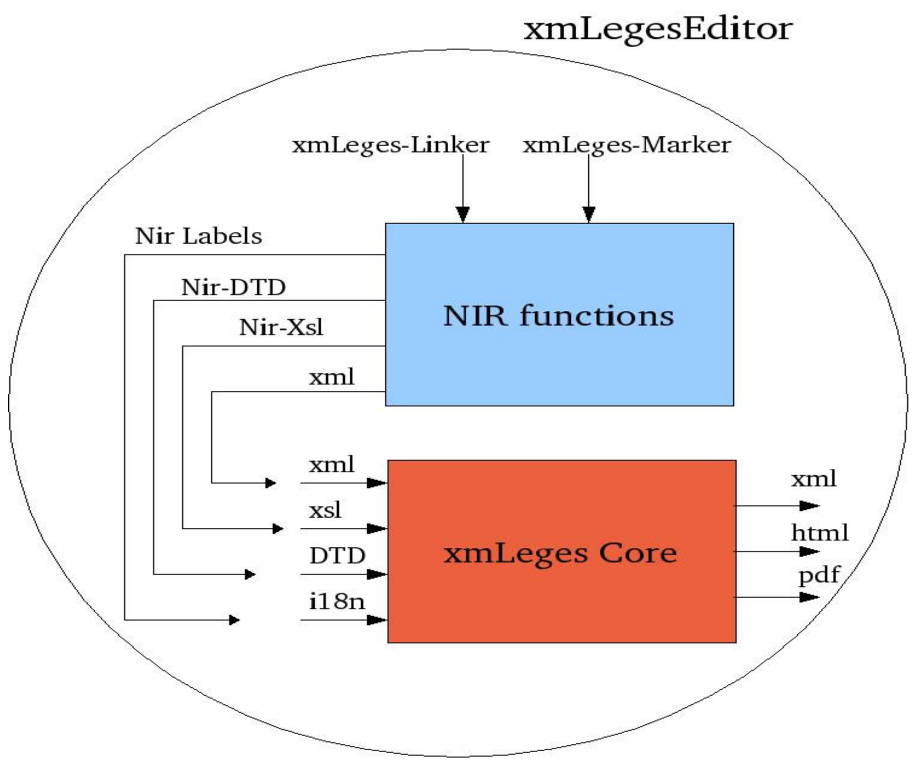
xmLegesEditor – Architecture

http://www.xmleges.org

- xmLegesEditor is a specific editor for legislative documents based on NIR (NormeInRete) Italian Legislative standard (DTD/Schema and URN);
- Entirely written in Java using opensource standard libraries (Apache)
- Native XML editor (directly manages XML documents compliant to NIR format)
- Visual XML Editor with word-processor lookandfeel (hidden XML syntax)
- Latest version supports NIR-DTD 2.2
- Entirely customizable to support different DTDs or XML-Schema
- Licensed in GPL

The Architecture

xmLegesEditor has been developed on the experience of prototype NirEditor. In this new version a complete reengineering of the software has been made. The aim was to guarantee maximum extensibility, modularity and reusability of the different software modules. To this end xmLegesEditor has been structured on a component-based architecture where each component provides specific services. Moreover a strict separation between generic XML modules and specific NIR modules has been followed. This, as a by-product, gave the significant result that the developed components can be reused in order to develop other specific visual editors for supporting other XML standards and in particular Legislative XML standards. The complete editor application is in fact obtained by composing the different services specified in an application composition XML file. Replacing NIR modules with other specific modules developed for supporting such different standards, a new editor can be obtained using the core XML editor infrastructure. In the following figure such architecture is depicted.



xmLegesCore: a generic and application independent Visual XML editor

Parametrized by:

- XML – the document
- XSL – visualization stylesheets, internal for the editing environment, external for the export
- DTD/XSD – The format to be supported, used for document validation and Contextual Rule Querying
- I18n –Internazionalization of the UI labels and images (menus, toolbars, icons)

Output :

- XML - Valid edited document
- HTML/PDF - Document export for publication, print etc..

