Nuxeo Core

nuxeo

Nuxeo Core

- Embeddable document management engine
  - For the server
  - For the client

- Pure POJO with a EJB3 facade

- Provides all low-level content management features and API for the Nuxeo 5 stack

nuxeo

Nuxeo Core Features

- Content storage and retrieval
- Content schemas management
- Indexing / query (using NXQL)
- Low level events
- Security (ACL-based, contextual security management)
- Versioning
- LifeCycle

nuxeo

Nuxeo Core – Modules (2)

- NXCore – The repository model
  - Content Storage and Retrieval
  - Storage oriented event model
  - ACL and security management
  - Versionning
- NXCore API – Interfaces
  - Defines NXCore API and interfaces
- NXCore Facade – EJB3 Facade
  - Remoting API
  - Security and Transaction integration

Nuxeo Core – Modules (2)

- NXSchemas
  - Schemas registration
  - Schemas management
- NXQuery
  - Query Model
  - Query Engine
- NXJCRConnector
  - Repository implementation on top of JSR 170 RI (JackRabbit)
- Additionnal modules
  - NXVersionning
  - NXDublinCore
  - ...

nuxeo

Nuxeo Core – Schemas

- Schemas management
  - Nuxeo Core stores content according to its attached schema
  - Nuxeo Core schema format is W3C XML Schemas (XSD)
  - Schemas compliance are enforce at the storage level to insure data integrity
- Schemas contains fields
  - Data type
  - Default values
  - Lazy attributes
- XSD Complex types are supported!
- Schemas are transparently mapped to JCR Nodes Types
- Schemas are registred using contributions to corresponding extension point

nuxeo

Schema Registration example

```xml
<?xml version="1.0"?>

<component name="org.nuxeo.ecm.core.CoreExtensions">

  <extension target="org.nuxeo.ecm.core.schema.TypeService" point="schema">
    <schema name="core-types" src="schema/core-types.xsd" />
    <schema name="common" src="schema/common.xsd" />
    <schema name="dublincore" src="schema/dublincore.xsd" />
    <schema name="uid" src="schema/uid.xsd" />
    <schema name="file" src="schema/file.xsd" />
    <schema name="note" src="schema/note.xsd" />
  </extension>

</component>
```

Schemas registration

Schemas definition

```xml
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:complexType name="content">
  <xs:sequence>
    <xs:element name="encoding" type="xs:string"/>
    <xs:element name="mime-type" type="xs:string"/>
    <xs:element name="data" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

ECM Core – Document Type

- A Document type is defined by
  - a identifier (name)
  - a set of schemas (that can be aliased)
  - a set of facets

- A Facet is a declarative marker
  - It marks the document as compliant with a given behavior
    - Folderish, Versionnable, Downloadable ...
  - Most of the time it is not used directly by the core itself

- Like Schemas, Document Types support inheritance

nuxeo

Type Definition Example

```
  <extension target="org.nuxeo.ecm.core.schema.TypeService"
point="doctype">

    <doctype name="Folder" extends="Document">
      <schema name="common" />
      <schema name="dublincore" prefix="dc" />
      <facet name="Folderish" />
    </doctype>

