Towards Unity 2.0

Krzysztof Benedyczak

ICM, Warsaw University
Outline

- Unity so far
- Recent developments
- Unity 2.0
- Summary
Development started in 2013

16 releases
- 10 base ones

107k NLOC
- ~41k/38% - Vaadin web UI base and Admin
- ~39k/36% - core Unity engine

Well over 500 tickets
- majority community driven
- most of them was implemented

Already some forks are known and large external contributions are in progress
Unity in the wild

- Growing list of deployments
  - UNICORE in PL-Grid
  - EuDAT (B2Access)
  - UNICORE in HBP
  - EGI platform for long tail of science: access.egi.eu
  - EPOS: TCS AH https://tcs.ah-epos.eu/
  - AAI for CTA (ACK Cyfronet)
  - Cracow synchrotron https://synchro.grid.cyfronet.pl
  - Infona @ ICM
  - ...
- The first fully commercial deployment coming soon
  - US SaaS startup
Recent achievements
Inviting users

- Since a long time Unity supported user registration by offering a registration form to fill:
  - either standalone under a fixed link
  - activated at login (manually or automatically)
- Inviting was a manual process
Inviting users

- Invitations can be defined in Admin UI or via REST API
- Are bound to a concrete registration form
- User gets email with registration link
- Invitation can pre-fill the form
- Invitations has unique codes so invited users can be auto-accepted
Communication with existing users was problematic.
- Asking about acceptance of updated terms and conditions?
- Asking about additional required information attributes?
- or credential?
User enquiries are a new twin brother of registration forms
- intended for already registered users
Share most of the features
But are activated either by clicking a link in enquiry email or after login into one of Unity web interfaces.
### General form settings

<table>
<thead>
<tr>
<th>Displayed name:</th>
<th><img src="image1.png" alt="Image" /></th>
</tr>
</thead>
<tbody>
<tr>
<td>Form information:</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

### Collected/displayed information

| Allow for free text comments | ![Image](image3.png) |

### Agreements

| Attribute: | postalcode |
| Attribute's group: | /infra |

- [ ] Show attribute group in the form
- [ ] Optional parameter

#### Collected attributes

- [ ] OK
- [ ] Cancel
Registration post-processing

- There were many feature requests related to registration handling
  - redirect after registration acceptance or... error
  - remove an attribute if another related was not provided
  - if user is registering from Zoogle then mark email as verified, when from Handbook then as not verified
  - and tons more...
- The original static form post-processing was clearly not enough
  - allowed only to assign some fixed data in addition to what was collected by the form
A similar solution as the one used for translation profiles was added.

A flexible rule engine can be configured to postprocess each request.

Some of the operations are executed immediately upon submission, the most after acceptance.

It is possible to:
- filter, enrich and modify submitted information
- set initial or schedule registered user state changes
- redirect the user after submission, change the confirmation message
- conditionally drop, accept or reject the request

All is integrated with confirmation of emails.
**Condition:** *rattr contains 'staff'*
**Action:** *addToGroup*
**Action parameters:**
**group:** *'/specialOnes'*

---

**Condition:** *validCode*
**Action:** *autoProcess*
**Action parameters:**
**action:** *accept*
Unity's LDAP integration was optimized for use with large LDAP directories.

- Solution for troubles with discovering user groups among 7000

A new mechanism was added to automatically import users

- so far the only implementation is for LDAP, but the subsystem is modular

Reuses the same configuration as LDAP authentication (triggering JiT import)

Can be triggered via REST or... 3rd party query

- This is the UNICORE case when using Unity as an attribute source.
Dynamic attribute statements

- Attribute statements solve problems of:
  - attribute propagation between groups
  - assignment of dynamic attributes as
    - creating attribute from identity
    - a composite attribute
    - modified attribute names/values

- Unity offers now a much more flexible mechanism:
  - Attribute assignment is driven by custom conditions
  - For performance maximum of one extra group can be used for statement evaluation
  - Attribute can be fixed or generated with a dynamic expression.
Unity 2.0
Towards Unity 2.0

- With tons of new features Unity codebase started to get clumsy.
- Couple of development bottlenecks were identified:
  - the persistence module was badly separated from business logic and the most messy part
  - JSON serialization of all Unity artefacts become a must have, but was difficult for some basic classes as Attribute
  - few minor problems caused severe implementation issues
    - for instance enumeration of Spring beans in XML caused too coarse grained use of managed components
  - performance of both typical query and write operations should be improved
    - Unity tends to be too slow on large databases with high significant use.
Unity 2 architecture

- The main focus is on refactoring Unity core parts which are used by the rest of the stack.
Unity 2 refactoring

- The main focus is on refactoring Unity core parts which are used by the rest of the stack.
- Separate persistence API
  - Unity maintains currently 26 different types of objects
  - Maximum reuse of verbs
  - Different implementations must be possible
  - Implement basic DAO, not a composite DAO
- Simple types
  - ubiquities JSON serialization without external dependencies,
  - Proper POJO contract.
Unity 2 refactoring

- Many additional cleanups
  - as a much thinner engine-api module, which previously exposed to much
  - switch to automatic beans discovery
- **Hazelcast storage** as an alternative to pure RDBMS
  - data is loaded at startup from RDBMS and kept in memory
  - all reads are served from memory
  - writes are flushed to RDBMs with a background thread
  - high performance
  - allows for horizontal scaling
  - might be initially considered as experimental
- Unity storage module has now over **1300 unit tests**
Proper design is the best optimization

- Directory schema:
  - 11 groups each with 2 attr stmts
  - 1000 entities
    - each with 11 identities
    - member of all groups
    - 10 attributes in each group

- Tested “operation”, for an entity:
  - get all attributes (>100)
  - get all groups (11)
  - get status & identities (11)

- Average from 3 runs for all 1k entities on the same hardware and MariaDB instance.

- For Unity 2 those are VERY preliminary results
  - not fully tested, zero optimizations
New features for Unity 2

- Completely rewritten automatic contents initialization
  - Should be easy to prepare/edit data to initially populate the database
- Log4j 2.6
- Complete Web Admin UI
- Complete REST API
- Clean and easy to read JSON, same everywhere
Summary

- Unity development is and will be open source
  - we are working on increasing a number of regular contributors
  - feel invited!
- With Unity 2 engine the development and maintenance will be way more rapid.
- Commercial support and SaaS offering is considered in future.