General Functionalities*:*

- Undo/redo
- Drag & drop (partially implemented)
- Cut Copy & Paste
- Find & Replace

- Comments/Processing Instructions management
- Document Tree View
- Attributes View
- Document Outline (as XSL visualization)
- Single Document support
- Multipanel (synchronized) view
- Open/close/Validate/Validation Errors log
- Save, Save As
- Image / Pictures insertion
- Table support (only html tables tags or depending on the format as noncore functionality)
- Export (HTML,PDF,rtf) xsl,xsl-fo,css support
- Spellcheck supports MySpell Openoffice OpenSource dictionaries

  http://lingucomponent.openoffice.org/
- Editable Xsl filtered visualization of the document

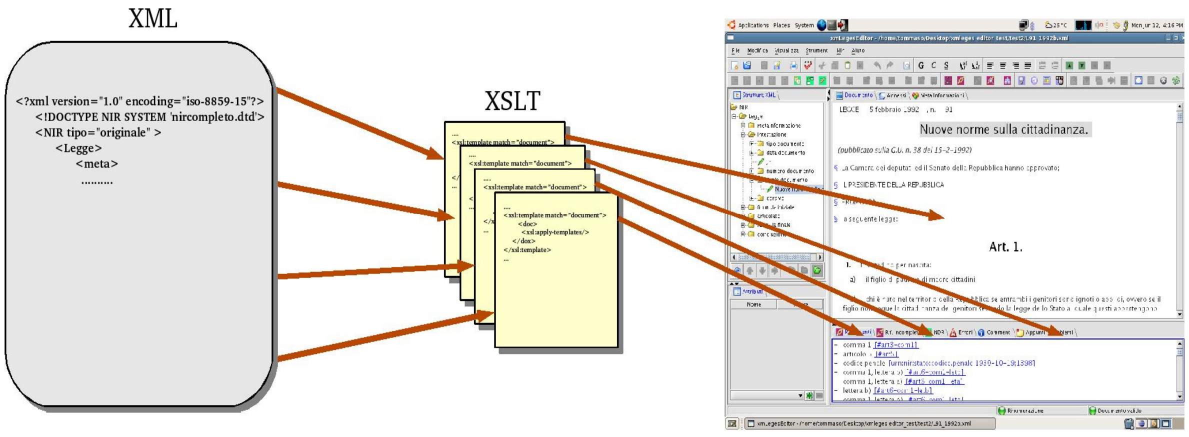
CORE Features:

xmLegesCore provides functionality for composing a basic generic visual XML editor. Its parameters are: the DTD or XSD files of the standard to be supported, the XSLT stylesheet files for internal visualization inside the editor and for document exportation in output formats as HTML or Pdf, one or more files containing textual labels and icons for the complete customization and internazionalization of the user interface. All this aspects have been kept as application's parameters rather than cabled inside the code in order to guarantee maximum reusability. xmLegesCore provides complete functionalities for:

Validity Management: This is accomplished through the access to the rulesManager service. A distinguishing feature of xmLeges is that only valid operations are allowed in the XML document editing: this is obtained by contextually querying the ***rulesManager*** component which provides methods to access the DTD or XSD parsed in a Finite State Automata graph structure in order to a priori validate in such structure the effects of a certain operation without actually operate over the document. Only valid documents can be produced (*"a priori validation"*). Changing the grammar automatically affects rulesManager answers. This allows to provide such Validity Management strategy to any document standard expressed both in DTD or XSD.

Document Management: A dedicated component (DocumentManager) provides services for accessing a generic XML Document. The document is initially parsed into a Document Object Model format and then made available in such format for access and modification to any other component. Functions for opening, saving, management of a multi-level undo/redo are made available by the *DocumentManager* component.

