● **Operations Portal ~ 2004**
  ○ **Data retrieved from services developed in various countries**
    > Technologies were heterogeneous
    > Technologies were potentially replaced with newer ones
  ○ **Data retrieved from many distributed sources**
    > Memory overload issues
    > Data source availability and latency issues

● **Many other applications need to aggregate data ~ Now**
  ○ Multiplicity of technologies, protocols, formats
  ○ We need to factorize the development efforts for supporting:
    > New technologies
    > Common features (robustness, monitoring, security)
What does Lavoisier?
What does Lavoisier?

Heterogeneous
• protocols
• data formats
Heterogeneous
- protocols
- data formats

Standard
- data format (XML)
- query language (XPath)
- query API (REST)
What does Lavoisier?

Heterogeneous
- protocols
- data formats

Standard
- data format (XML)
- query language (XPath)
- query API (REST)

Rendering format chosen by user
Lavoisier architecture
Lavoisier architecture

plug-in
A chain of plug-ins

**EXAMPLE**
- Retrieve a distant CSV File
- Extract columns
- Provides a html output
EXAMPLE

- Retrieve a distant CSV File
- Extract columns
- Provides a html output.

A chain of plug-ins

Retrieve a distant CSV File

HTTP Connector

CSV Serializer
A chain of plug-ins

EXAMPLE
- Retrieve a distant CSV File
- Extract columns
- Provides a html output

- Retrieve a distant CSV File
- CSV Serializer
- XML Template
- Extract columns
A chain of plug-ins

EXAMPLE
- Retrieve a distant CSV File
- Extract columns
- Provides a html output

- HTTP Connector
- CSV Serializer
- XML Template
- HTML Renderer
Into the “views” file

```xml
<views xmlns="http://software.in2p3.fr/lavoisier/application.xsd">
  <view name="my_example_view">
    ...
  </view>
</views>
```
<views xmlns="http://software.in2p3.fr/lavoisier/application.xsd">
  <view name="my_example_view">
    <connector type="HTTPConnector">
      <parameter name="url">http://software.in2p3.fr/lavoisier/input.csv</parameter>
    </connector>
  </view>
</views>
<views xmlns="http://software.in2p3.fr/lavoisier/application.xsd">
  <view name="my_example_view">
    <connector type="HTTPConnector">
      <parameter name="url">http://software.in2p3.fr/lavoisier/input.csv</parameter>
    </connector>

    <serializer type="CSVSerializer">
      <parameter name="header">true</parameter>
    </serializer>
  </view>
</views>
<views xmlns="http://software.in2p3.fr/lavoisier/application.xsd">
  <view name="my_example_view">
    <connector type="HTTPConnector">
      <parameter name="url">http://software.in2p3.fr/lavoisier/input.csv</parameter>
    </connector>

    <serializer type="CSVSerializer">
      <parameter name="header">true</parameter>
    </serializer>

    <processors> [XML TEMPLATES ] </processors>
  </view>
</views>
<views xmlns="http://software.in2p3.fr/lavoisier/application.xsd">
  <view name="my_example_view">
    <connector type="HTTPConnector">
      <parameter name="url">http://software.in2p3.fr/lavoisier/input.csv</parameter>
    </connector>
    <serializer type="CSVSerializer">
      <parameter name="header">true</parameter>
    </serializer>
    <processors> [XML TEMPLATES ]</processors>
  </view>
</views>

http://myserver:8080/lavoisier/my_example_view?accept=html
Structure of a view

```
&view name="my_structure">

<!-- the connector is adapted to the technology, protocol or format of the data source -->
<connector type="""> [...] </connector>

<!-- optional if the connector output is already in xml format →
<!-- in case select the serializer corresponding to the connector output format -->
<serializer type="""> [...] </serializer>
```

</view>
Structure of a view

```xml
<view name="my_structure">
  <!-- the connector is adapted to the technology, protocol or format of the data source -->
  <connector type=""> [...] </connector>

  <!-- optional if the connector output is already in xml format →
  in case select the serializer corresponding to the connector output format -->
  <serializer type=""> [...] </serializer>

  <!-- validates the structure or the content of the output →
  <validator></validator>
</view>
```
<view name="my_structure">

<!-- the connector is adapted to the technology, protocol or format of the data source -->
<connector type=""> [...] </connector>

<!-- optional if the connector output is already in xml format → -->
<!--  in case select the serializer corresponding to the connector output format → -->
<serializer type=""> [...] </serializer>

<!-- validates the structure or the content of the output → -->
<validator></validator>

<!-- describe the rules to transform the xml structure-->-->
<processors> [ XML Templates ] </processors>

</view>
<view name="my_structure">

  <!-- the connector is adapted to the technology, protocol or format of the data source -->
  <connector type="" [...] </connector>

  <!-- optional if the connector output is already in xml format →
  in case select the serializer corresponding to the connector output format -->
  <serializer type="" [...] </serializer>

  <!-- validates the structure or the content of the output →
  <validator></validator>

  <!-- describe the rules to transform the xml structure→
  <processors> [ XML Templates ] </processors>

  <!-- describe how to cache data and the associated refresh triggering rules →
  <cache>
    <trigger type=""""</trigger>
  </cache>

</view>
<view name="my_structure">

<!-- the connector is adapted to the technology, protocol or format of the data source -->
<connector type="""> [...] </connector>

<!-- optional if the connector output is already in xml format →
<!-- in case select the serializer corresponding to the connector output format -->
<serializer type="""> [...] </serializer>