    <doctype name="Domain" extends="Folder" />

    <doctype name="WorkspaceRoot" extends="Folder" />

    <doctype name="Workspace" extends="Folder">
      <!-- for logo -->
      <schema name="file" />
    </doctype>
  </extension>
```

nuxeo

Life Cycle

- The lifecycle represent the content state (from a functional point of view, not a technical one)
  - Ex: In Progress, Approved, Under Review, Obsolete, Cancelled, etc.
- A lifecycle scheme is associated to each document type
  - The mapping is done via an Extension Point
- LifeCycle != Workflow
  - LifeCycle does not represent a process
  - LifeCycle does not specify security restrictions nor actors
- The Life Cycle defines
  - all allowed states of a document
  - transitions between these states

LifeCycle: default scheme

- States
  - Project (Work)
  - Review (in process)
  - Approved (destination state for a process)
  - Obsolete
- Transitions
  - Review
  - Approve
  - Back to Project
  - Make Obsolete

ECM Core – Security Management

- Nuxeo 5 Security Model
  - ACE : Access Control Entry
    - User / Group – GRANT/DENY – Permission / Group of Permissions
  - ACL : Access Control List
    - Ordoned list of ACE
  - ACP : Access Control Policy
    - Ordoned list of ACL
- Each document can be associated with a ACP
  - Security is placefull
  - Security is inherited
- A document can have several ACL
  - One ACL for basic rights
  - One ACL for each process

nuxeo

Nuxeo Core – Security Management

- Nuxeo Core
  - Stores security descriptors
  - Provides an API for managing ACPs/ACLs/ACEs
  - Check security on each access
- Security descriptors are handled by a specific Core service
  - currently, security informations are stored on the document
  - it could be stored into a separated location (via an EP)

Nuxeo Core - Events

- Nuxeo Core offers an Event system
  - Before and After each document related operation
- Event Handlers can be hooked inside the Core
  - Using an Extension Point (Synchronous)
    - Used for implementing some build-in features
      - Audit
      - DubinCore schema management (modification date ...)
      - Pluggable Versionning Policy
- Events are also for forwarded as JMS messages
  - Enables the implementation of asynchronous event handlers
  - Enabled non-core components to register and get core's events

nuxeo

Nuxeo Core - Events

- Declaring an EventHandler is so easy! :-)

```
<?xml version="1.0"?>

<component name="DublinCoreStorageService">

<implementation
class="org.nuxeo.ecm.platform.ec.dublincore.service.DublinCoreStorageService"/>

<require>org.nuxeo.ecm.core.listener.CoreEventListenerService</require>

<extension target="org.nuxeo.ecm.core.listener.CoreEventListenerService"
   point="listener">
<listener name="dclistener"
  class="org.nuxeo.ecm.platform.ec.dublincore.listener.DublinCoreListener"
  />
</extension>