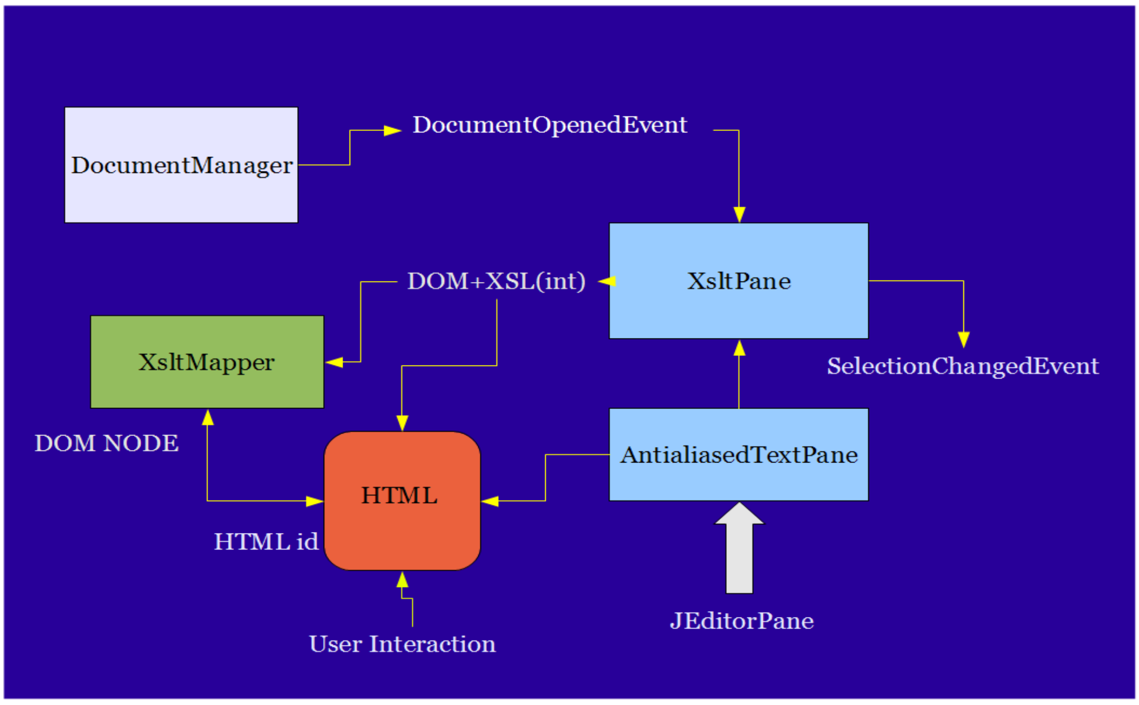
Document Visualization: *WYSIWYG – XSL filtered visualization*. The strategy in document visualization consists in the visualization in the various different panels of an HTML view obtained by applying an XSLT stylesheet to the XML Document. The number and the content of visualization panels is configurable.



```
<?xml version="1.0" encoding="iso-8859-15"?>
    <!DOCTYPE NIR SYSTEM 'nircompleto.dtd'>
    <NIR tipo="originale" >
        <Legge>
            <meta>
             ..........
```

```
....
<xsl:template match="document">
    <doc>
      <xsl:apply-templates/>
    </dox>
</xsl:template>
...
```

In order to provide editing functionalities from such views, a mapping between the original XML document elements and the visualized HTML text is provided in order to make this process bidirectional and store the input from the editor panels into the corresponding XML elements. This is accomplished by two different components *XsltMapper* and *XsltPane* which are at the heart of xmLegesCore visualization functionality.



Internazionalization: All the labels and the icons appearing in the editor User Interface are actually loaded from localization files accessed through a unique identifier. This means that by simply changing some locale file the whole UI can be translated and customized with different icons without actually changing the code. This service is provided by the *i18n* component.

xmLegesEditor can be seen as the overlapping of the core layer with a Format specific layer implementing functionalities for its full support. In the case of NormeInRete:

xmLegesNIR At the top level xmLegesNIR provides parameters to the Core level in order to specialize it to the NIR standards. Moreover, a number of NIR specific components have been developed to provide enhanced functionalities to the support of NIR-Elements with customized visualization and specific menus, toolbars, forms. At this level external modules integration is also provided.

Application Architecture – some Technical Detail:

- *xmLeges* is developed on a *component* based architecture where each component provides a specific service. Implementation of the IOC Pattern "Inversion of Control" (Direct Injection) http://www.martinfowler.com/articles/injection.html
- ServiceManager inspired to *Apache Avalon framework* implemented in the package it.cnr.ittig.services.manager , see http://excalibur.apache.org/framework/index.html for documentation on the Apache Framework.
- ServiceManager implements a restricted set of interfaces, keeps the Framework structure.
- The application is "*composed*" on startup by instantiating a set of components as specified in the application composition file *xmLegesEditor.xml* read by the ServiceManager (Plug-in like architecture).

  Main class: it.cnr.ittig.services.manager.Run

  Arguments: xmlegesEditor.xml

Component LifeCycle:

- A component LifeCycle specifies the methods that can be invoked over it and their order
- Each component exposes its "*lifecycle methods*" implementing the "*lifecycle interfaces*". The interfaces defined in it.cnr.ittig.services.manager.Run are:
  - **Serviceable** (the component can request other services)
  - **Loggable** (a Logger is provided to the component)
  - **Configurable** (The component can be configured)
  - **Initializable** (The component can execute some code in the initialization phase)

- **Startable** (The component can execute some code in the *start* phase – after initialization – or in the *stop* phase – before dispose)
- **Disposable** (The component can execute some code in the *dispose* phase – application shutdown or crash)

