« Logical Layer »

The Logic part

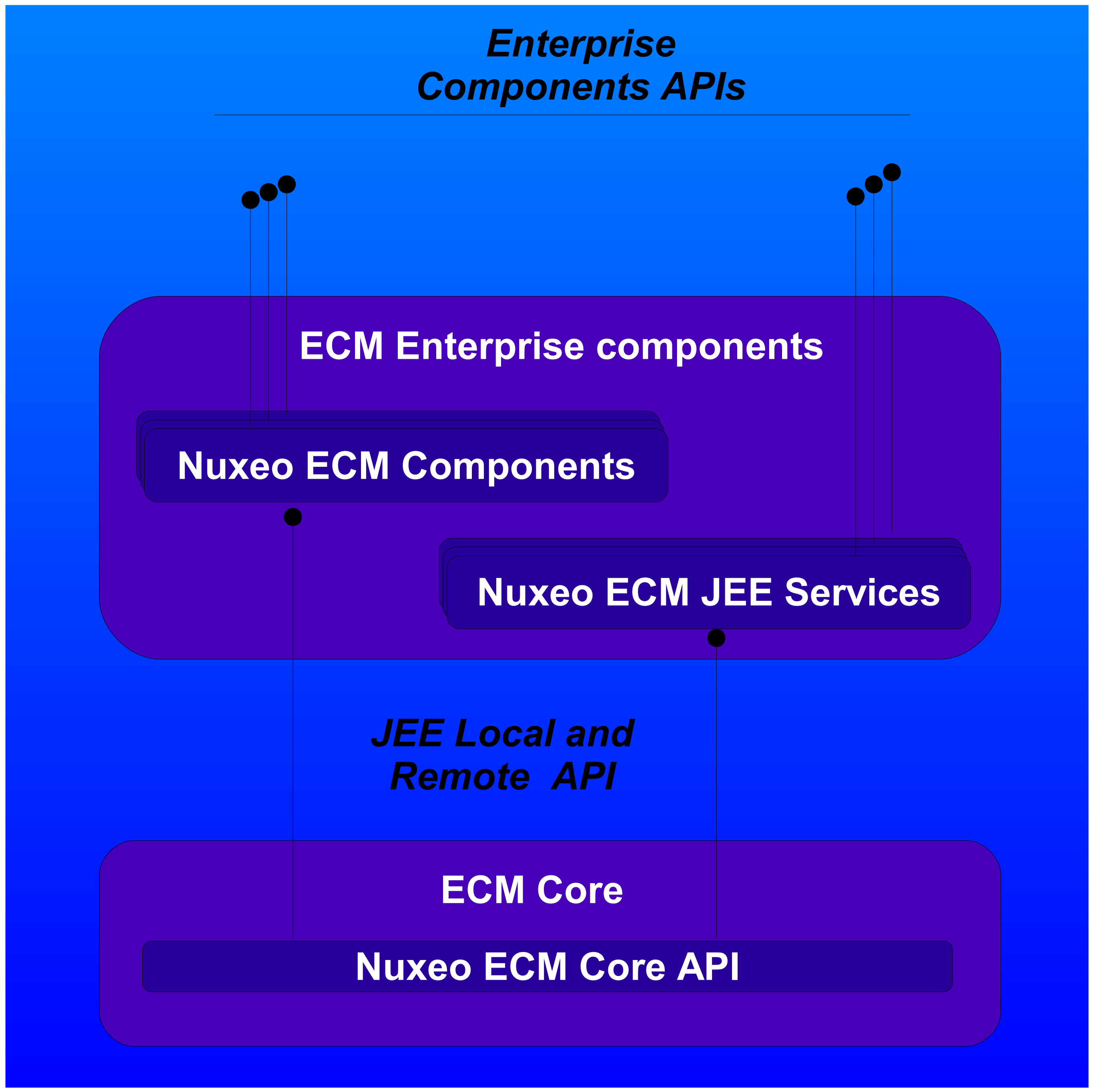
- High level ECM services
  - Audit and Log services
  - Workflow service
  - Comment service
  - Transformation service
  - etc.

- Provides business logic
  - Nuxeo Core provides the infrastructure
  - Enterprise Components provide behaviour and features related to content / documents

Architecture Diagram

- Packaged as several separate bundles

- Leverages JEE 5
  - EJB3 SB
  - EJB3 MDB
  - JAAS integration



Nuxeo Service – Typical impl.