<!-- validates the structure or the content of the output →
<validator></validator>

<!-- describe the rules to transform the xml structure-->
<processors> [ XML Templates ] </processors>

<!-- describe how to cache data and the associated refresh triggering rules -->
<cache>
  <trigger type="""></trigger>
</cache>

<!-- organise data around 2D formats : table, chart -->
<pre-renderers> [...] </pre-renderers>

<!-- overwrite default values for the renderers, including html templates
<renderers>
  <renderer type="""> [...] </renderer>
</renderers>

</view>
Structure of a view

```xml
<view name="my_structure">
    <!-- the connector is adapted to the technology, protocol or format of the data source -->
    <connector type=""> [...] </connector>

    <!-- optional if the connector output is already in xml format →
    <!-- in case select the serializer corresponding to the connector output format -->
    <serializer type=""> [...] </serializer>

    <!-- validates the structure or the content of the output →
    <validator/>

    <!-- describe the rules to transform the xml structure→
    <processors> [ XML Templates ] </processors>

    <!-- describe how to cache data and the associated refresh triggering rules →
    <cache>
        <trigger type=""/></trigger>
    </cache>

    <!-- organise data around 2D formats : table, chart →
    <pre-renderers> [...] <pre-renderers>

    <!-- overwrite default values for the renderers, including html templates
    <renderers>
        <renderer type=""> [...] </renderer>
    </renderers>
</view>
```
PERFORMANCE

- Low memory usage thanks to on-the-fly processing of data streams (rather than in-memory data structures).
- Each data stream can be cached.
**Benefits**

**PERFORMANCE**
- Low memory usage thanks to on-the-fly processing of **data streams** (rather than in-memory data structures).
- Each data stream can be **cached**.

**ROBUSTNESS**
- **Validation** of the data
- Tolerance to disruptions of the data sources (**cache**)
- Management of errors: fallback
○ Easy to add/replace plug-ins in case of data sources change
○ Numerous plug-ins adapted to diverse technologies and protocols
○ > 100 reusable plug-ins: HTTP, LDAP, RDBMS, XSLT ... 
... extendable with your own plug-in
Easy to add/replace plug-ins in case of data sources change
Numerous plug-ins adapted to diverse technologies and protocols
> 100 reusable plug-ins: HTTP, LDAP, RDBMS, XSLT...
... extendable with your own plug-in

Numerous plug-ins to improve performances
  Choice of cache type and refresh rules
Easy to add/replace plug-ins in case of data sources change
Numerous plug-ins adapted to diverse technologies and protocols
> 100 reusable plug-ins: HTTP, LDAP, RDBMS, XSLT ...
... extendable with your own plug-in

Numerous plug-ins to improve performances
  - Choice of cache type and refresh rules

Multiple choices of rendering: CSV, XML, JSON, HTML, chart ...
... through the REST API
DATA PROCESSING – XML Templates

- Set of rules which describe how to transform the data / the structure of the data
- Remove / replace / add elements or attributes
DATA PROCESSING – XML Templates
- Set of rules which describe how to transform the data / the structure of the data
- Remove / replace / add elements or attributes

SECURITY
- Authentication and authorization plug-ins
- X509, pwd (white List, Kerberos), OAuth2, Cas, Ip
Integrated Tools

Lavoisier Data Processing Service
LavoisierEditorPlugin
1 plug-in for IDEA IDE

Lavoisier Data Processing Service

Integrated Tools
LavoisierEditorPlugin
1 plug-in for IDEA IDE

LavoisierConsole
1 Web console for administration and monitoring

Integrated Tools

Lavoisier Data Processing Service
LavoisierEditorPlugin
1 plug-in for IDEA IDE

LavoisierConsole
1 Web console for administration and monitoring

LavoisierDashboard
1 graphically customizable dashboard for users

Lavoisier Data Processing Service
Automatically deploy one Lavoisier application

... on a puppetized machine@CC

... in Docker / Kubernetes
~ 15 data sources
- HTTP, SQL, LDAP
- JSON, CSV, XML, HTML

~ 12 data sources
- HTTP, SQL, LDAP, Telnet, OpenStack, Shell
- JSON, CSV, XML, HTML

~ 5 data sources
- HTTP, SQL, LDAP, Shell
- JSON, XML, HTML
- 20 ldap queries ~ average output per query 200 MB
• **Facilitate / accelerate developments and maintenance**
  ○ Normalized configuration of plug-ins
  ○ Normalized languages, homogeneous process
  ○ Numerous reusable plug-ins
  ○ View modularity
  ○ Ecosystem : IDE integration, deployment tools
  ○ Automatic features : monitoring, security
  ○ Roles separation: Lavoisier developer, application developer, administrator, user

• **Robustness / Performances**
  ○ Caches
  ○ Streaming process
An original idea from **Sylvain Reynaud** …

**Director**: Sylvain Reynaud

**Assistant Director / Press Officer**: Cyril L’Orphelin

**Technical Staff**: Bernard Chambon, Cyril Flieller, Olivier Lequeux, Jonathan Moutarde, Lionel Schwarz, Alexandre Zicaro

**Set Decorator**: Philippe Correia

**Distributor**: Justin Bussery, Christelle Eloto, Rémi Ferrand, Benjamin Guillon
Questions?

lavoisier@cc.in2p3.fr

http://software.in2p3.fr/lavoisier