standard_master can be thought of as the only skin currently available for Nooron. It is responsible for placing and formatting all the html screens you see when surfing a stock Nooron instance. A future direction is for programmatic selection of standard_master or some alternative skin to occur at the top of each skin. Probably the thing to do would be to rejigger standard_master so it takes responsibility for implementing the selection of a skin. Nothing stands in the way of current Nooron site operators modifying standard_master in any way they choose, including getting it to swap skins based on knowledge or cookies or something.

1.9. U00500

1.9.1. Q

When I surf around under<code>/know</code> why do some links go to a page over at the Stanford Research Institute?

1.9.2. A

These are 'deep' links into the beautifully constructed OKBC Spec 2.03 at SRI. Since PyOKBC aims to be a faithful implementation of the OKBC Spec, the spec serves as documentation for it.

1.10. U00600

1.10.1. Q

What is a frame?

1.10.2. A

The term ’frame’ comes from the Knowledge Representation tradition in the AI community. The term was coined by Minsky and means a data structure, capable of representing nearly anything, which has slots on it and which exists inside a knowledgebase. If you are going to use Nooron at this stage in its development you will need to understand the OKBC Knowledge Model.

1.11. U00700

1.11.1. Q

What is an ontology?

1.11.2. A

An ontology is a set of formal definitions of concepts and things in some domain. These machine-understandable artifacts can range in scope from narrow to universal. The Yahoo and Open Directory project’s hierarchies are often mentioned as examples of ontologies, but are more properly called taxonomies because they are mostly just hierarchic classifications rather than having enough structural sophistication to provide a basis for inferences about their contents. There are several efforts to create universal ontologies, e.g. http://suo.ieee.org/ and the not-to-be-confused-with-opensource http://www.opencyc.org/

1.12. U00900

1.12.1. Q

What are .pykb files?

1.12.2. A

Hopefully, soon to be replaced. They are the only file format supported by PyOKBC 0.1.1 (the release which was bundled with Nooron 0.2.0). As a shortcut (to avoid having to write a parser) .pykb files are simply files full of python calls to the PyOKBC API. This business about executing .pykb files full of python code has a number of consequences which means that they should soon be replaced by a more conventional file format. Remember, this is Alpha code! A more lasting solution will be to support the reading and writing of files filled with OKBC 'sentences'. This ought to suffice as a basic file format for some time and provide an acceptable format for knowledge 'syndication'.

1.13. U01000

1.13.1. Q

Will a command line version of Nooron be available?

1.13.2. A

A command line interface to Nooron might prove handy, but for most purposes simply fetching an appropriately constructed url from a running Nooron instance will be equivalent to most anything one might want a command line for. This is thanks to the interaction of the two patterns REST and URLsHaveMeaning

1.14. U01100

1.14.1. Q

How do I make my own pattern_language_app?

1.14.2. A

Put a file called <code>SOMENAME_pattern_language.pykb</code> in /know with contents like:<pre>
put_direct_parents(['pattern_language_wardrobe', 'SOMENAME_pattern_language_data'])
put_instance_types(current_kb(),['pattern_language_app'])
put_frame_pretty_name(current_kb(),"SOMENAME Pattern Language") </pre> Put your data in another file called <code>SOMENAME_pattern_language_data.pykb</code> with contents like: <pre>
put_direct_parents(['pattern_language_ontology'])
put_instance_types(current_kb(),['nooron_app_data']) </pre> followed by
<code>create_individual</code> calls that look like those in
<code>/know/nooron_pattern_language_data.pykb</code>

1.15. U01200

1.15.1. Q

How do I render '.dot' files myself?

1.15.2. A

For the moment glean what you can from <a
href="http://www.pinkjuice.com/howto/dot.txt">http://www.pinkjuice.com/howto/dot.txt

1.16. U01300

1.16.1. Q

Is it safe to run Nooron? What with all the python in the templates?

1.16.2. A

Umm. Yes. (but no warranty!) Notice that when Nooron starts up there is the line: <code>Relax: NPT SAFETY is ON</code>. If you see <code>Warning: NPT SAFETY is OFF</code> and your Nooron instance is exposed to the wilds of the internet, then you should set SAFETY = 1 up near the top of <code>code/NooronPageTemplate.py</code>. The SAFETY setting controls whether a restricted

execution environment is used to execute the code within NooronPageTemplates (NPTs). There probably exist a few holes in the Nooron application of the ZPT restrictions. Please report them if you find any.

1.17. U01400

1.17.1. Q

Where is some documentation for Zope Page Templates (the basis of Nooron Page Templates)?

1.17.2. A

FrontPage of ZPT ZPT Error Handling Strategies ZPT Examples (and DTML equivalents)

1.18. U01500

1.18.1. Q

Is there more documentation?

1.18.2. A

Not yet, this is it...