Nothing is lost, nothing is created, everything is `<xml:transformed>`
What is Lavoisier?

An open source framework for developing data aggregation applications.
What are the main issues when developing a data aggregation application?

How does Lavoisier help?

- Common data format: XML
- Common query language: XPath
- Bytes/XML processed in **streaming**
  - detect the smallest data structure to build

**Heterogeneity**
- Protocols
- Data formats

**Performance**
- Too big for memory
- Data source latency

**Quality**
- Reliability
- Robustness
- Monitorability

**Security**
- Authentication
- Authorization
What are the main issues when developing a data aggregation application?

- **Heterogeneity**
  - Protocols
  - Data formats

- **Performance**
  - Too big for memory
  - Data source latency

- **Quality**
  - Reliability
  - Robustness
  - Monitorability

- **Security**
  - Authentication
  - Authorization

**How does Lavoisier help?**

- Common data format: XML
- Common query language: XPath
- Bytes/XML processed in **streaming**
  - detect the smallest data structure to build
- Tune cache according to constraints of
  - data: size, time-to-live, dependencies
  - technology: latency, throughput, availability
  - users: patience, access frequency
- Data validation
- Fallback, cache
- Web console
- Authentication by chaining plugins
- Authorization with XPath expressions

**Protocols**

- Data formats

**Heterogeneity**

- Too big for memory
- Data source latency

**Performance**

- Reliability
- Robustness
- Monitorability

**Quality**

- Authentication
- Authorization

**Security**
What are the main issues when developing a data aggregation application?

| Heterogeneity       | • Protocols  
<table>
<thead>
<tr>
<th></th>
<th>• Data formats</th>
</tr>
</thead>
</table>
| Performance         | • Too big for memory 
|                     | • Data source latency |
| Quality             | • Reliability  
|                     | • Robustness  
|                     | • Monitorability |
| Security            | • Authentication 
|                     | • Authorization |

Lavoisier do all this for you…

…enabling you to focus on business code
Main benefits

- Faster to develop **quality** applications
  - performance, robustness, monitorability, testability, security…

- Easier to maintain: **declarative** programming language
  - a Lavoisier application is made of inter-dependent data views
  - each data view is a chain of plugins and templates
Main benefits

- Faster to develop **quality** applications
  - performance, robustness, monitorability, testability, security…

- Easier to maintain: **declarative** programming language
  - a Lavoisier application is made of inter-dependent data views
  - each data view is a chain of plugins and templates

- Factorize development efforts: **about 100 plugins** for
  - Technologies
    - HTTP, SQL, SSH, grid (JSAGA)…
  - Formats
    - JSON, YAML, CSV, LDIF, text…
  - Cache mechanism
    - on disk/in memory, indexed, BaseX…
  - Security
    - CAS, password, X509, IP, OAuth…

- Separate actors responsibilities (*see next slide*)
Main benefits: separate actors responsibilities

- heterogeneous data sources
  - flat file
  - RDBMS
  - WS

Lavoisier framework

- Business Layer (templates)
- Web console
- Presentation Layer
- REST API
- config

- Service administrators
- Users
- HTML, PDF...
- XML, JSON, CSV...

- Plugin developers
- Application developers

Presentation Layer

HTML
Projects using Lavoisier

**EGI projects**

- **ARGO**
  - availabilities, reliabilities

- **VAPOR**
  - toolkit for VO management

**CC-IN2P3 projects**

- Data import for **CMDB**
  - (see Emmanouil's talk)

- **CC-Status**
  - A. Vedaee, O. Lequeux

- Cache for **Grid Engine**
  - J-R. Rouet

- **CostModel**
  - R. Vernet

- Automatic generation of CE static information for site **BDII**
  - C. Eloto

- Part of data import (iRods) in **decision-making tool**
  - C. Evesque

(almost) all these use-cases have heterogeneous data sources
GUI for assisting development of Lavoisier applications