Center for Open Middleware

Building interoperable read-write Linked Data applications with the W3C Linked Data Platform and the LDP4j framework

ESWC 2015 - May 31st , 2015 - Portoroz (Slovenia)

Miguel Esteban Gutiérrez,
Nandana Mihindukulasooriya,
Raúl García Castro,
Center for Open Middleware / Ontology Engineering Group
Universidad Politécnica de Madrid,
Spain.

Building interoperable read-write Linked Data applications with the W3C Linked Data Platform and the LDP4j framework

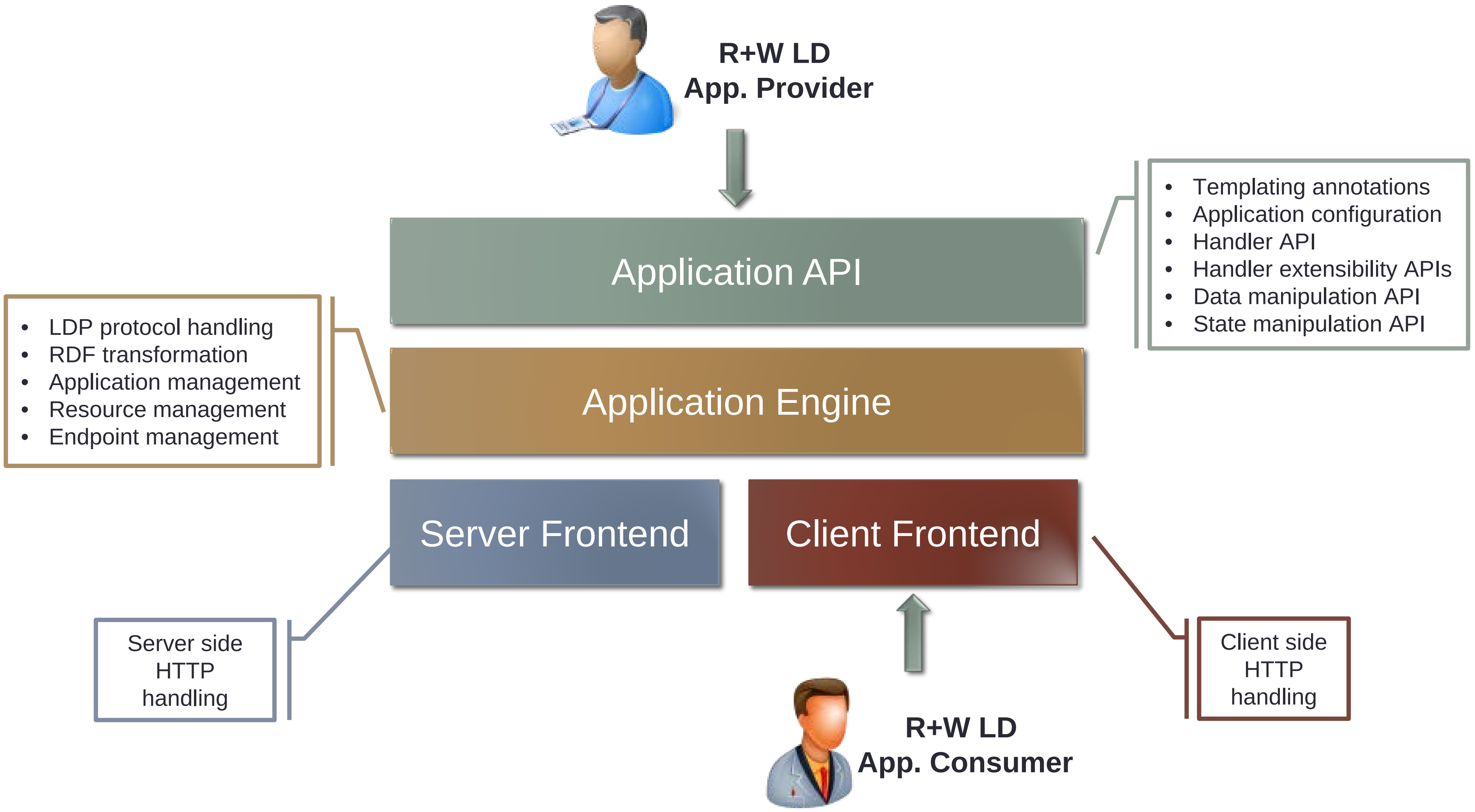
LDP4j 101

LDP4j to the rescue

- Java-based framework for the development of read-write Linked Data applications
  - **Motto:** Focus on your business logic, we will take care of the rest
  - **License:** Apache License, v2.0

- Features:
  - **Simplified business object handling:**
    - Lifting from and lowering to an intermediate representation that can be automatically unmarshalled from or marshalled to RDF, respectively.
  - **LDP support**
    - Protocol conversation control
    - Transparent metadata management
  - **REST aware**
    - Transparent publication of RDF URIRefs.
  - **HTTP compliant**
    - The framework takes care of fulfilling the requirements prescribed by the HTTP related RFCs (i.e., conditional requests processing, content negotiation, entity tag handling,…)

High level architecture



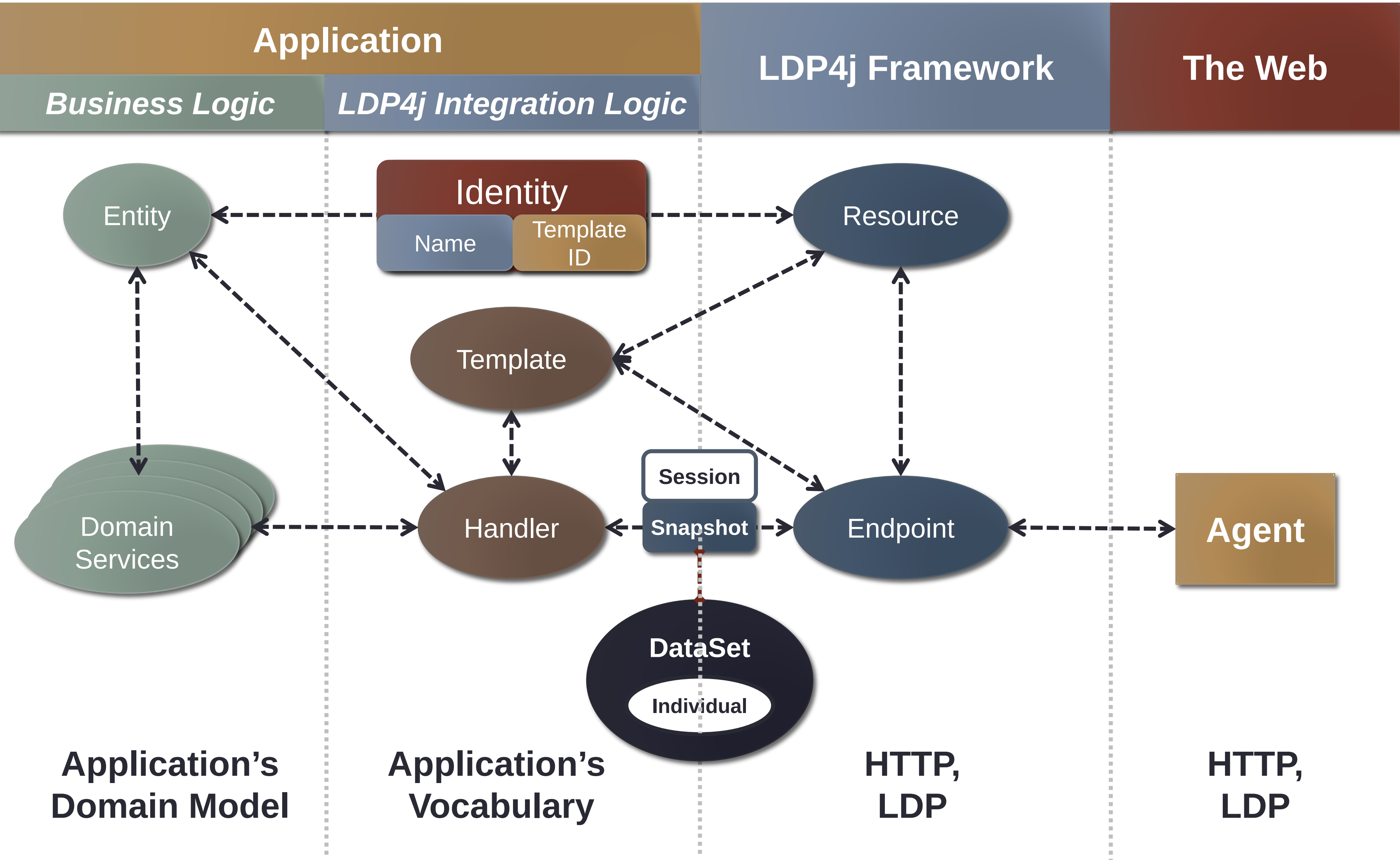
- Templating annotations
- Application configuration
- Handler API
- Handler extensibility APIs
- Data manipulation API
- State manipulation API

- LDP protocol handling
- RDF transformation
- Application management
- Resource management
- Endpoint management

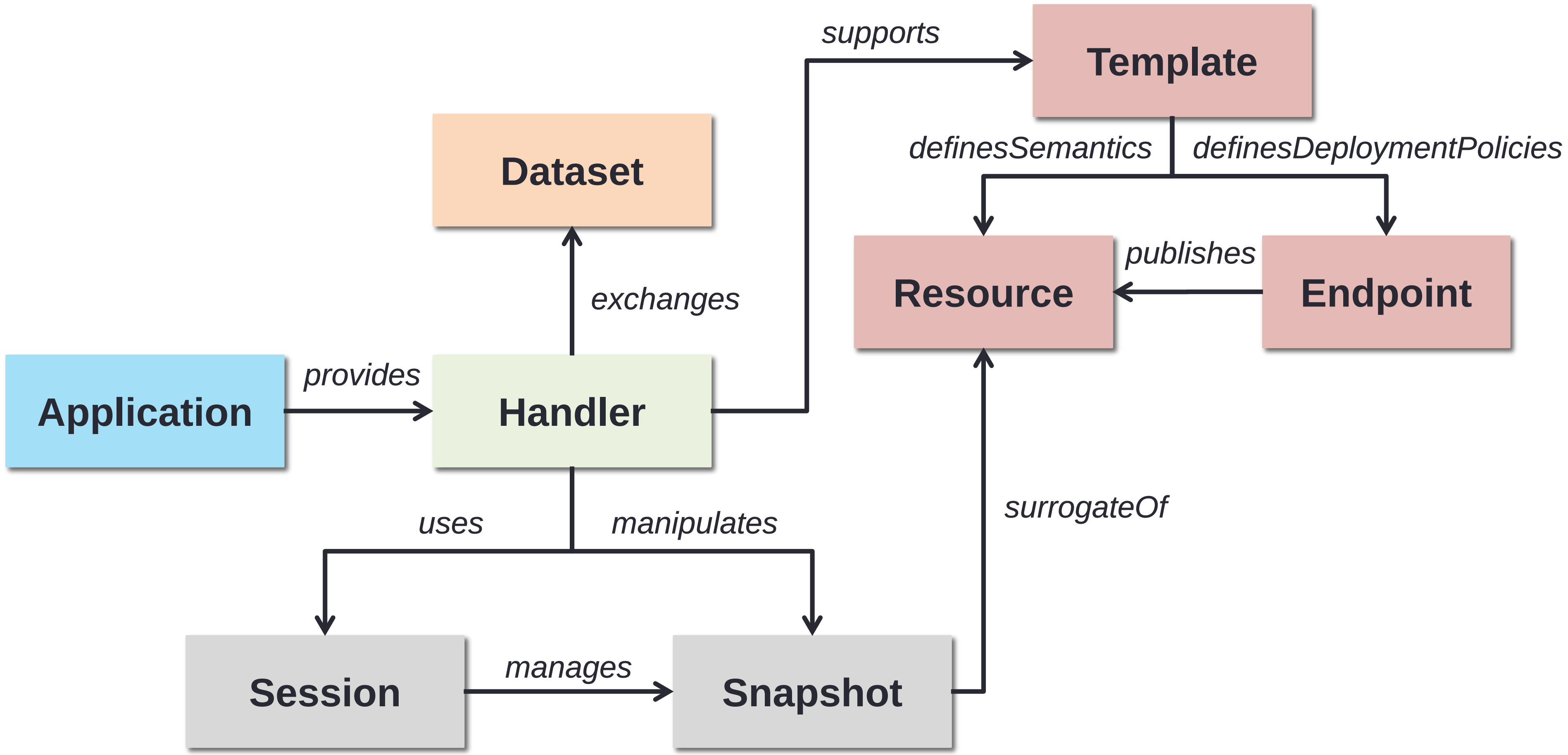
Server side HTTP handling

Client side HTTP handling

What's an LDP4j Application?



Application model in detail



Applications

- The *glue* that allow the LDP4j framework know about
  - how to publish ***resources*** for each of the entities defined by the application, and
  - how to interact with the application whenever a client requests the interaction with any of the published **resources**
- **Resources** represent entities of the application's domain model that the LDP4j framework knows about…
- … which are published via **endpoints** that implement the LDP specification.