- A Nuxeo Runtime Service
  - Leverage NXRuntime extensibility
    - Extension points
    - XML Configuration
  - Leverage NXRuntime platform independance
    - Runs on JBoss
    - Runs on Eclipse RCP
- A EJB3 Facade on top of Nuxeo Service
  - Provides Security and Transaction integration
  - Leverage EJB3 state management
  - Provides remoting support
- Clean & clear API

Nuxeo Services - SPI

- A lot of Nuxeo Services provide SPI interfaces
  - Leverage Runtime to let several providers implement the same features (using extension point / extension contribution)
- Some examples
  - Content Transformation service
  - Workflow engine
  - Relation service
- Most Nuxeo Services leverage NXRuntime for XML configuration and extensibility

Nuxeo 5 and EJB3

- We use the new (JEE5) EJB model: EJB 3
  - Simpler code (one simple class)
  - Annotations instead of XML descriptors
  - Almost POJO! :-)

- Nuxeo 5 uses
  - Session Beans for Services
    - Transaction, Security, Pooling, State management, Remote access
  - Message Driven Beans
    - JMS driven services
  - Entity Beans
    - When RDBMS storage is usefull/efficient (audit log, non-content data, etc.)

Nuxeo Services

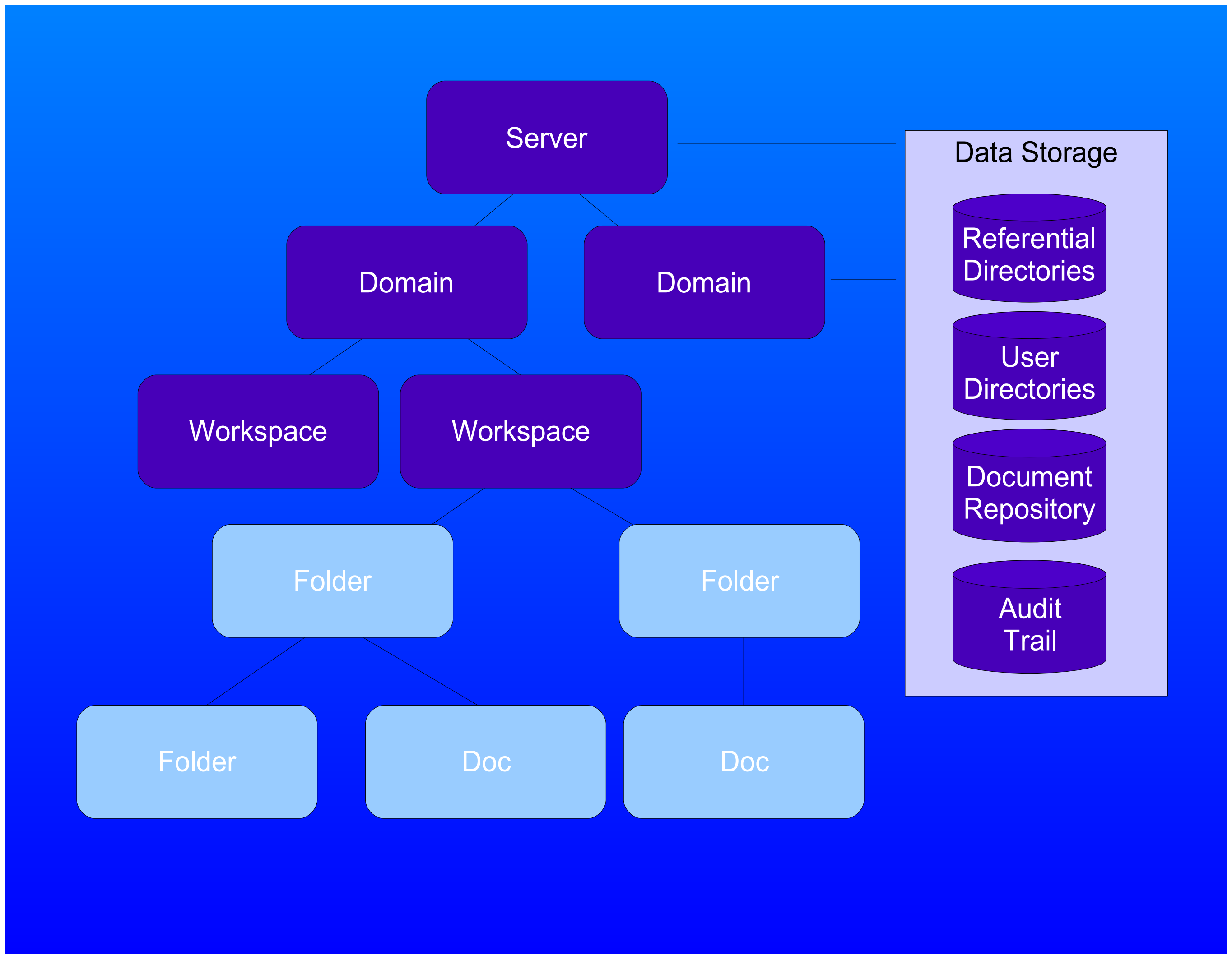
- NXWorkflow
  - Provides a WfMC-oriented API
  - Integrates BPM and Document Management
  - Provides a SPI to support several engines
    - Default engine: JBoss jBPM
- NXRelations
  - Provides RDF-based relations management between ressources/documents
  - Provides an SPI (default enfine: Jena RDF)
- NXEvent
  - JMS Messaging (default implementation: JBoss Messaging)
    - Relay Core Events
    - Provide an API for messaging