Component LifeStyle

- **ACTIVATION**: “*startup*” or “*lazy*” [DEFAULT: “lazy”]
  - **startup:** instantiation of the component on application startup.
  - **lazy:** instantiation of the component upon request of another component.
- **LIFESTYLE:** *“singleton”* or “pool” ” [DEFAULT: “singleton”]
  - **singleton:** a unique component instance is allowed and shared among the components.
  - **pool:** upon each service request a new component instance is created.
- Example: from xmLegesEditor.xml:

```
<component  activation="startup" lifestyle="singleton"
class="it.cnr.ittig.xmleges.core.blocks.lookandfeel.LookAndFeelImpl" >
    <configuration>
        <lookandfeel>
         com.jgoodies.plaf.plastic.Plastic3DLookAndFeel
      </lookandfeel>
    </configuration>
 </component>
```

Packages Organization:

```
xmLegesCore
        xmLegesCoreApi
        xmLegesCoreImpl
xmLegesEditor
        xmLegesEditorApi
        xmLegesEditorImpl
```

- xmLegesCore: Interfaces and Implementations of a Generic Visual XML Editor

```
it.cnr.ittig.xmleges.core.services.*    (API)
it.cnr.ittig.xmleges.core.blocks.*      (IMPL)
```

- xmLegesEditor: Interfaces and Implementations of the domain specific layer

```
it.cnr.ittig.xmleges.editor.services.*    (API)
it.cnr.ittig.xmleges.editor.blocks.*      (IMPL)
```

xmLegesCore main components:

it.cnr.ittig.xmleges.core.services.

- **EventManager**: service for *events* exchange management

- **ActionManager**: service for *actions* mangement and registration

- **Bars**: service for management and configuration of *menus, toolbars*, *status bar* (configurable, see below)

```
<configuration>
        <menus>
                <menu action="menu.file">
                        <item action="file.new" />
                        ....
                </menu>
        </menus>
        <tools>
                <toolbar name="file" floatable="false" rollover="true">
                        <item action="file.new" />
                        .......
                </toolbar>
        </tools>
<configuration>
```

- **DocumentManager**: service for XML document management: opening, parsing, validation, saving, editing transactions, undo/redo

- **SelectionManager**: service for the management of DOM nodes selection and text selection. ActiveNode management.

- **I18n:** service for labels, icons, menus, forms internazionalization management.

- **RulesManager:** service for Rules files (DTD, XSD) parsing, conversion into finite state automata, contextual querying, valid default content models management.

- **Frame:**   service for the mangement of the application visual frame. Each editing panel to be displayed should be registered on the Frame.   (CONFIGURABLE – panels location in the Frame)

```
<configuration>
        <title>xmLegesEditor</title>
                <!-- where=top-center | top-left | bottom-center | bottom-left -->
                        <!-- index=number -->
```

- **XsltMapper**: service implementing the bidirectional mapping between DOM document Nodes and HTML displayed Nodes for displaying and editing on XsltPane panels.

- **XsltPane**:   service for the management of a single textual panel for the visualization and editing of an XML document.The panel displays a styled text in HTML format by applying an XSLT+css stylesheet. The (valid as contrained by RulesManager) editing on the panel is seamlessly included in the XML-DOM of the document.

LIFESTYLE: *pool* - many different instances of XsltPanels displayed inside the application frame providing multiple views on the document.

CONFIGURABLE – can associate XSL and CSS to the panel .

- **AttributesPane:** service for the displaying and editing of a DOM-Node Attributes.
- **TreePane:** service for the DOM-Document Tree visualization and Navigation.
- **Form:** service for the creation and management of a dialog window with standardized lookandfeel.

Main events exchanged throuh EventMananger

ActionRegisteredEvent
DocumentChangedEvent
DocumentClosedEvent
DocumentOpenedEvent
DocumentSavedEvent
SelectionChangedEvent
PaneActivatedEvent
PaneDeactivatedEvent
PaneFocusGainedEvent
PaneFocusLostEvent
PaneStatusChangedEvent
SelectionChangedEvent
PaneActivatedEvent
ExecFinishedEvent
ExecStartedEvent

UI design

Forms designed using Abeille Forms Designer

"*Abeille Forms Designer is a tool for quickly designing cross-platform graphical user interfaces. Abeille is a Swing based application and uses Swing components. Abeille depends on the* **JGoodies FormLayout Manager**"