Templates

- Specificy the type of resources used by the application and their (hierarchical) organization

Resource	Basic Container	Direct Container	Indirect Container
id	id	id	id
name	name	name	name
description	description	description	description
handler	handler	handler	handler
attachments	attachments	attachments	attachments
	memberHandler	memberHandler	memberHandler
	memberPath	memberPath	memberPath
		membershipPredicate	membershipPredicate
		membershipRelation	membershipRelation
			insertedContentRelation

Attachment
id
name
description
handler
path
predicate

Mandatory	
Optional	

⬛	Informational	🟥	Behaviour
🟩	Organization	🟪	Vocabulary

Handlers (I)

- Implement the behaviour, *a.k.a.*, *business logic*, of the resources defined by the templates
- Minimum required behaviour:
  - **`org.ldp4j.application.ext.ResourceHandler`**
    - Common operations for RDF sources (*i.e.*, retrieval)
  - **`org.ldp4j.application.ext.ContainerHandler`**
    - Common operations for Containers (*e.g.*, creation of new resources)
- Behaviour can be extended as needed:
  - **`org.ldp4j.application.ext.Modifiable`**
    - Support for resource updates
  - **`org.ldp4j.application.ext.Deleteable`**
    - Support for resource deletion

Handlers (II)

- Responsibilities:
  - **Management of the life-cycle of the resources**,
    - To let the LDP4j framework
      - manage of the **endpoints** required for publishing the resources, and
      - control protocol specific metadata required by the LDP specification *(e.g.,* entity tags*)*
    - Carried out via **Sessions** and **Snapshots**
  - **Data manipulation**:
    - Consume data provided by a request (*i.e.*, POST, PUT)
      - Map the data provided in a request defined in terms of the application's vocabulary to objects of the application's domain model
    - Provide data to send in a response (*i.e.*, GET)
      - Map objects of the application's domain model to data that can be marshalled in the response in terms of the application's vocabulary
    - Carried out using **DataSets, Individuals, Properties,** and **Literals**

Resource life-cycle management (I)

Actors

- **Snapshots:** act as surrogates of the resources during the execution of a handler operation
  - **`org.ldp4j.application.session.ResourceSnapshot`**
    - Manage attachments
  - **`org.ldp4j.application.session.ContainerSnapshot:`**
    - Manage containment relations between resource
  - **`org.ldp4j.application.session.AttachmentSnapshot:`**
    - Manage parent-child relation between resources
- **Sessions:** coordinates the management process
  - **`org.ldp4j.application.session.ReadSession:`**
    - Available on resource retrieval operations
  - **`org.ldp4j.application.session.WriteSession:`**
    - Available on resource modification operations

Snapshot management (I)

Operations (I)

- **Snapshot creation:**
  - How:
    - Attachment creation (on write operations)
    - Member creation (on write operations)
    - Root resource creation (on application startup)
  - Side-effects:
    - Create new resources, create endpoints for publishing the new resources, and update the entity tags and last modified dates for resources whose description must be modified as a result of the creation of the resources (*e.g.*, resources that have attached membership aware containers upon new member creation)

Constrained by the templates defined by the application

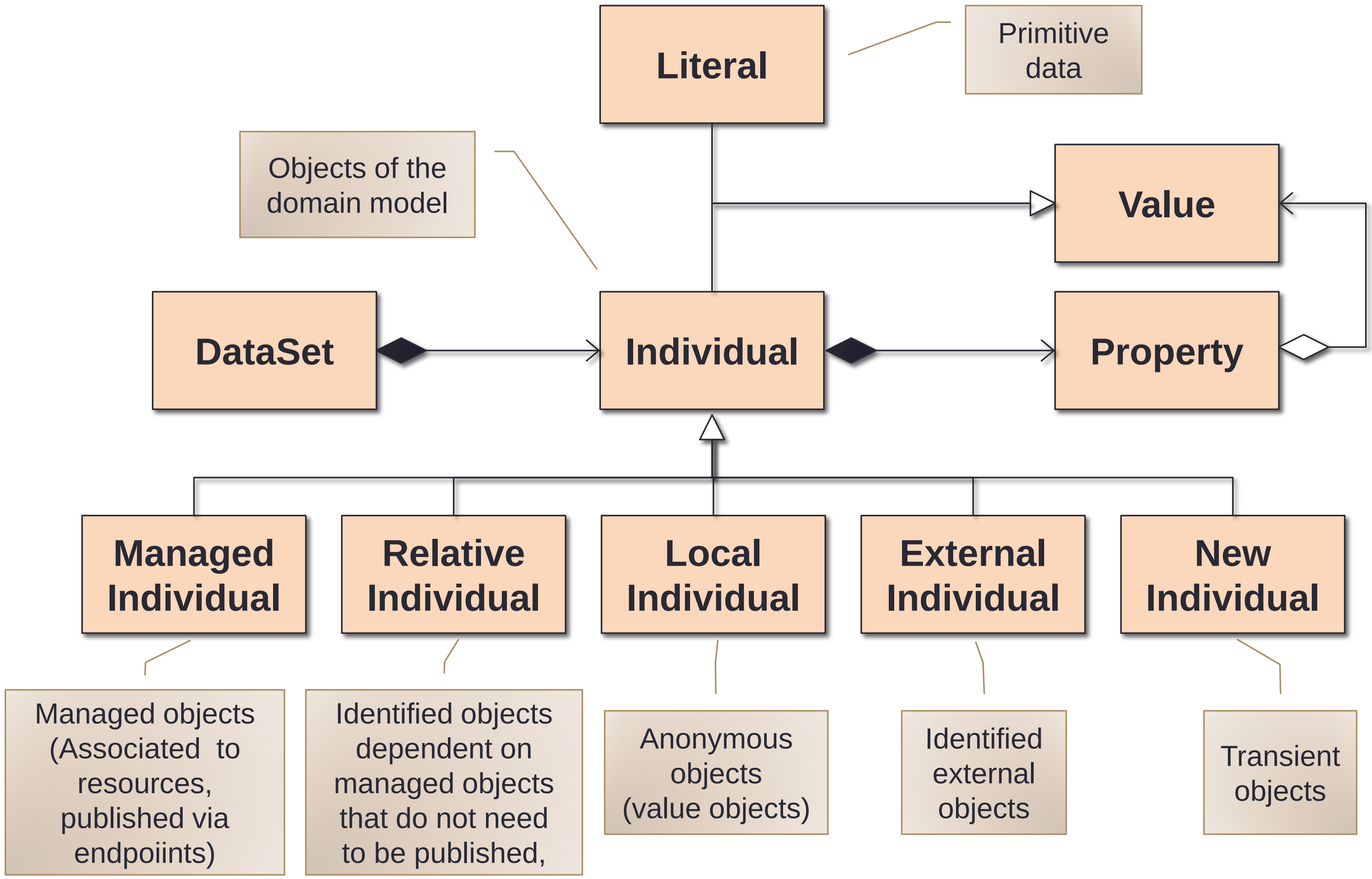
- **Snapshot update:**
  - How:
    - Via session (on write operations)
  - Side-effects:
    - Update the entity tag and last modified date of the affected resource.

Resource life-cycle management (III)

Operations (II)

- **Snapshot deletion:**
  - How:
    - Attachment deletion (on write operations)
    - Member deletion (on write operations)
    - Via session (on write operations)
  - Side-effects:
    - Delete existing resources, remove the endpoints used for publishing them, and update the entity tags and last modified dates of resources whose description must be modified as a result of the deletion of the resources (*e.g.*, containers upon deletion of a member)
    - Hierarchy aware
- **Save changes:**
  - Make changes permanent
- **Discard changes**

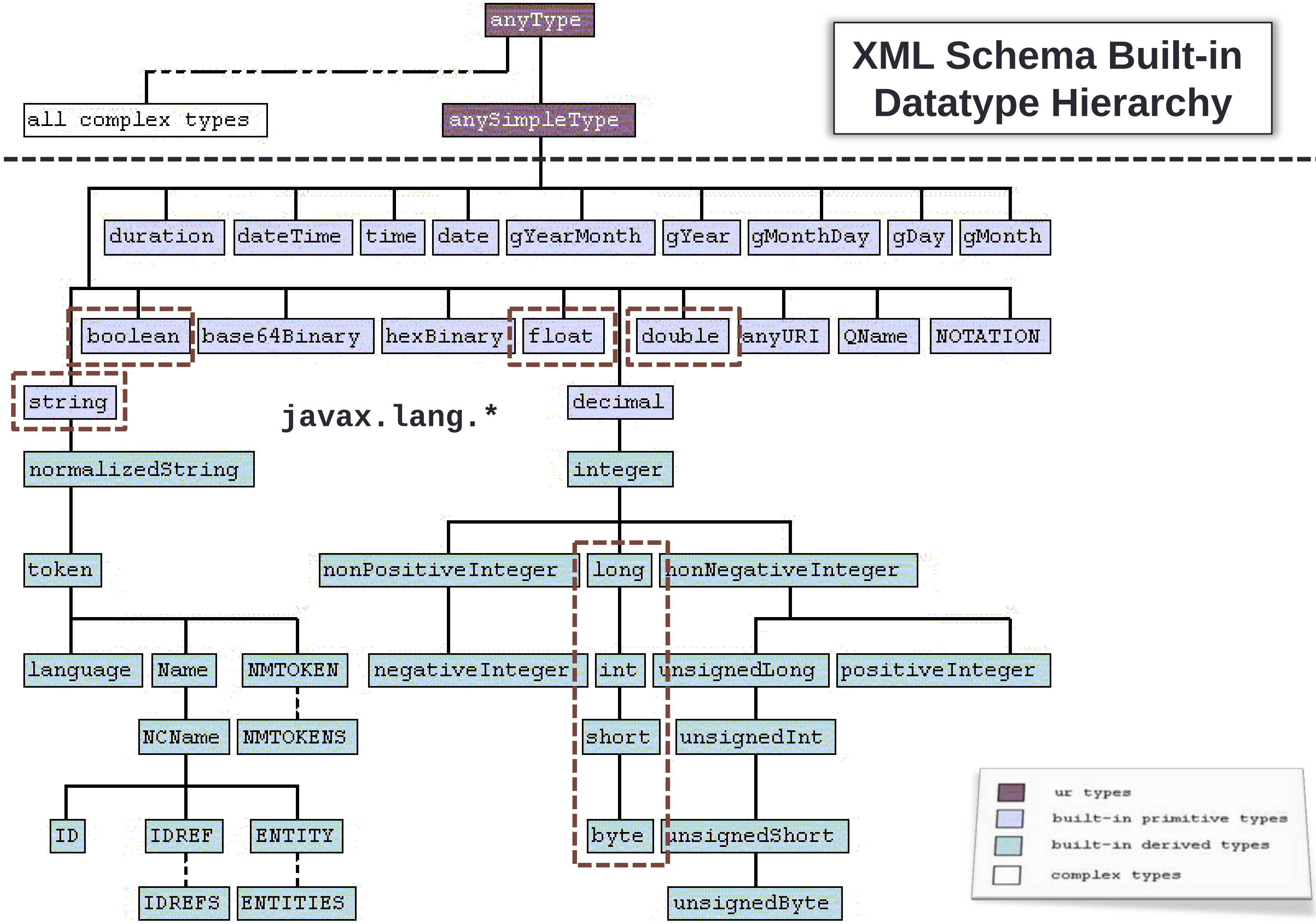
Data manipulation (I)


Literal
Primitive data
Objects of the domain model
Value
DataSet
Individual
Property
Managed Individual
Relative Individual
Local Individual
External Individual
New Individual
Managed objects (Associated to resources, published via endpoiints)
Identified objects dependent on managed objects that do not need to be published,
Anonymous objects (value objects)
Identified external objects
Transient objects

Data manipulation (II)