Nuxeo Services

- NXDirectory
  - (Meta-) directory manager (CPSDirectory-inspired)
  - SQL and LDAP support
  - Configuration via XML/EP
  - Usefull to manage any non-content data list (keyword, list of values, for forms, users, groups, units, etc.)
- NXUserManager
  - Provides a UserManager (like CPSUserFolder)
  - Uses NXDirectory directories as user/group datasources
- NXTransforms
  - Content Transformation service
  - Transformers are plugins (extension contributions)

Nuxeo Services

- NXMimeTypes
  - Pluggable Mime Type detection service
- NXPlatform
  - Default content hierarchy for web application
    - Provide very basic content logic over the core
    - Enable multi-cores instances



Nuxeo Services

- NXFileManager
  - dedicated to « file to document » mapping
    - file (mime-type, location, ...) => document type
  - exposes File management API (upload, download, etc.)
    - for Microsoft Office Helper
    - for Browser Plugins
- NXAudit
  - Customizable audit log
  - Based on NXEvents
- NXJobs
  - Job running service (cron-like)

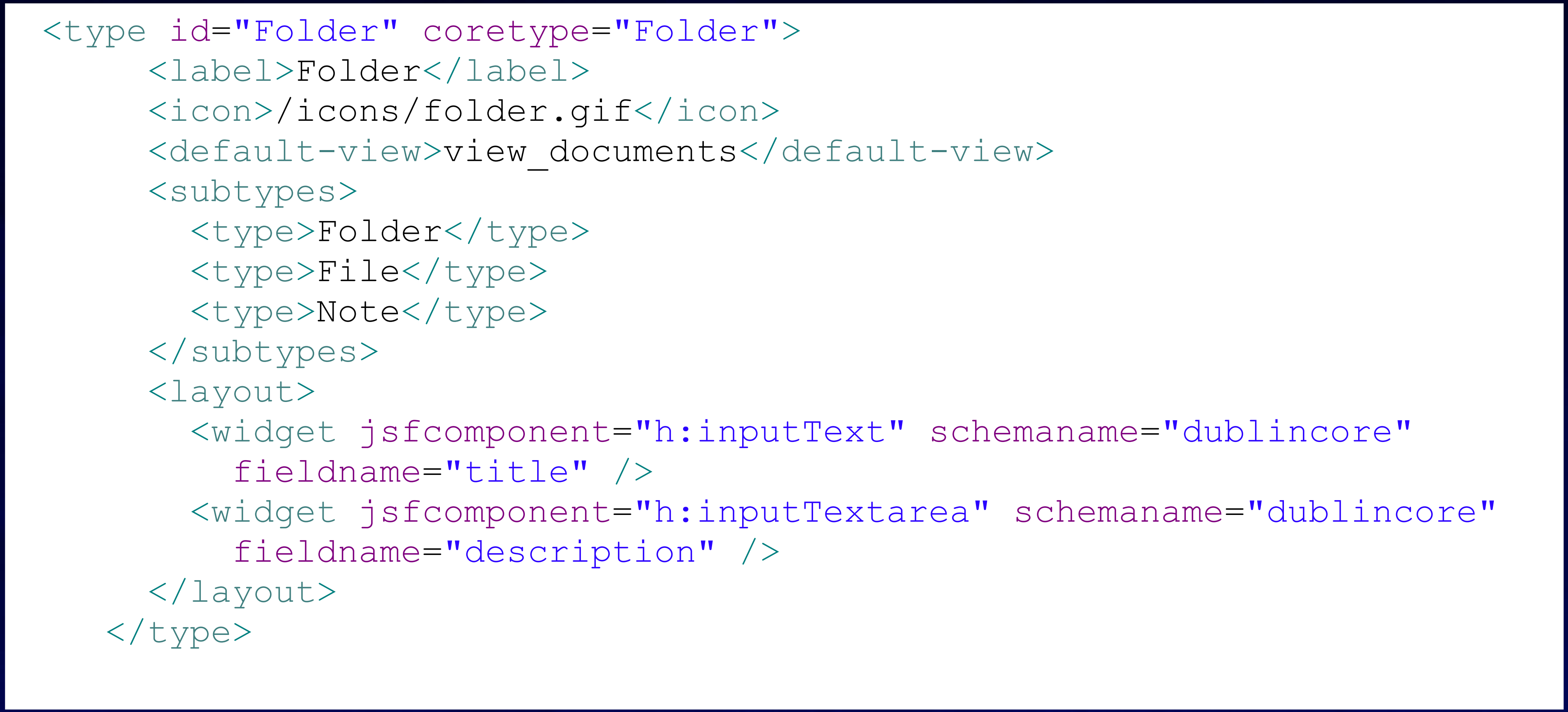
Nuxeo Services - Cache