OpenSource Tool: see https://abeille.dev.java.net/

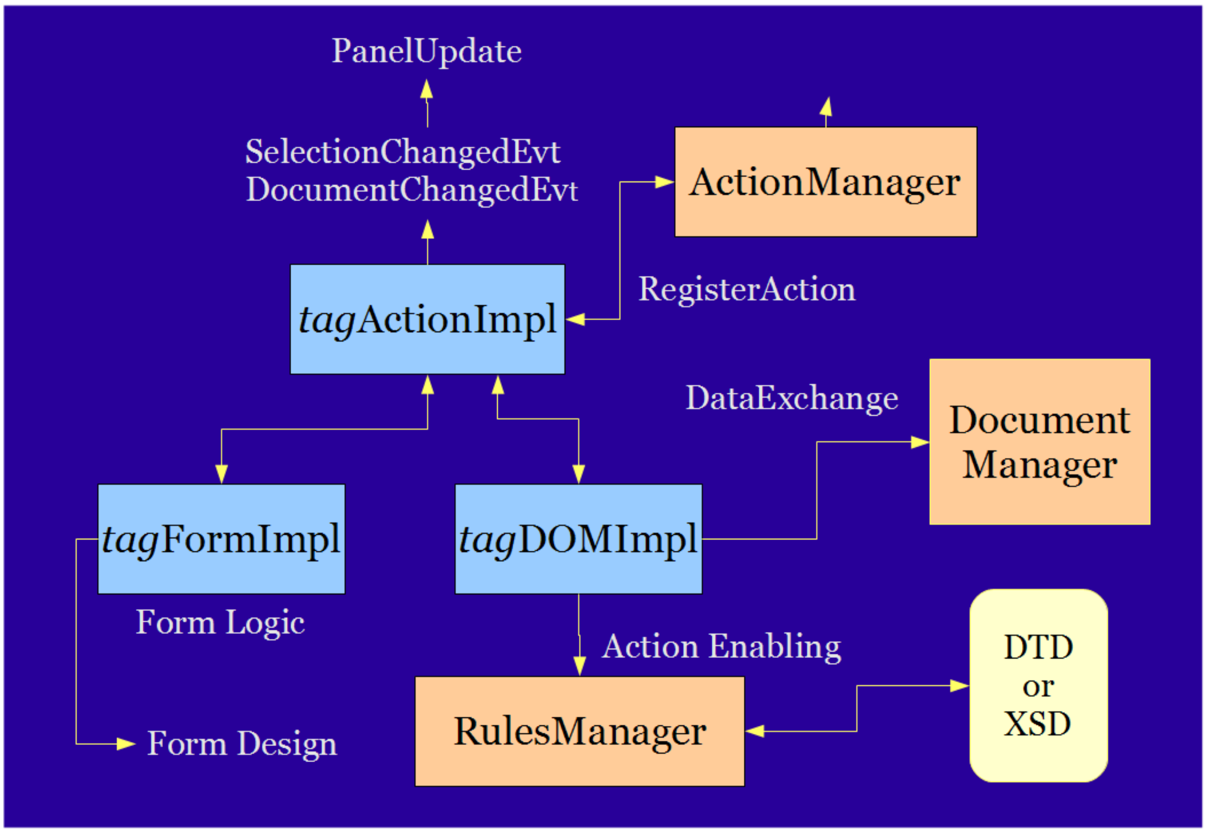
"a priori validation"

Each action in the menus and toolbars is enabled conditioned to the RulesManager query response in order to check if the requested operation will keep the document validity according to the loaded grammar. Otherwise the buttons are disabeld.

In the DOM functions each operation on the DOM tree (insert, append, delete) of the document is executed provided that it is declared admissible by querying the RulesManager.

RulesManager provides the minimal complete valid template for each tag (Element and its mandatory children and attributes)

Starting from a valid preexisting document or minimal valid document template it is impossible to produce an invalid document within the xmLeges editing environment *(some constraints can be relaxed if needed)*. XML validation functionality is transparent to the user.



xmLeges Extensibility:

Extending *xmLegesCore* to support different Legislative standards consists in:

- **Reusing** xmLegesCore functionalities "*as is* " allowing for basic valid document editing and contextul element and attribute insertion and deletion from right-click.

- **Developing** the format specific layer with:
  - Internal XSL visualization stylesheet to customize the WYSIWYG editing panels.
  - External XSL export stylesheet for HTML and PDF publication.
  - Developing the specific DOM components and related Forms, Actions, Menus and Toolbars according to the schema depicted above invoking the available core services previoulsy described.
  - Editing the i18n properties files for translation of the interface and customization of the icons in the chosen language.

Declare the new configurations and developed components in a new application composition file replacing xmLegesEditor.xml

Tommaso Agnoloni - ITTIG-CNR

agnoloni@ittig.cnr.it