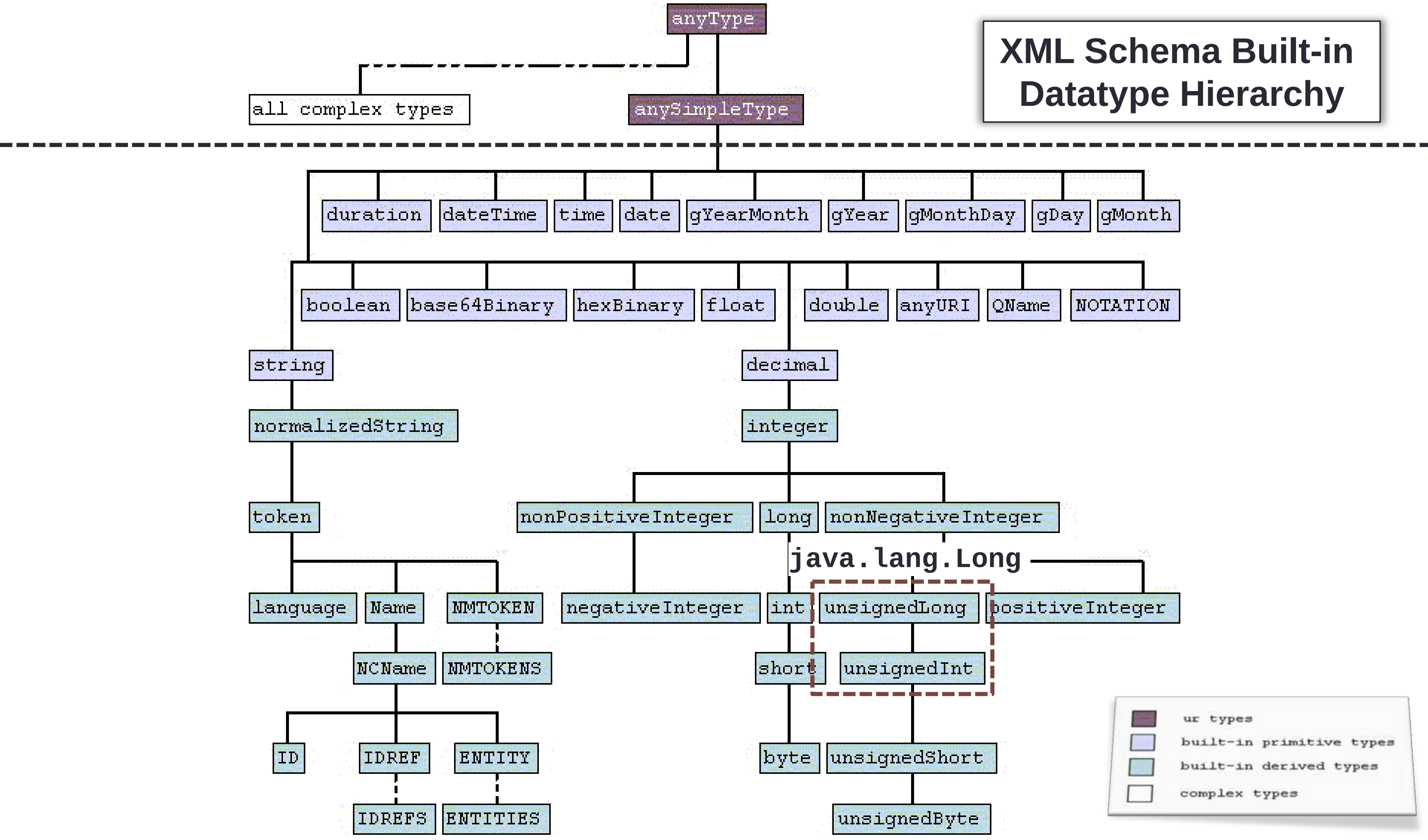
Literal boxing/unboxing

XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

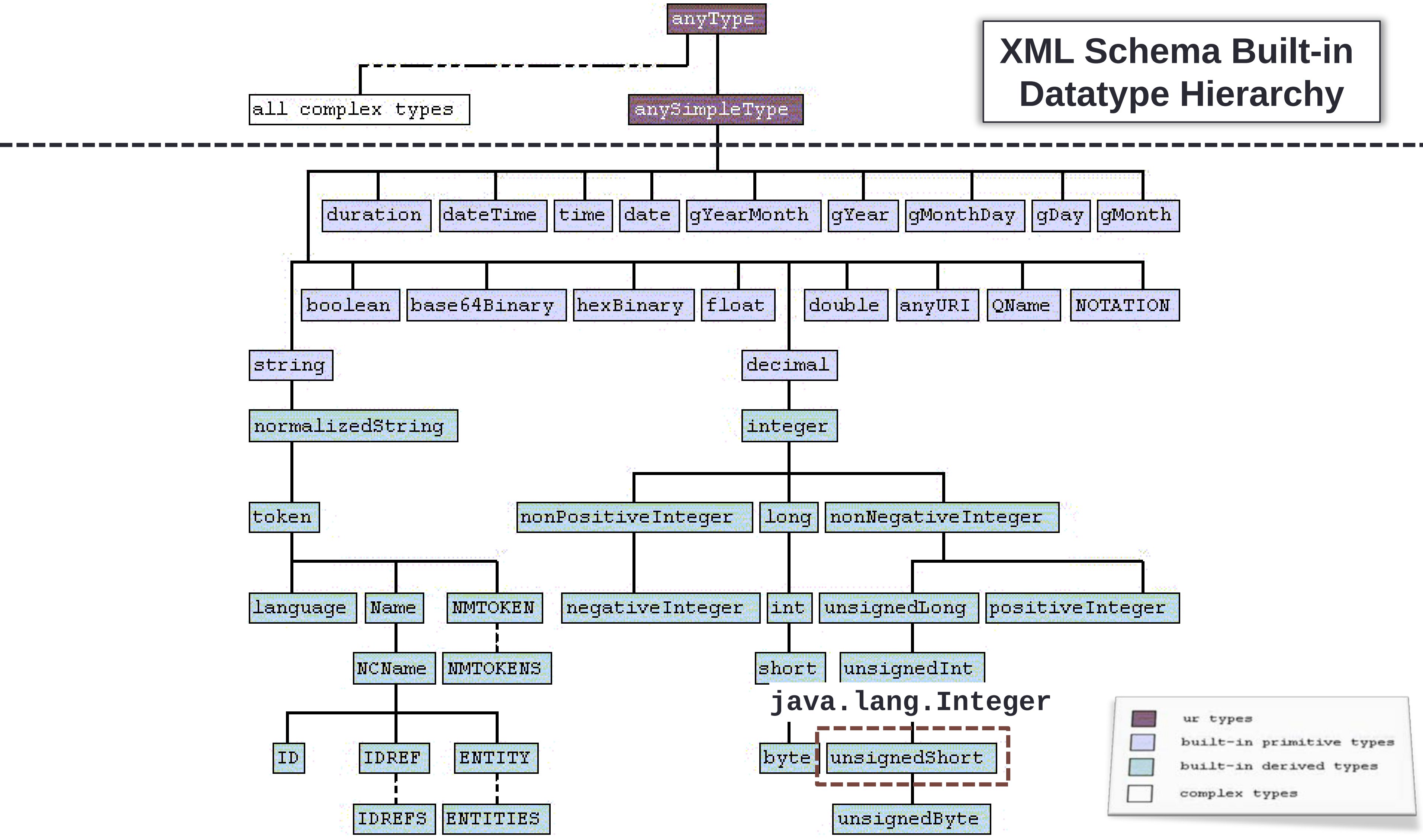
Literal boxing/unboxing



Data manipulation (II)

Literal boxing/unboxing

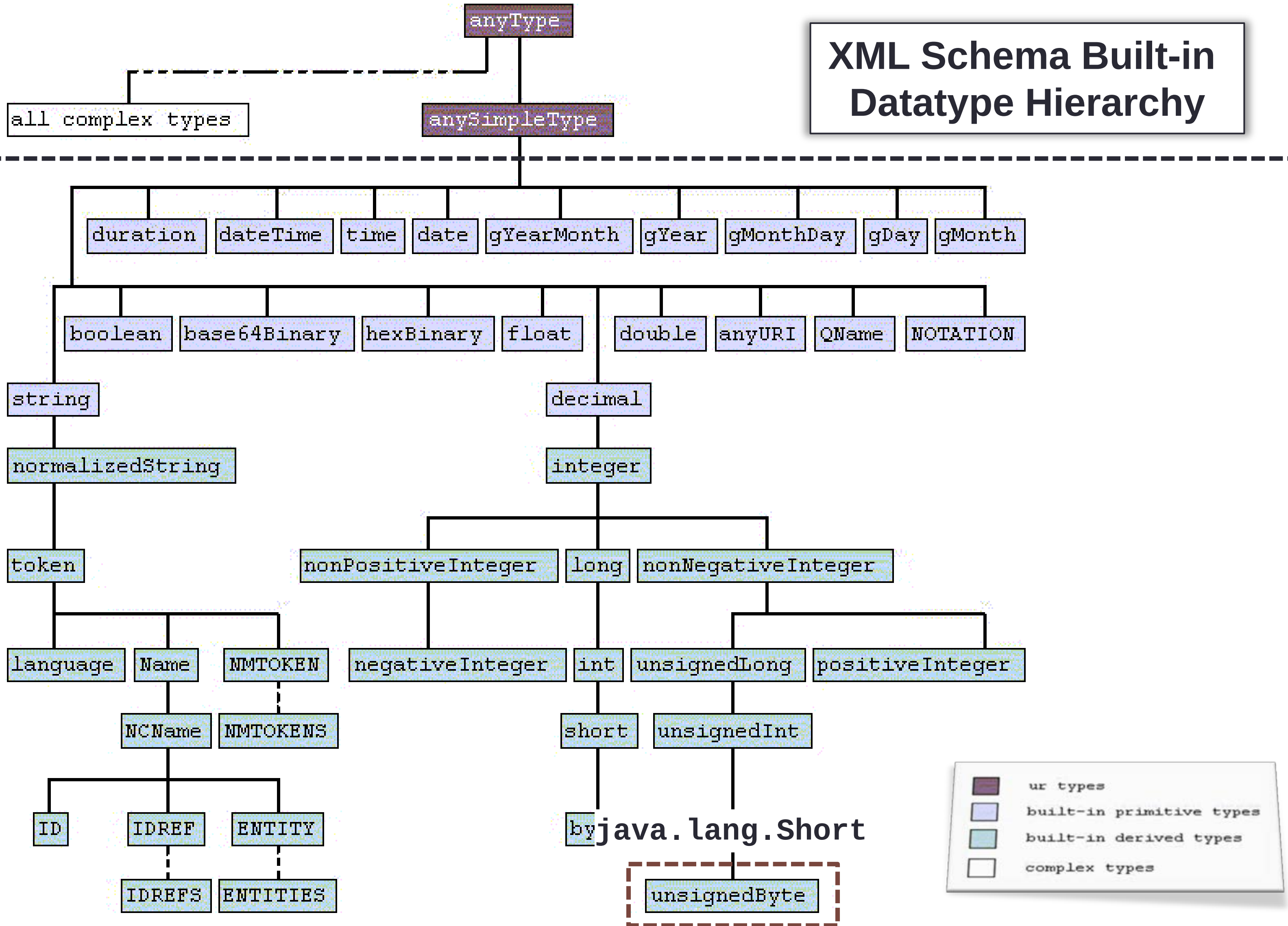
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

Literal boxing/unboxing

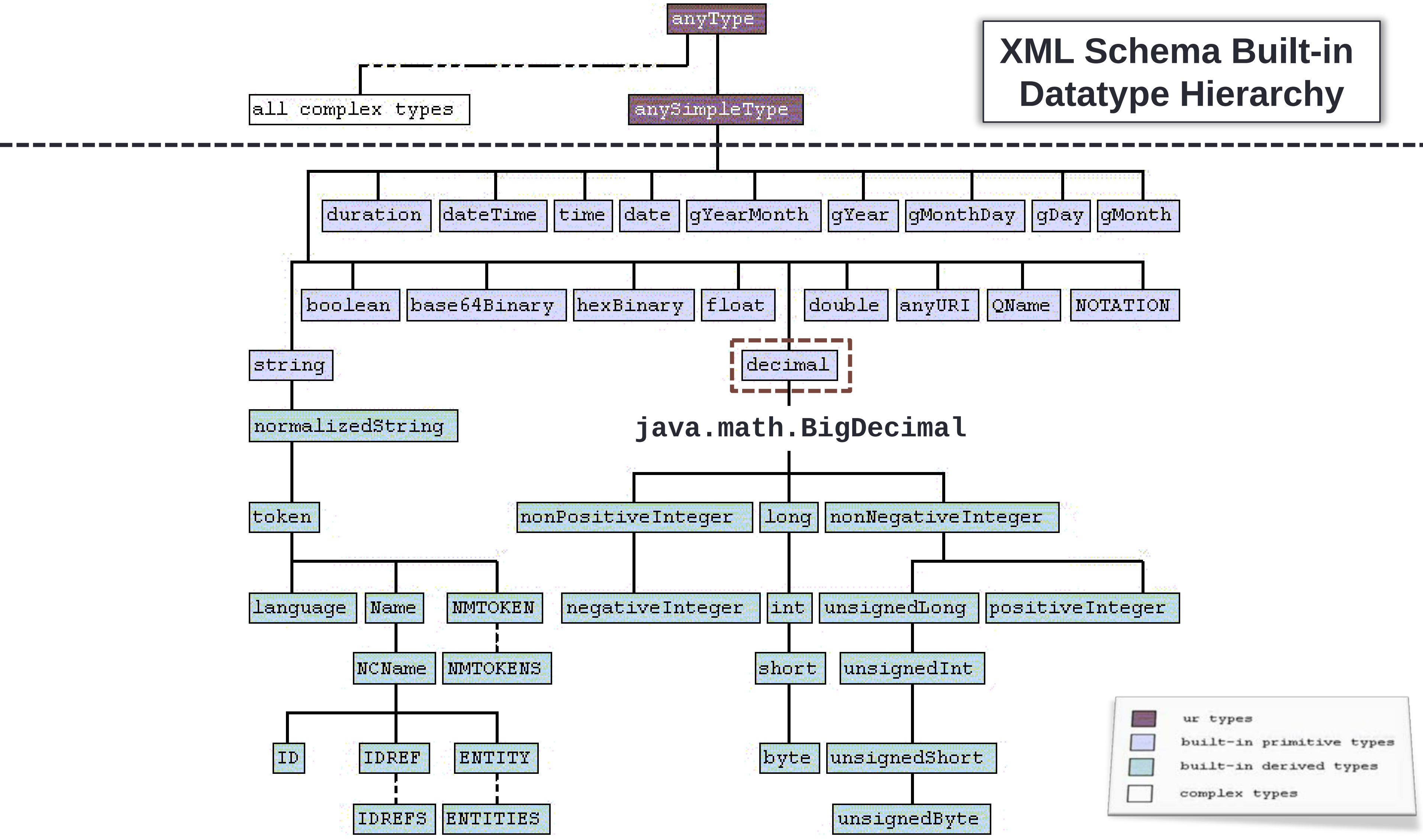
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

Literal boxing/unboxing

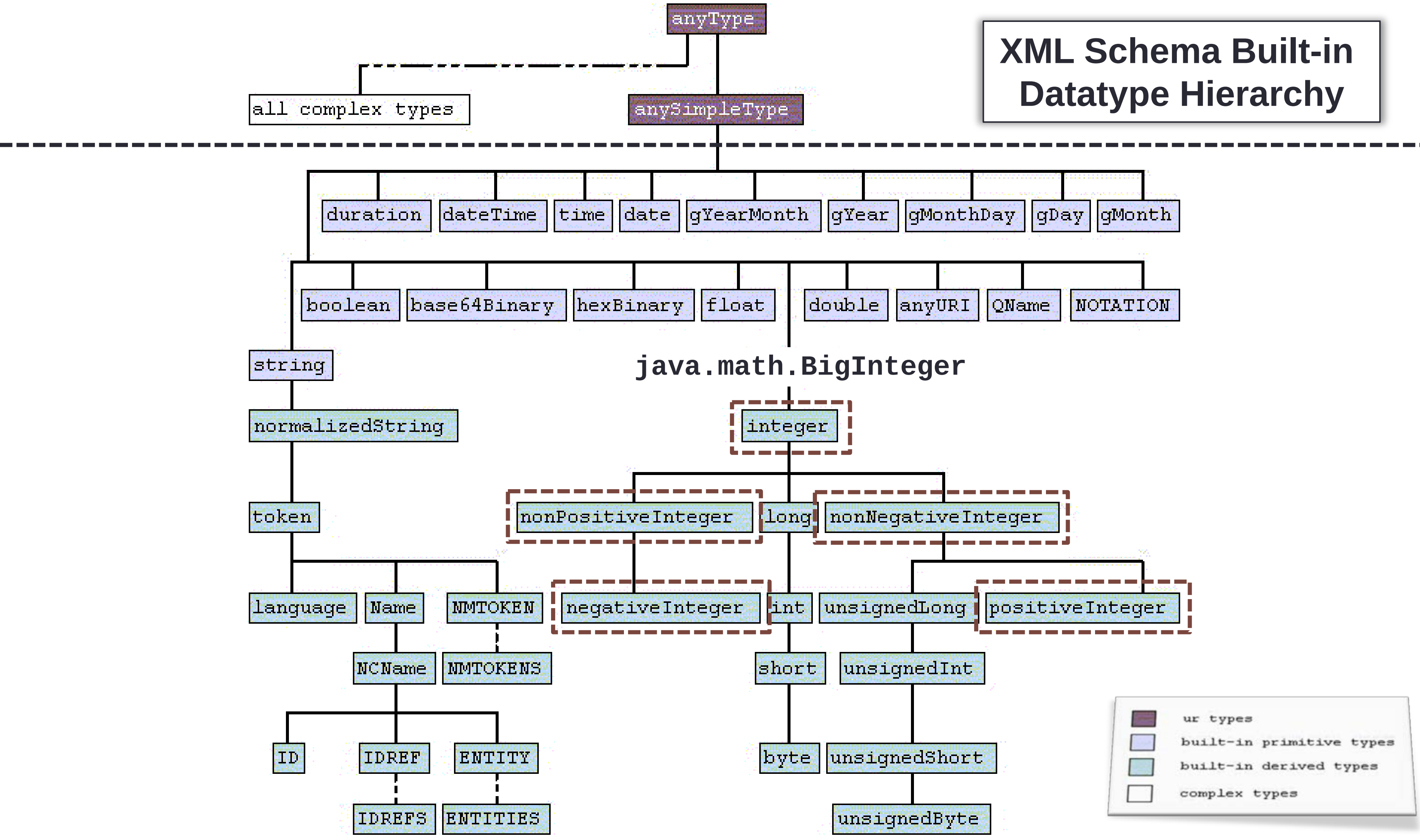
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