- NXServerCache
  - Distributed cache system
  - Intercepts all access and events to Nuxeo Core
  - Fills a JBoss Cache instance with DocumentModels
  - Forward invalidation messages to the cache

- NXClientCache
  - Distributed cache system (client part)
  - Intercepts all access to the DocumentManager
  - Transparently looks up the cache before contacting the server

Nuxeo Services – UI related

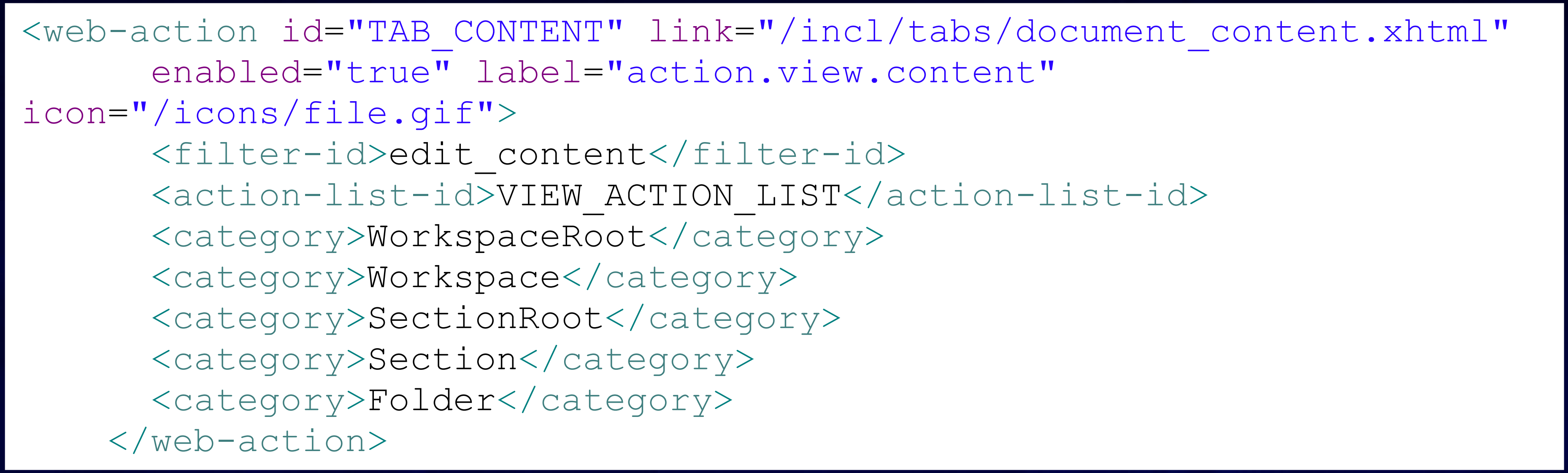
- NXTypeManager
  - Provides configuration for UI management of Documents
    - default views
    - basic layout : Widget and Fields Mapping
    - define allowed subtypes



```
<type id="Folder" coretype="Folder">
     <label>Folder</label>
     <icon>/icons/folder.gif</icon>
     <default-view>view_documents</default-view>
     <subtypes>
       <type>Folder</type>
       <type>File</type>
       <type>Note</type>
     </subtypes>
     <layout>
       <widget jsfcomponent="h:inputText" schemaname="dublincore"
         fieldname="title" />
       <widget jsfcomponent="h:inputTextarea" schemaname="dublincore"
         fieldname="description" />
     </layout>
   </type>
```