</component>
```

Nuxeo Core - Query

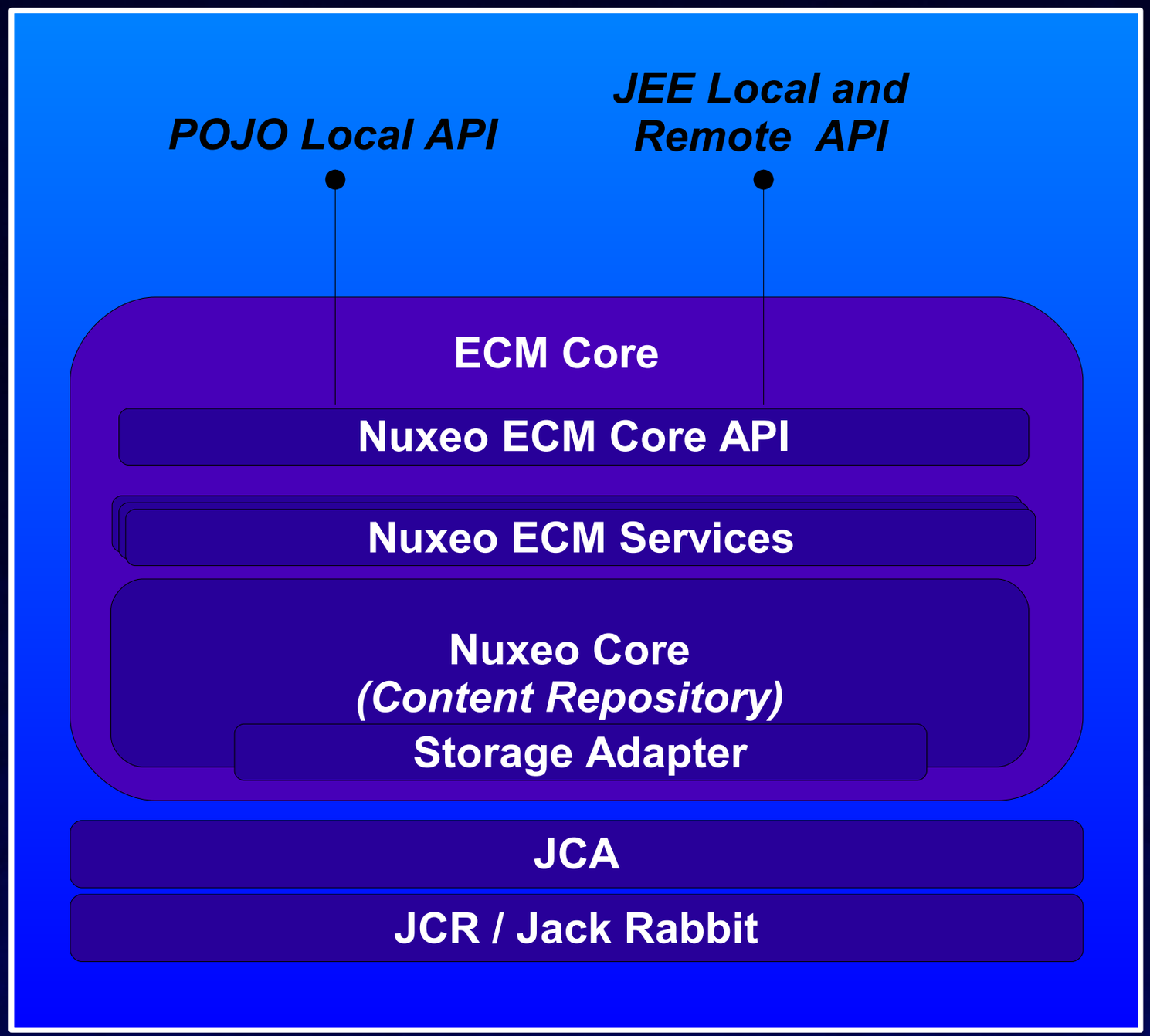
- Nuxeo Core includes a Query System
  - NXQL : SQL-like query language for document-oriented data
- Returns lists of *DocumentModels*
- Simple yet powerfull Query langage!
  - SELECT * FROM document WHERE dc:contributors = '?' ORDER BY dc:modified DESC
  - SELECT * FROM document WHERE ecm:path STARTSWITH '?/' ORDER BY dc:modified DESC

nuxeo

Nuxeo Core Facade

- Nuxeo Core it self is Pure POJO
  - Not remote accessible
  - No JEE integration
- Core Facade provides JEE Integration using a EJB3 layer on top of Nuxeo Core's services
- It provides
  - Remoting
    - EJB3 Remoting or SOAP
  - JEE Security integration
    - JAAS Principal is transmited
  - Transaction integration via JCA
  - A Connection Pooling system
- This is the DocumentManager

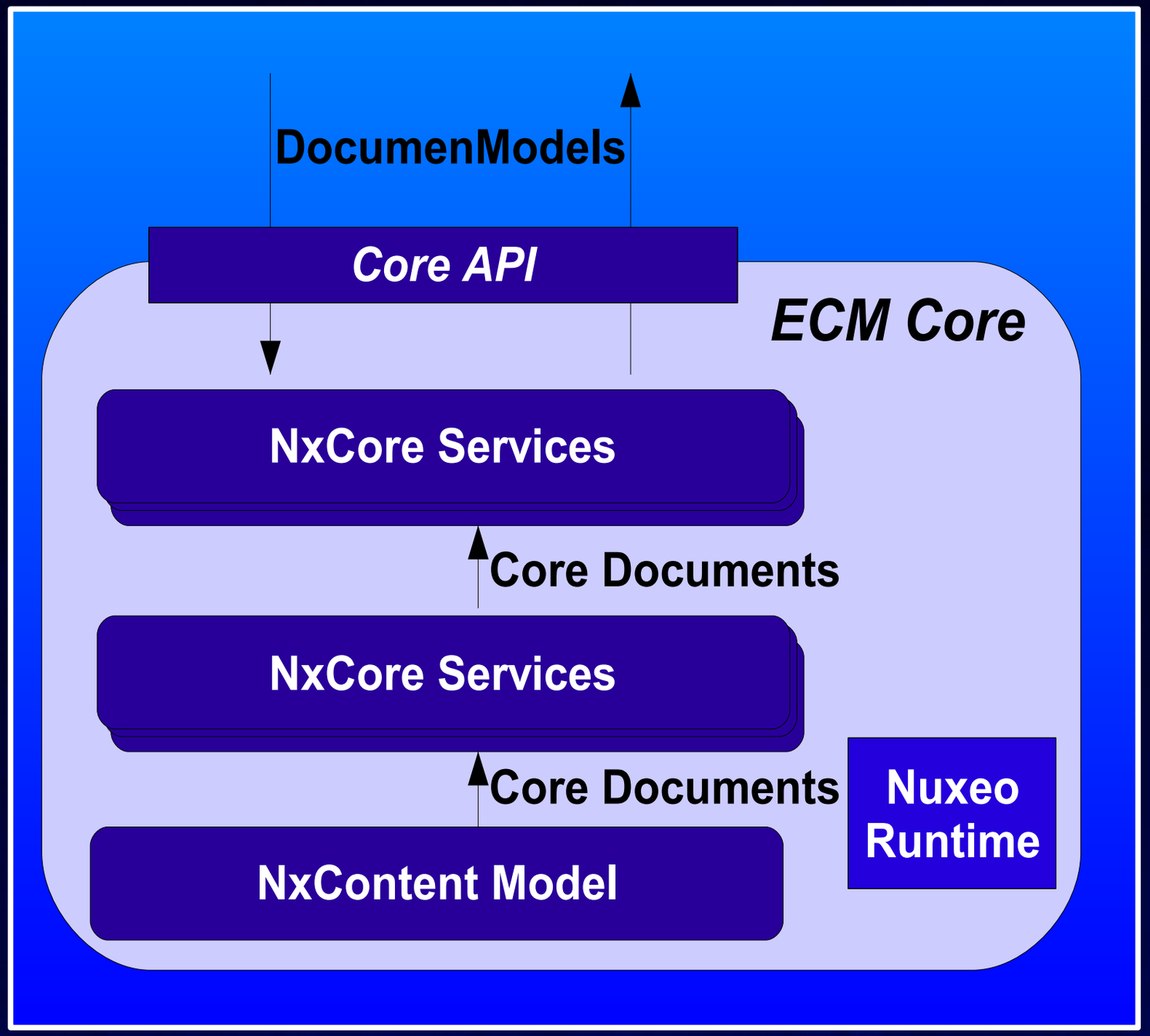
Nuxeo Core Architecture Diagram



nuxeo

Document Object

- Inside the Core
  - CoreDocuments
- Outside the Core (In and Out)
  - DocumentModel



CoreDocument

- CoreDocument is
  - a type identifier (name)
  - a set of schemas
  - a set of facet (Folderish, Orderable ...)
- CoreDocument has no GUI information
- CoreDocument always holds all informations (references)

Motivations for DocumentModel

- Detach document from Nuxeo Core
  - Use Serializable objects
  - Reduce network calls
- Transparent Lazy fetching
  - DocumentModel know how to reconnect to the Core to fetch missing data
- Transmit a consolidated artifact of the content object
  - DocumentModel will be completed by the service layer
- Caching and Invalidation management

nuxeo

Nuxeo Core Extensions

- NXDublinCore
  - MetaData update

- NXUIDGenerator
  - Customize UID generation

- NXDocumentUpdater
  - Synchronize File and Document MetaData

- NXVersionning
  - Pluggable version policy management