Literal boxing/unboxing

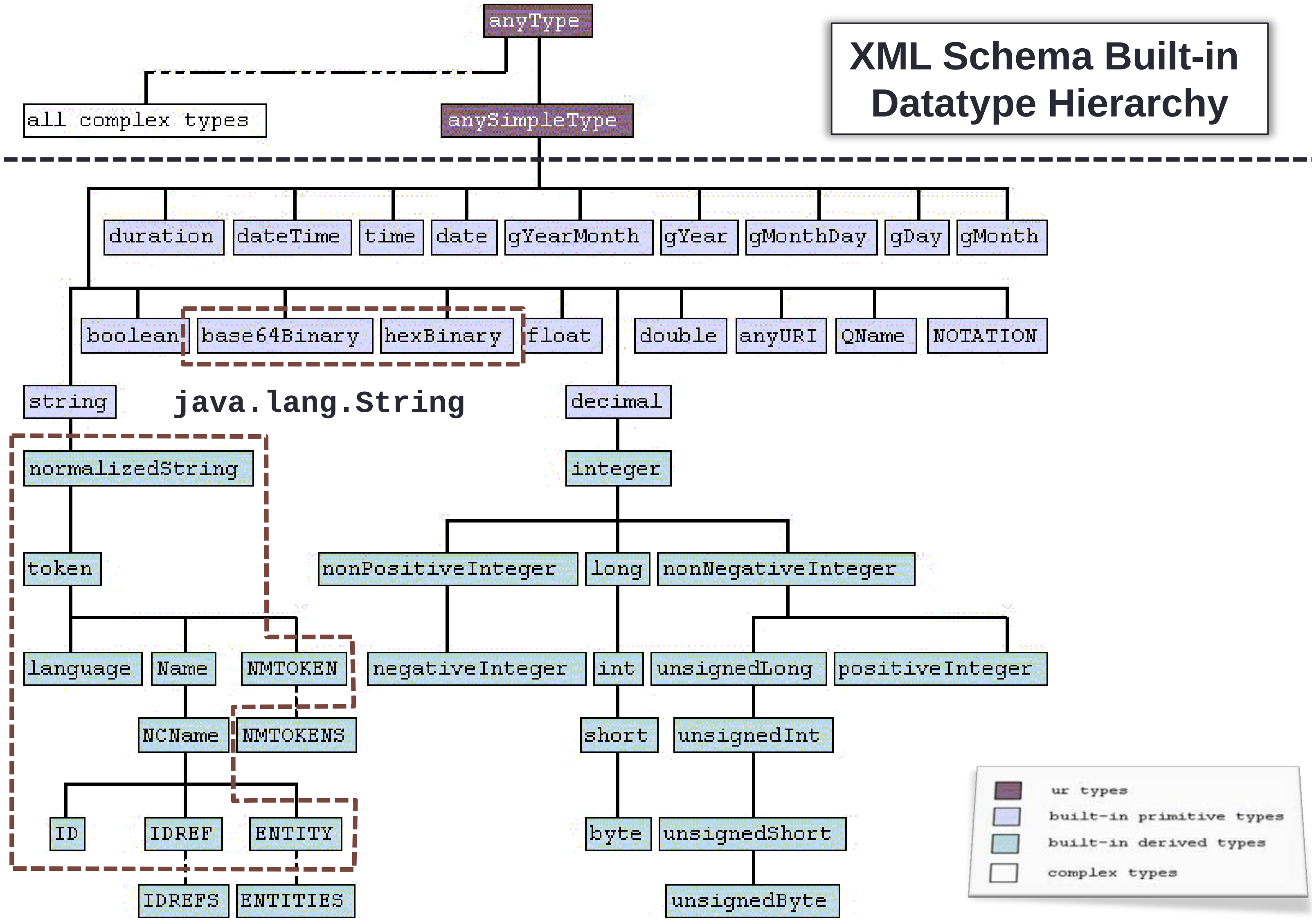
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

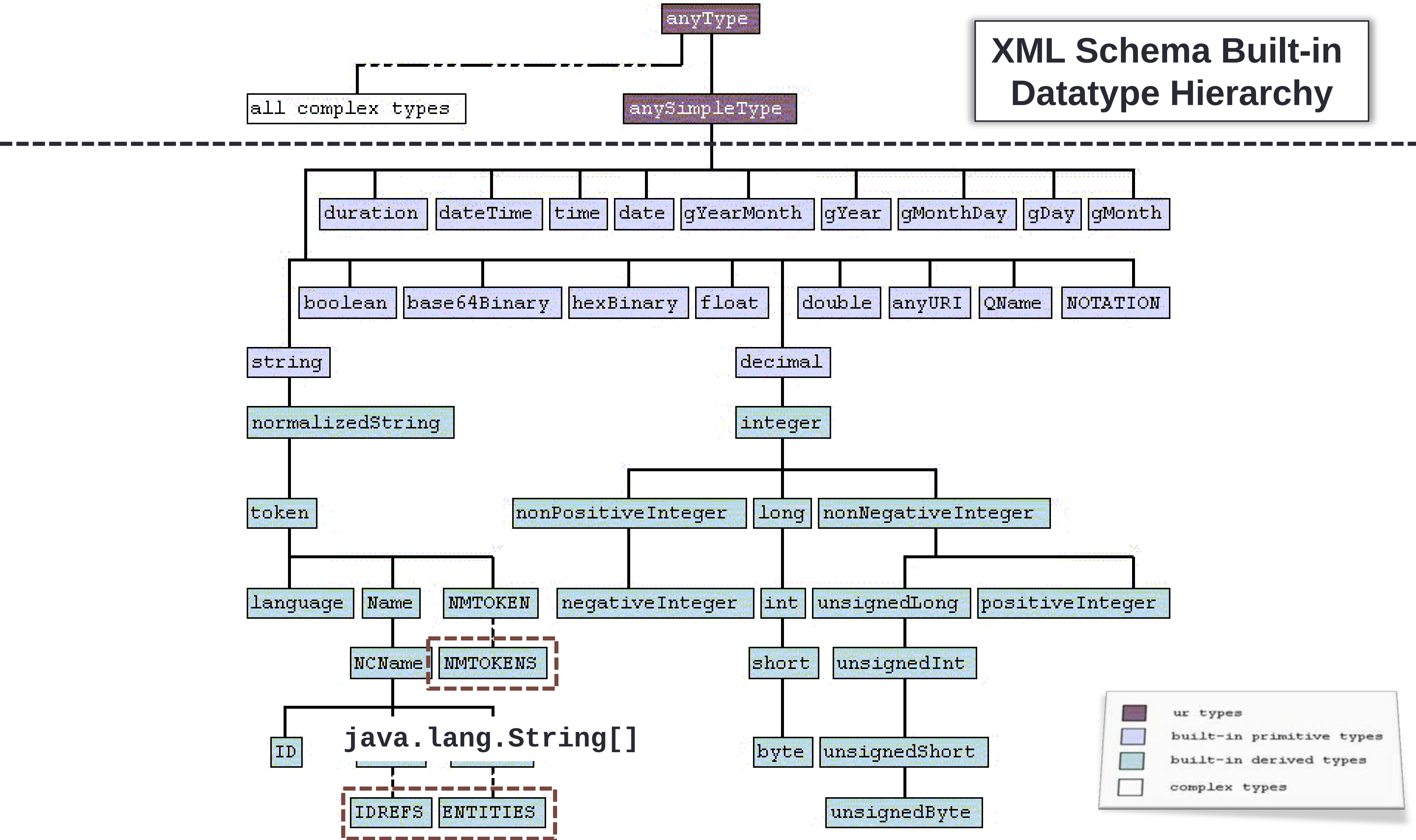
Literal boxing/unboxing

XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

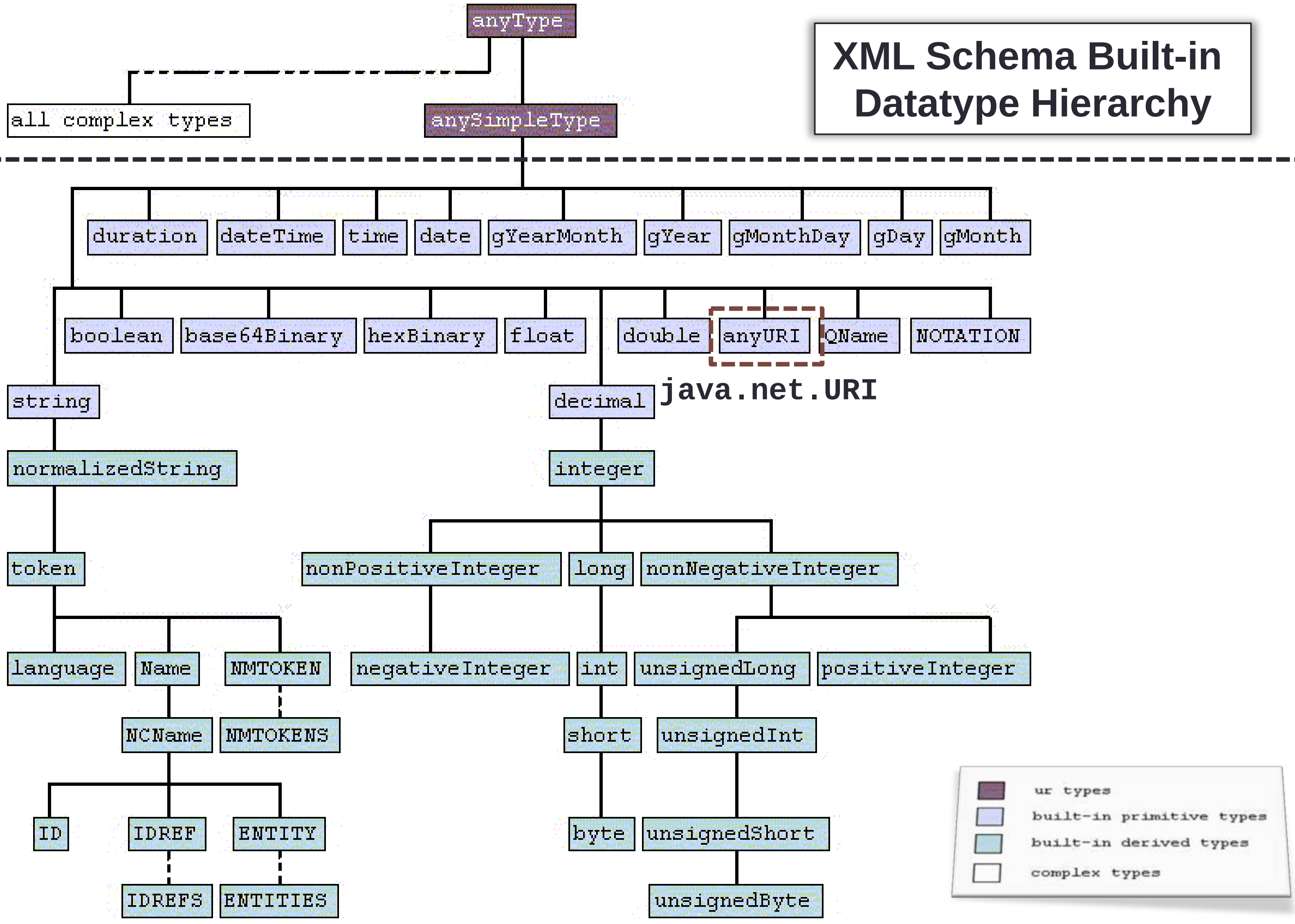
Literal boxing/unboxing



Data manipulation (II)

Literal boxing/unboxing

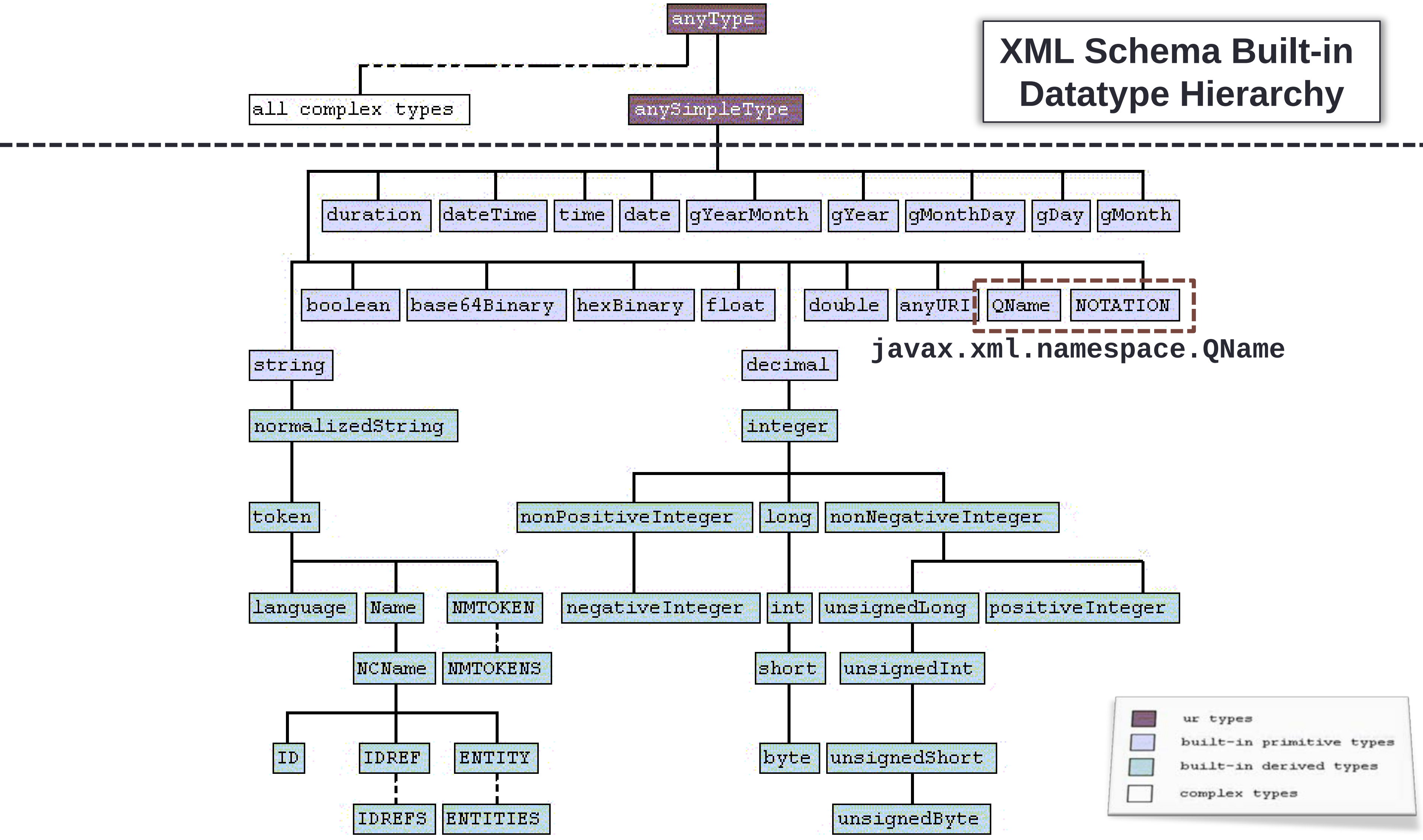
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