Nuxeo Services – UI related

- NXActionManager
  - UI Action management (CPS-like)
  - Used to define applicative UI actions' links/buttons depending on the context



```
<web-action id="TAB_CONTENT" link="/incl/tabs/document_content.xhtml"
      enabled="true" label="action.view.content"
icon="/icons/file.gif">
      <filter-id>edit_content</filter-id>
      <action-list-id>VIEW_ACTION_LIST</action-list-id>
      <category>WorkspaceRoot</category>
      <category>Workspace</category>
      <category>SectionRoot</category>
      <category>Section</category>
      <category>Folder</category>
    </web-action>
```

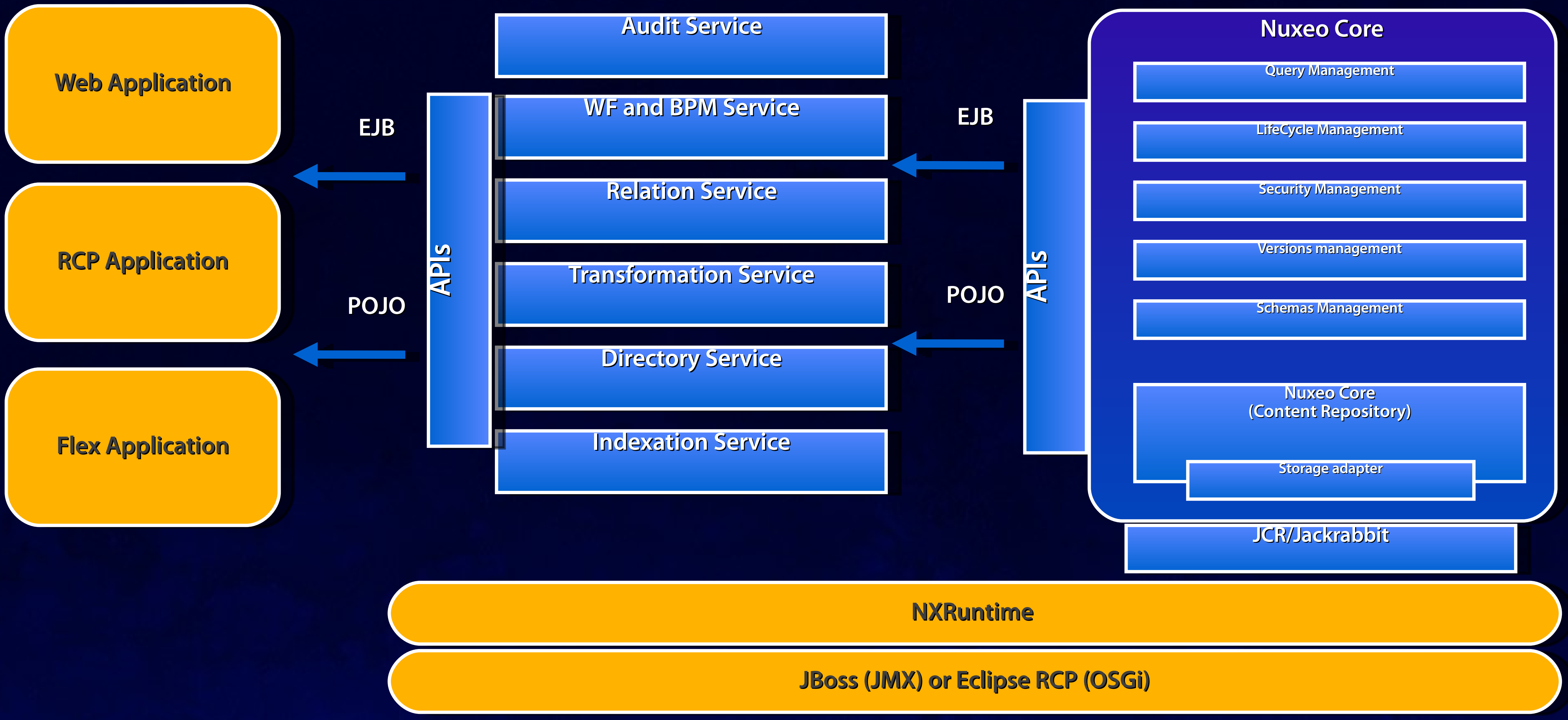
Nuxeo Services - 5.1

- NXSearchService
  - Unified and pluggable Full Text Search Service
- NXPlaceFullConfiguration
  - Attache configuration to a content node
    - Subscriptions
    - local themes configuration
    - local creation rules ...
- To be continued... ;-)

Nuxeo Service Layer

- Most of services are not EJB3 dependants
  - they can be used client side (RCP) or server side (JEE)
- Each service exposes an API
  - POJO API for most of them
  - EJB3 API for all
- All services are remote accessibles
  - can be used from any client type

Nuxeo Service Layer



nuxeo