Literal boxing/unboxing

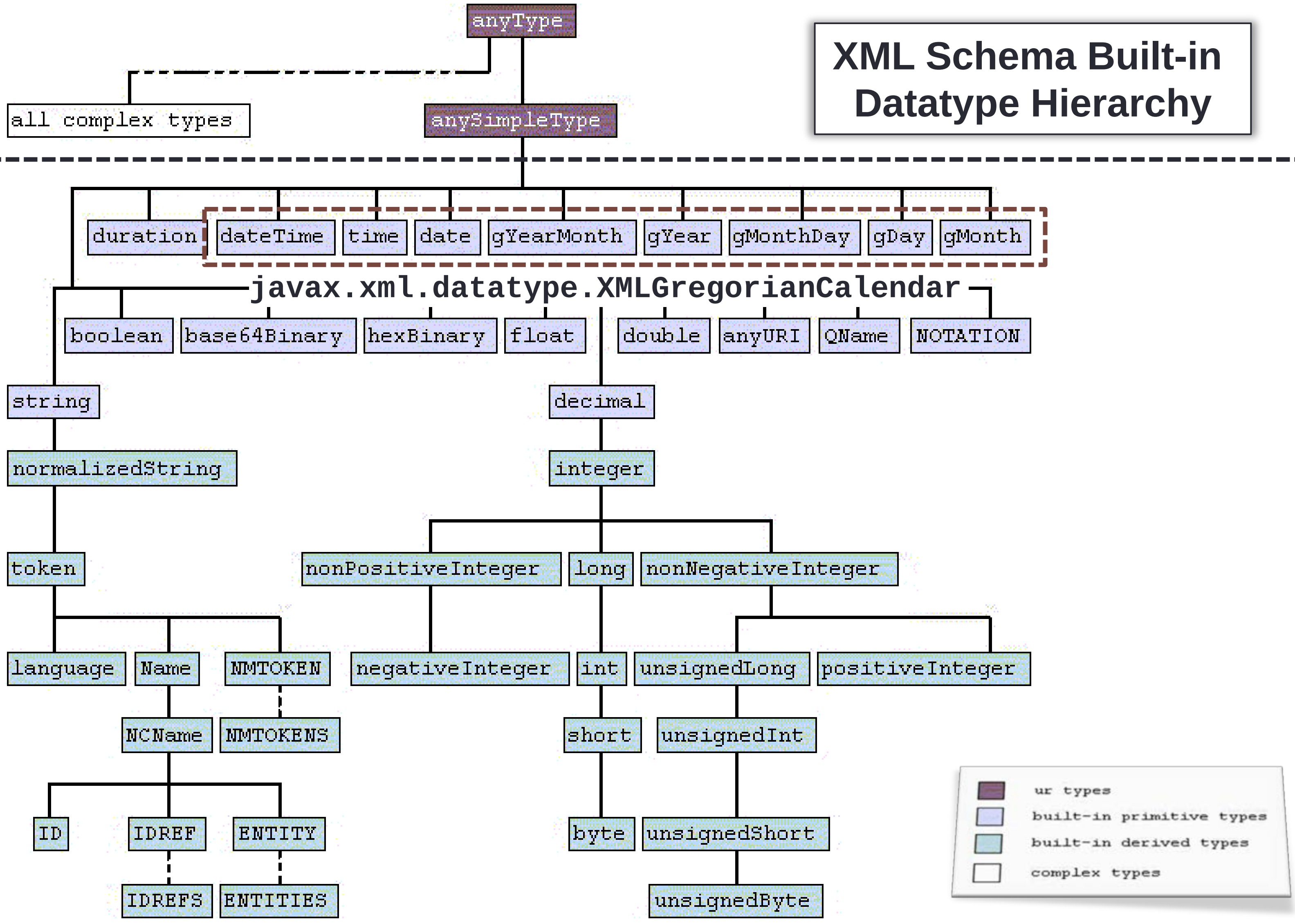
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

Literal boxing/unboxing

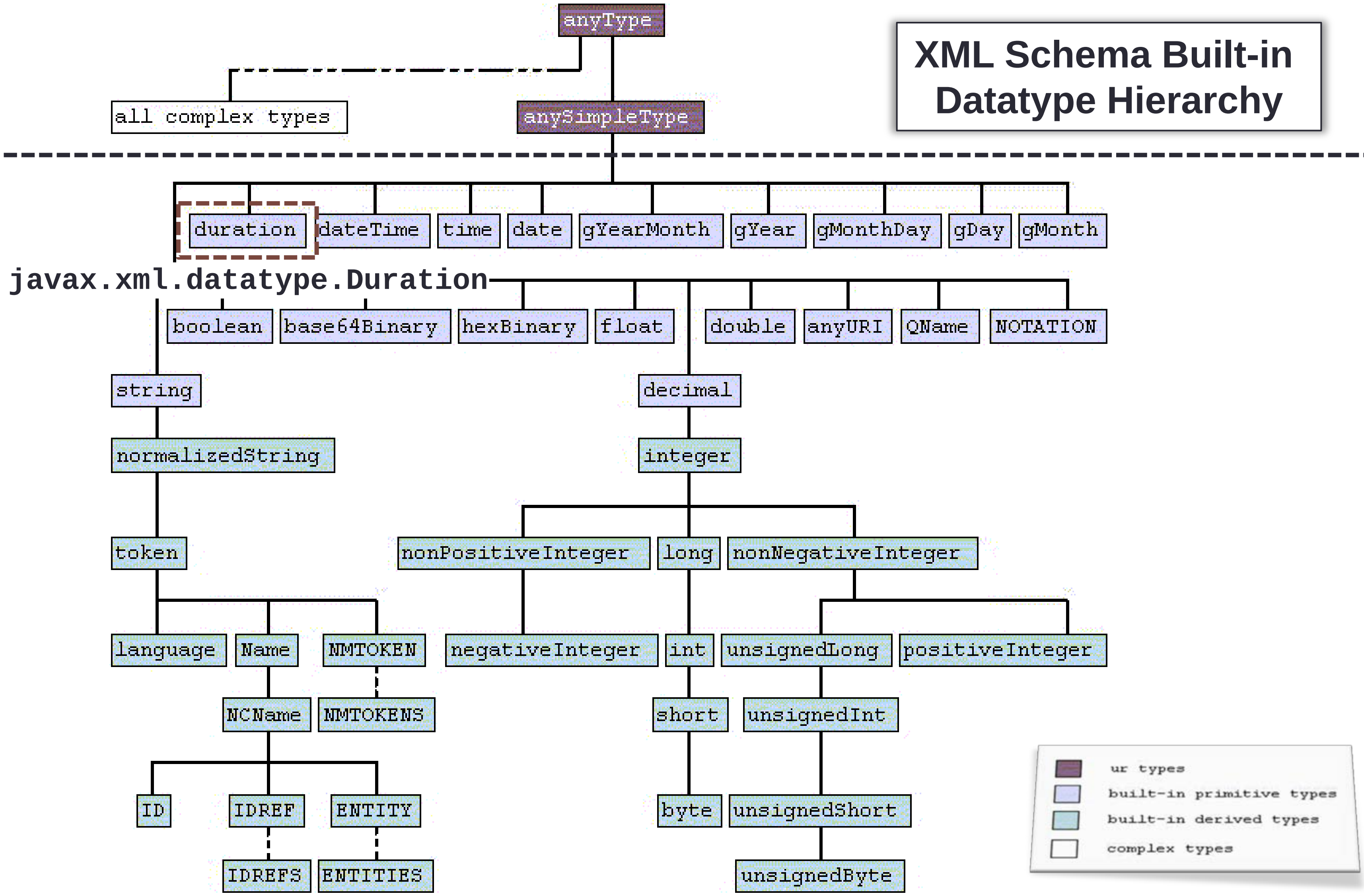
XML Schema Built-in Datatype Hierarchy



Data manipulation (II)

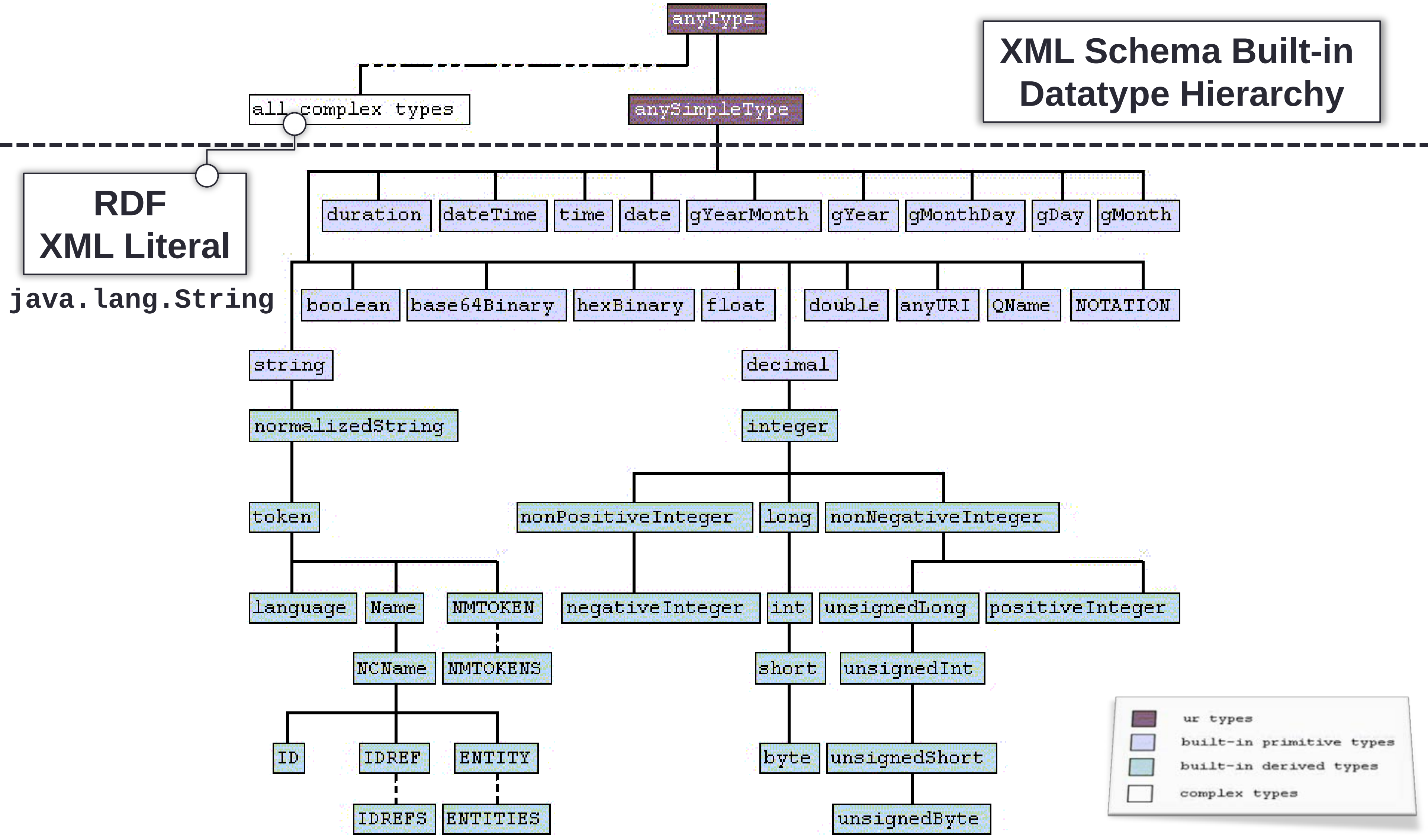
Literal boxing/unboxing

XML Schema Built-in Datatype Hierarchy

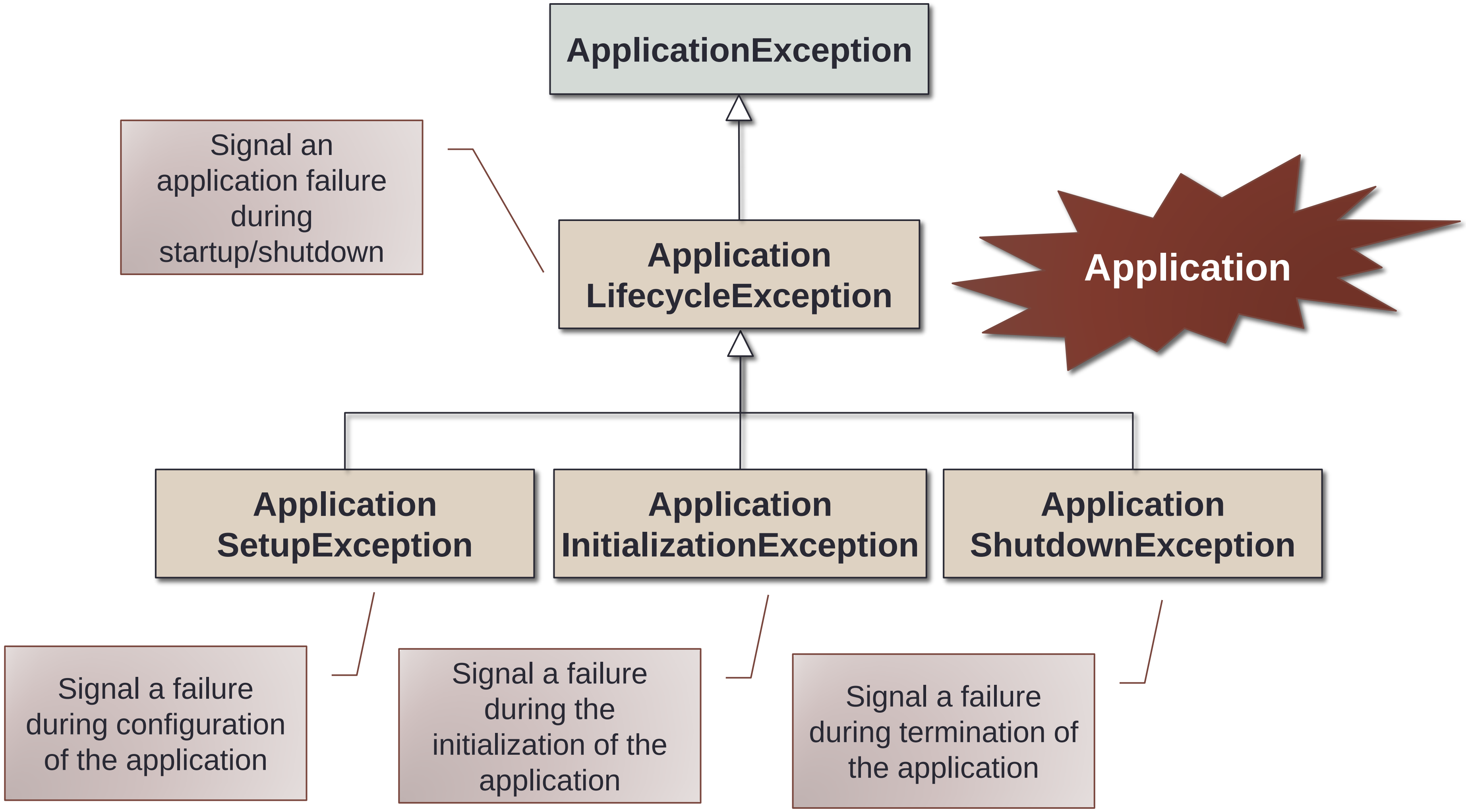


Data manipulation (II)

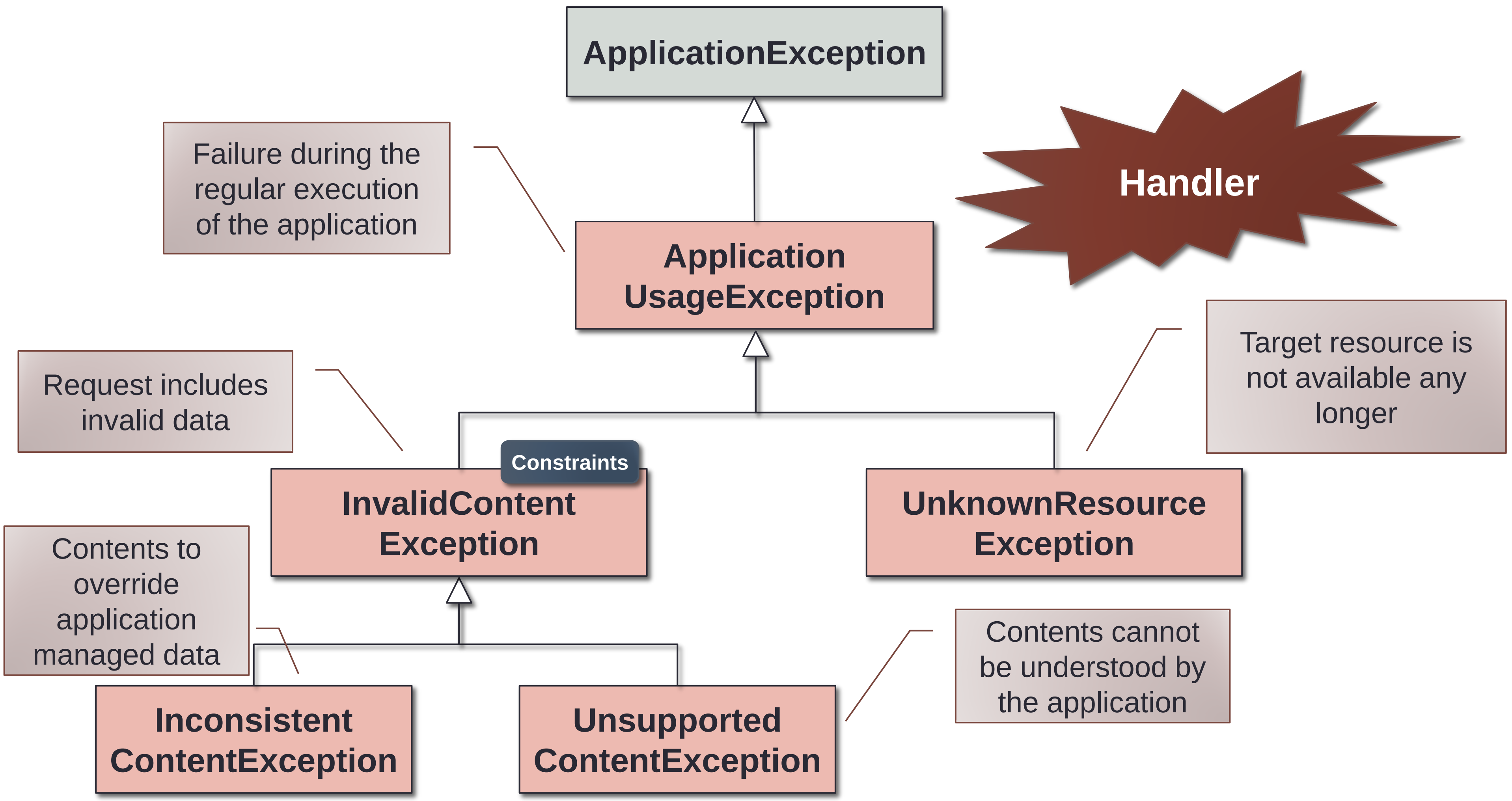
Literal boxing/unboxing



Dealing with failures (I)


ApplicationException
Signal an application failure during startup/shutdown
Application LifecycleException
Application
Application SetupException
Application InitializationException
Application ShutdownException
Signal a failure during configuration of the application
Signal a failure during the initialization of the application
Signal a failure during termination of the application

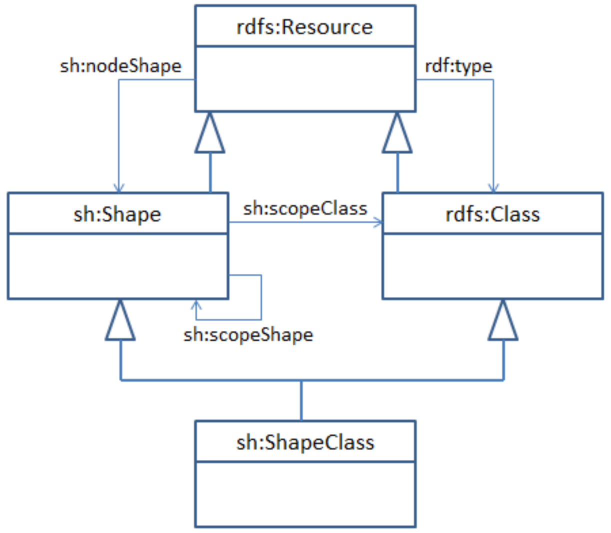
Dealing with failures (II)

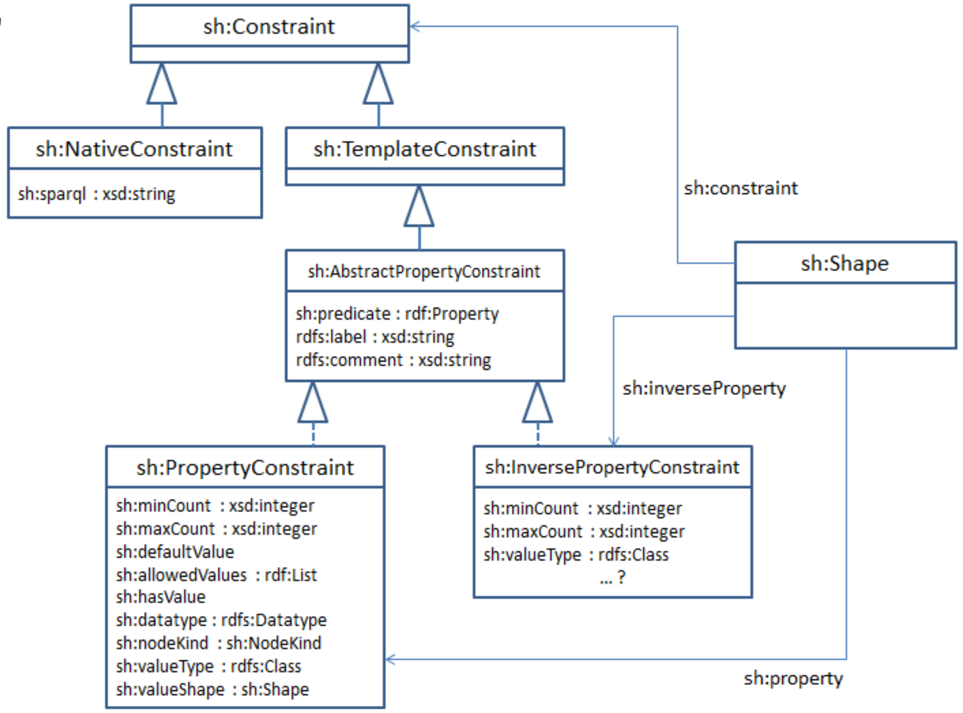

ApplicationException
Handler
Failure during the regular execution of the application
Application UsageException
Request includes invalid data
Target resource is not available any longer
Constraints
InvalidContent Exception
UnknownResource Exception
Contents to override application managed data
Contents cannot be understood by the application
Inconsistent ContentException
Unsupported ContentException

Dealing with failures (III)

Expressing constraint failures

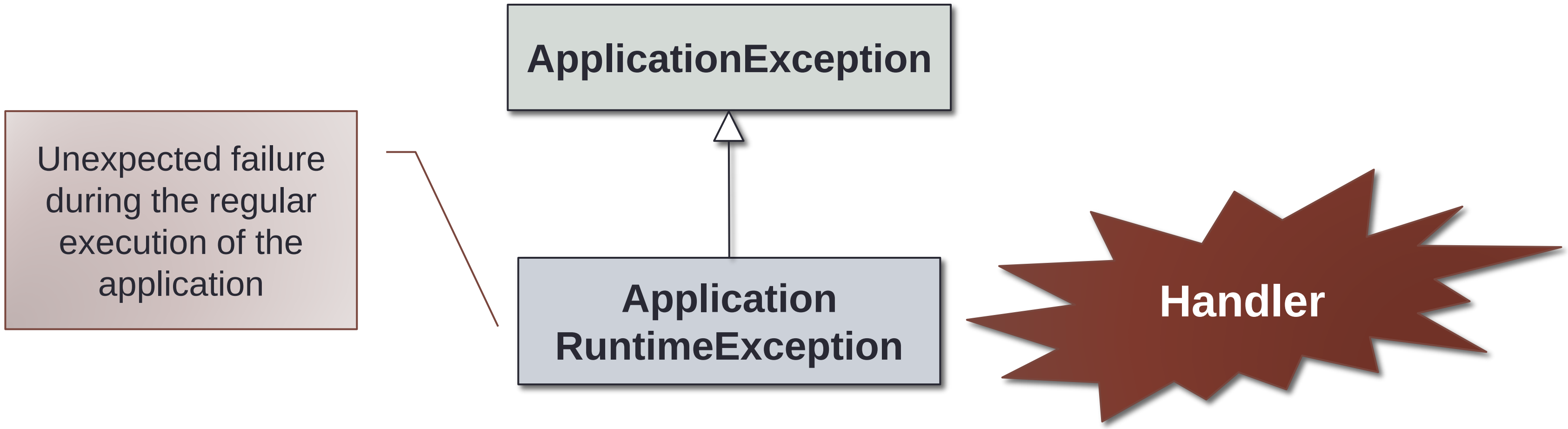
- Rely on the Shape Constraint Language (SHACL) [1] to define constraints on data set:
  - “[…] an vocabulary for describing RDF graph structures. Some of these graph structures are captured as "shapes", which group together constraints about the same RDF nodes. Shapes provide a high-level vocabulary to identify predicates and their associated cardinalities, datatypes and other constraints.”



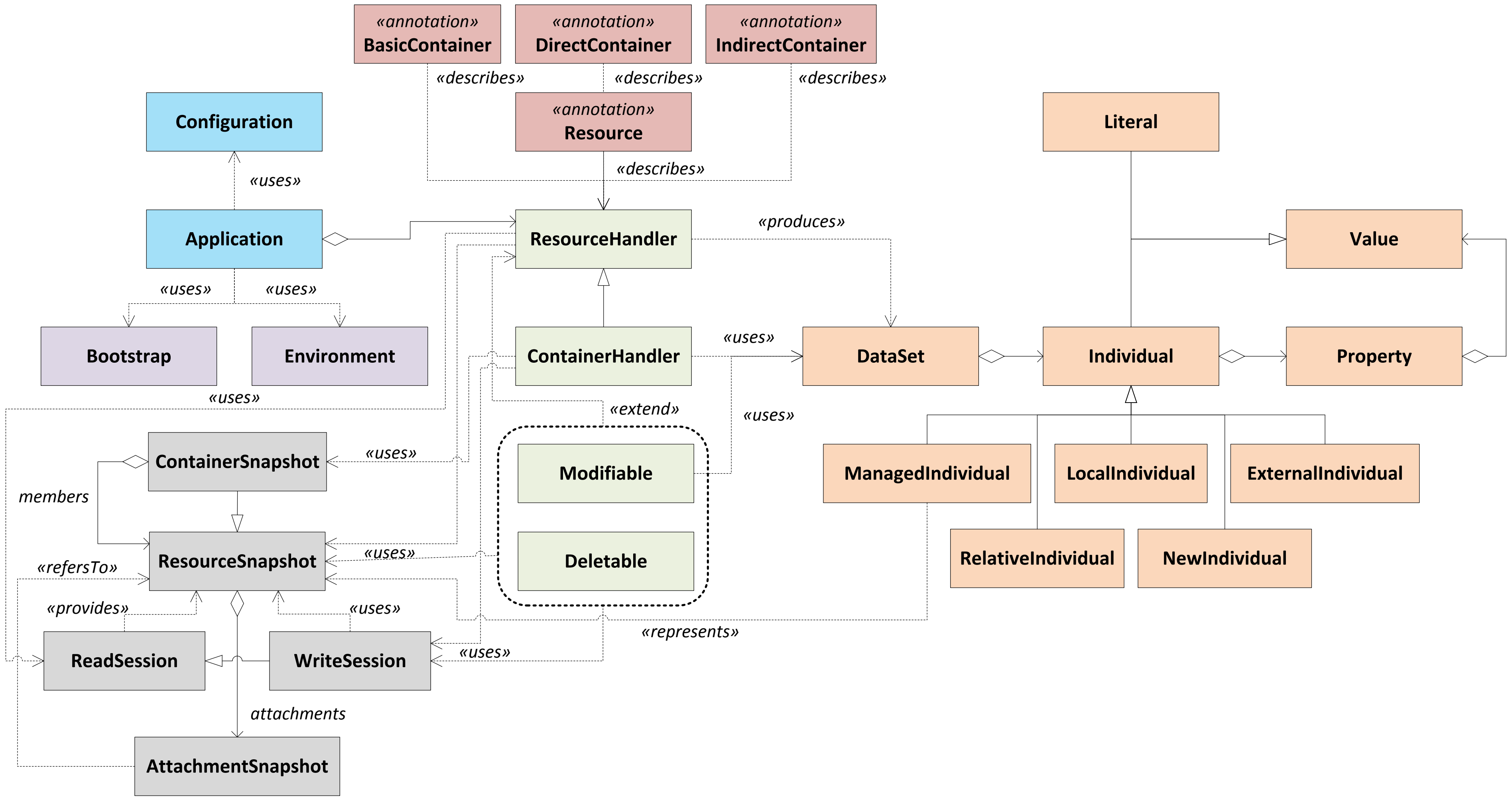

sh:Constraint
sh:NativeConstraint
sh:sparql : xsd:string
sh:TemplateConstraint
sh:constraint
sh:AbstractPropertyConstraint
sh:predicate : rdf:Property
rdfs:label : xsd:string
rdfs:comment : xsd:string
sh:Shape
sh:inverseProperty
sh:PropertyConstraint
sh:minCount : xsd:integer
sh:maxCount : xsd:integer
sh:defaultValue
sh:allowedValues : rdf:List
sh:hasValue
sh:datatype : rdfs:Datatype
sh:nodeKind : sh:NodeKind
sh:valueType : rdfs:Class
sh:valueShape : sh:Shape
sh:InversePropertyConstraint
sh:minCount : xsd:integer
sh:maxCount : xsd:integer
sh:valueType : rdfs:Class
... ?
sh:property

[1] http://www.w3.org/TR/shacl/

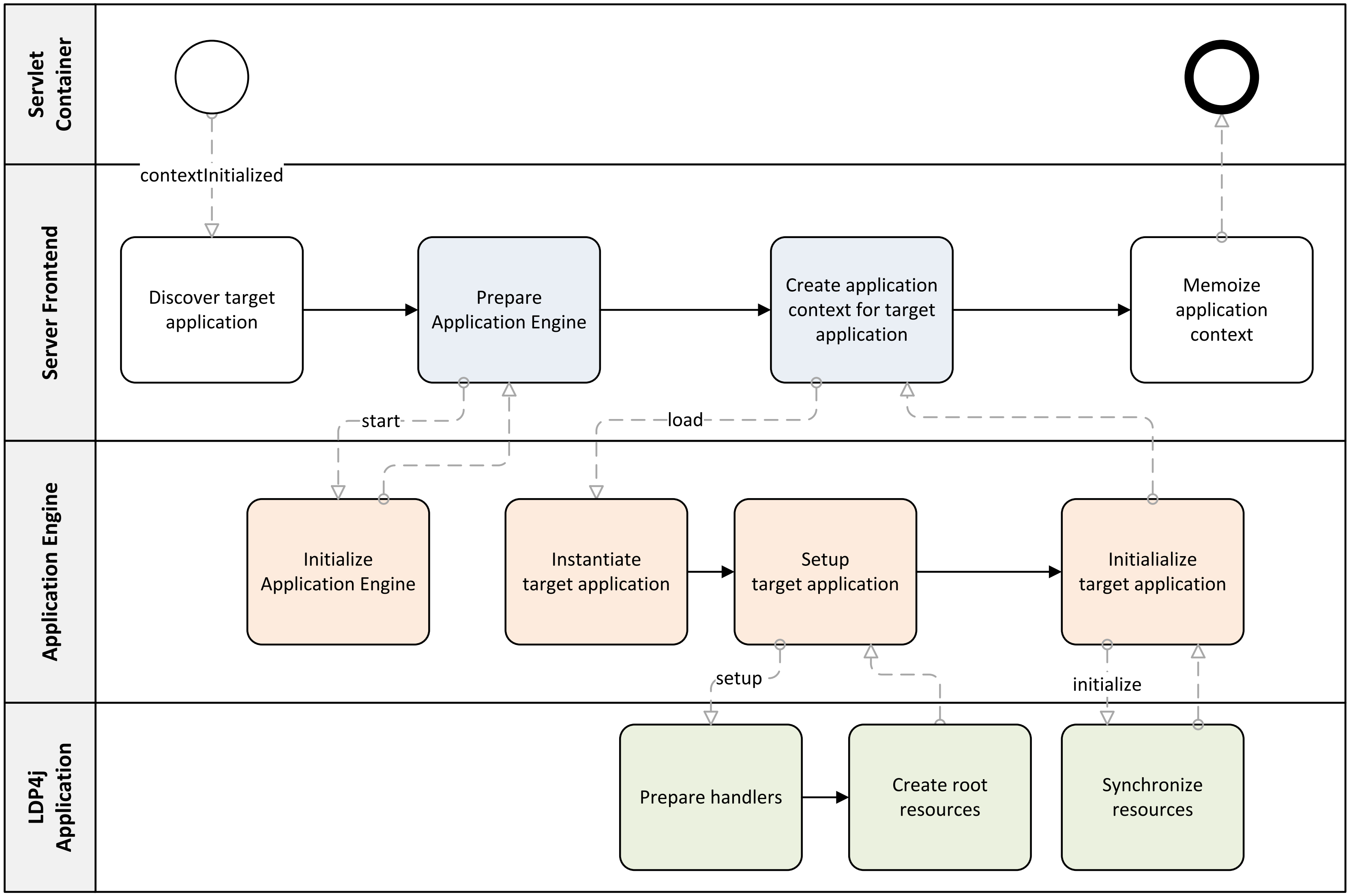
Dealing with failures (III)


ApplicationException
Unexpected failure during the regular execution of the application
Application RuntimeException
Handler

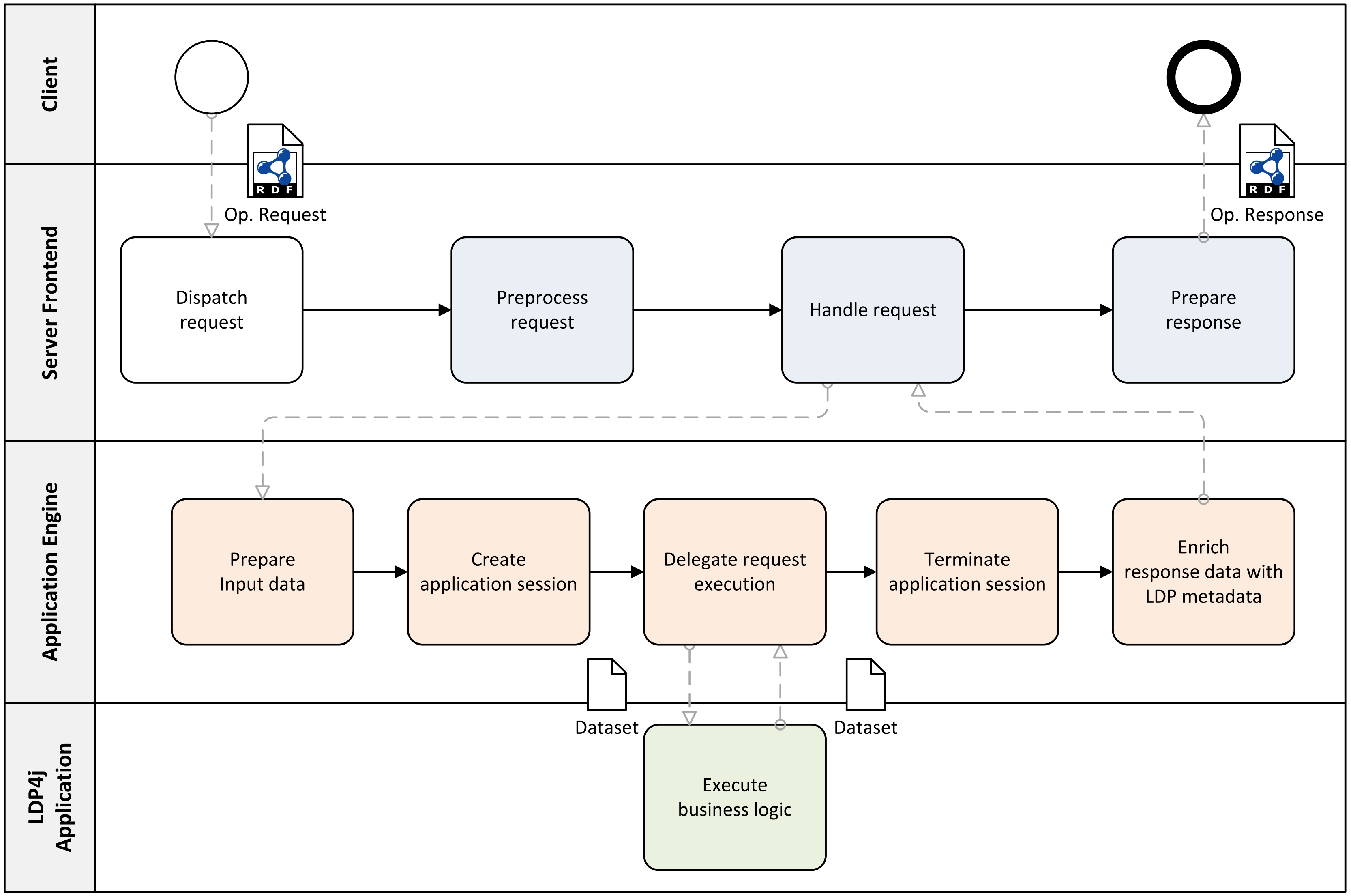
Application API in detail



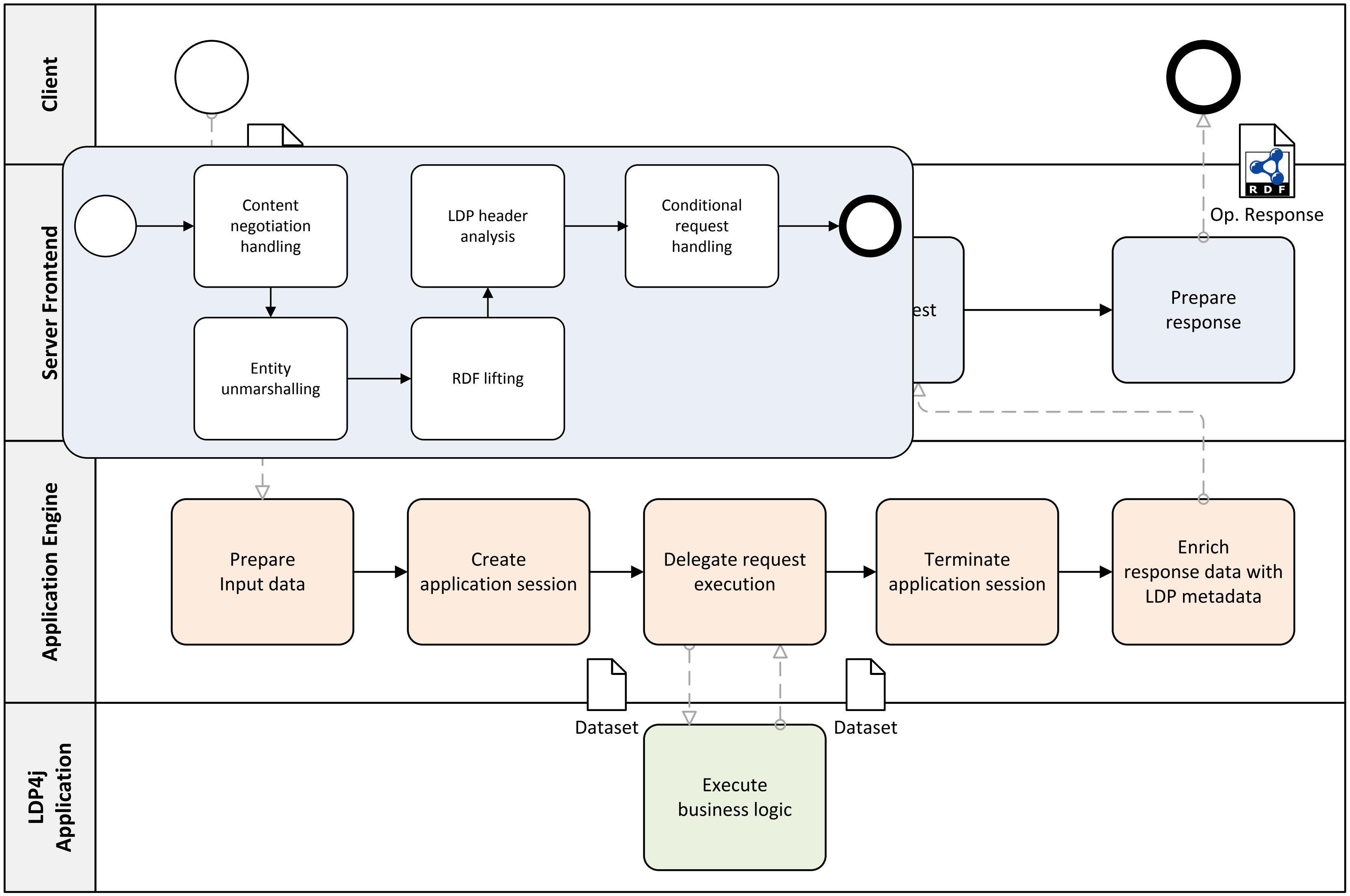
Application bootstrap process


Servlet Container
Server Frontend
Application Engine
LDP4j Application
contextInitialized
Discover target application
Prepare Application Engine
Create application context for target application
Memoize application context
start
load
Initialize Application Engine
Instantiate target application
Setup target application
Initialize target application
setup
initialize
Prepare handlers
Create root resources
Synchronize resources

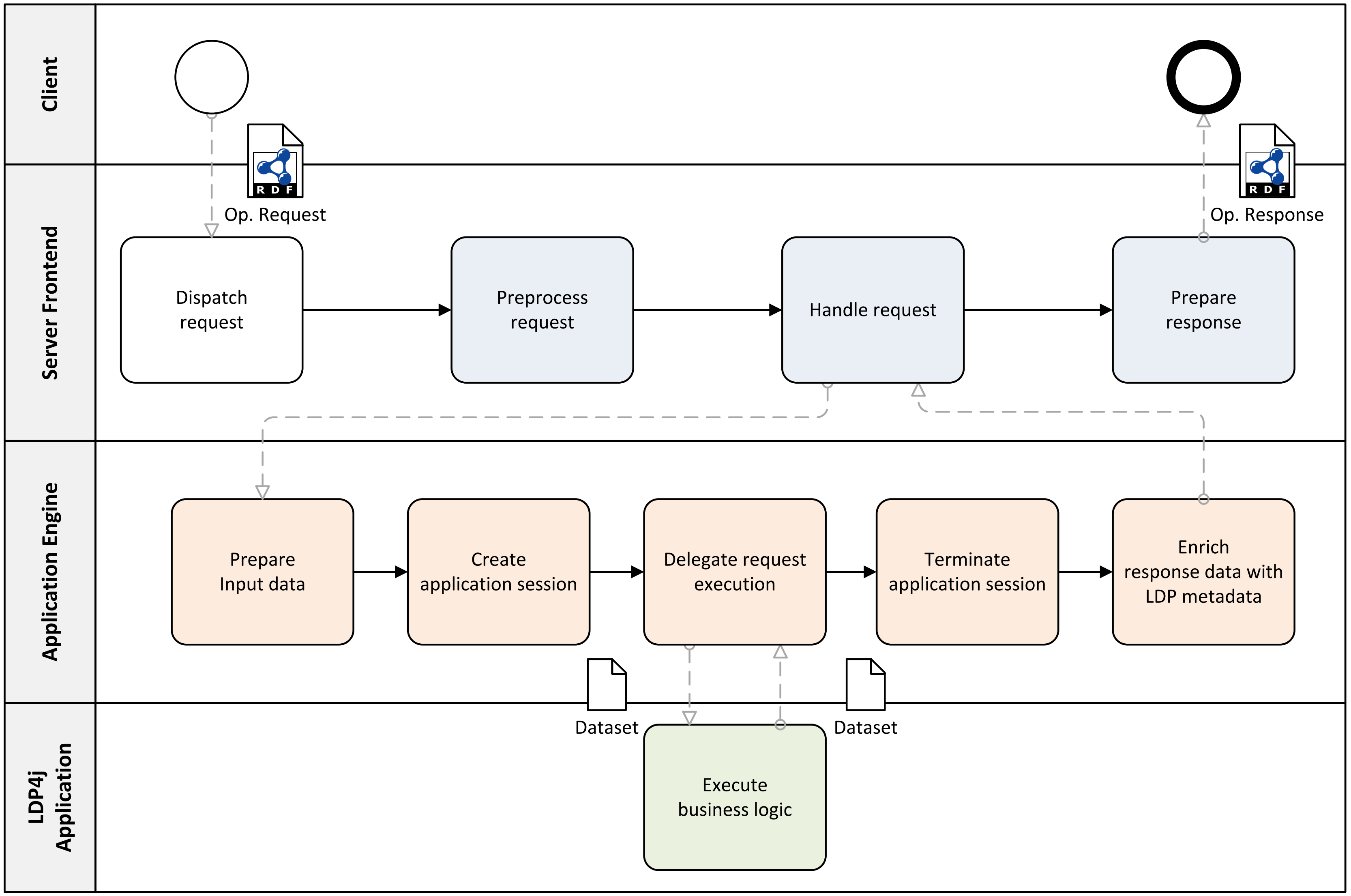
Application execution process



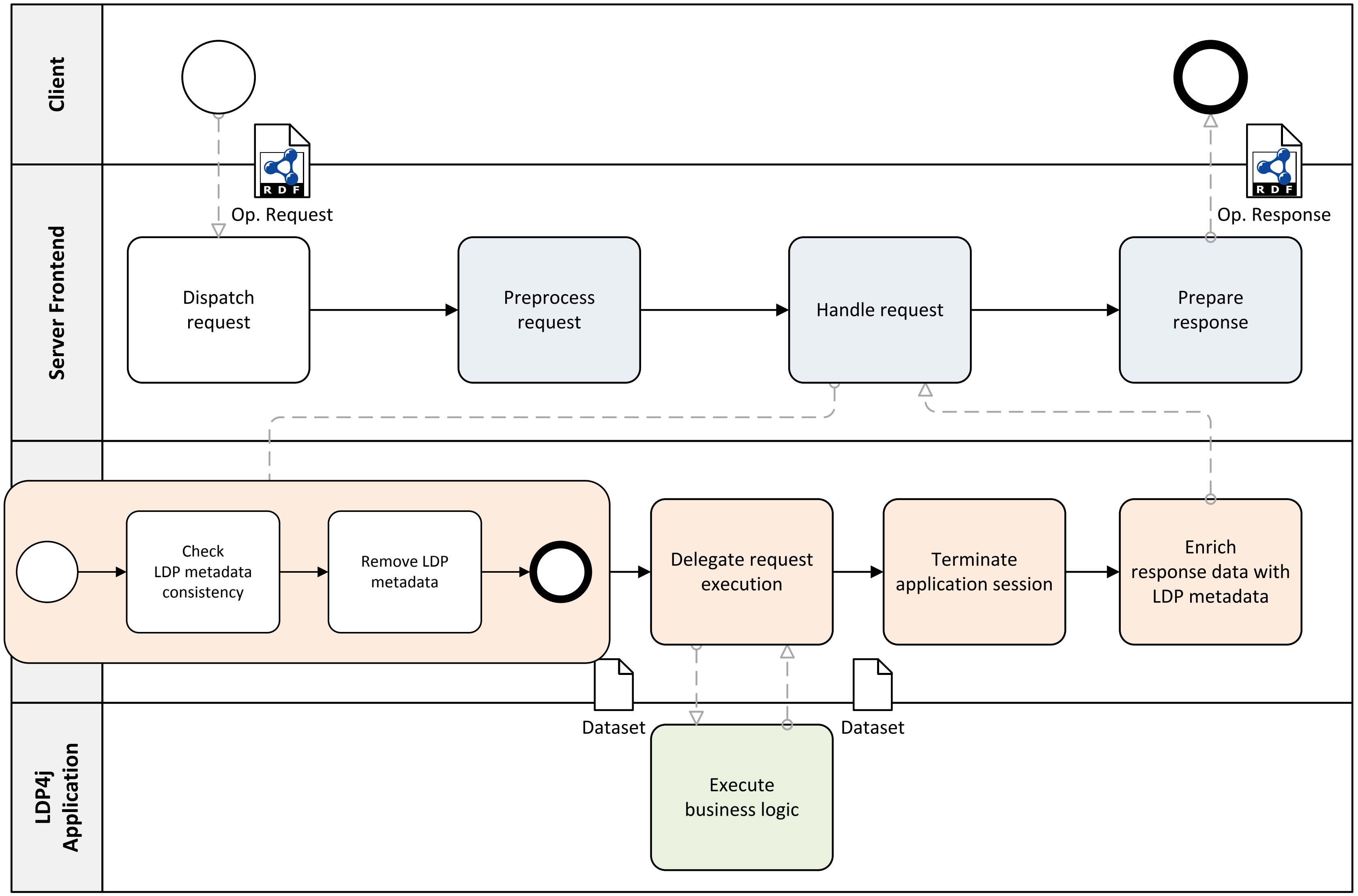
Application execution process



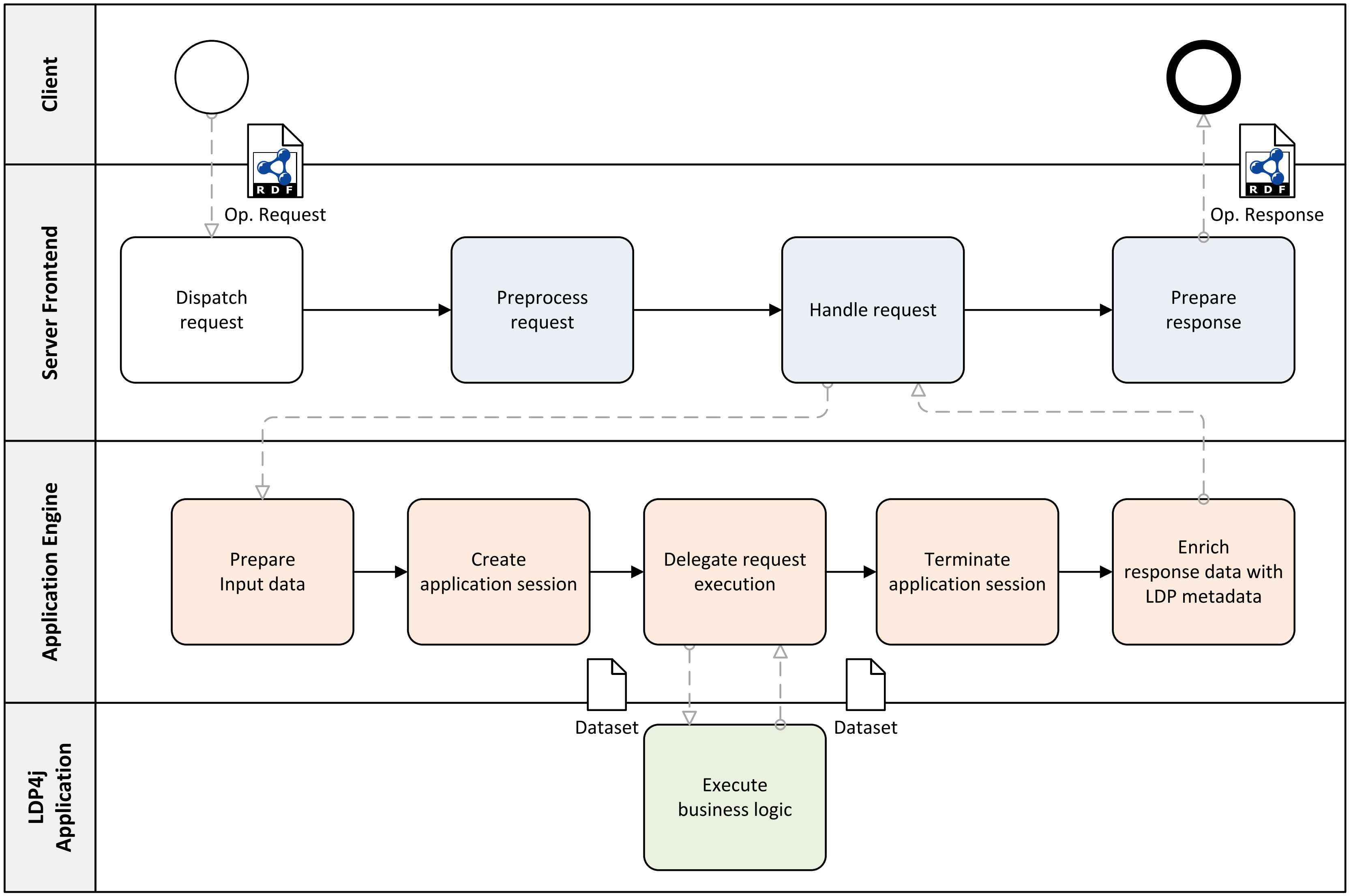
Application execution process



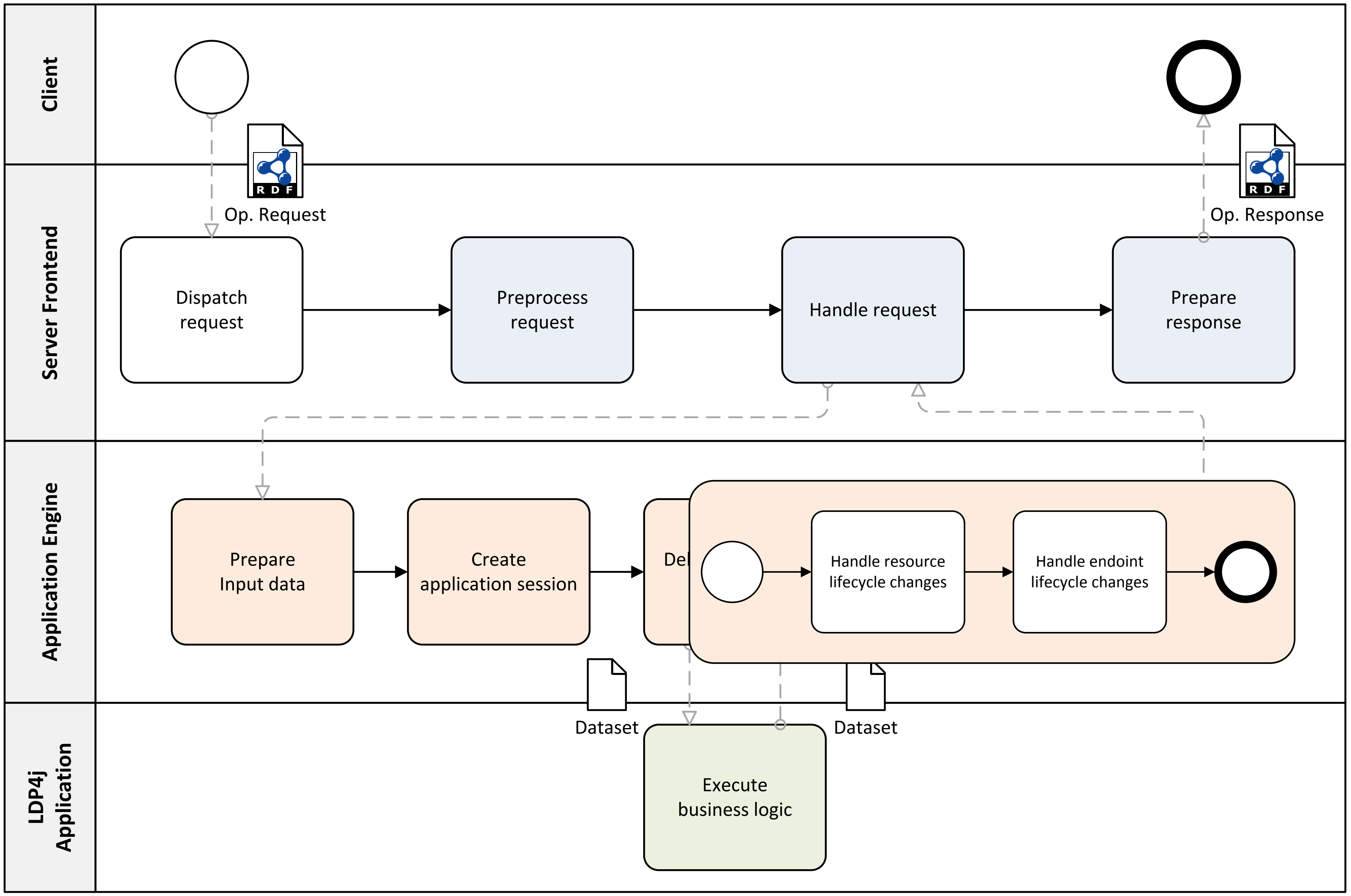
Application execution process



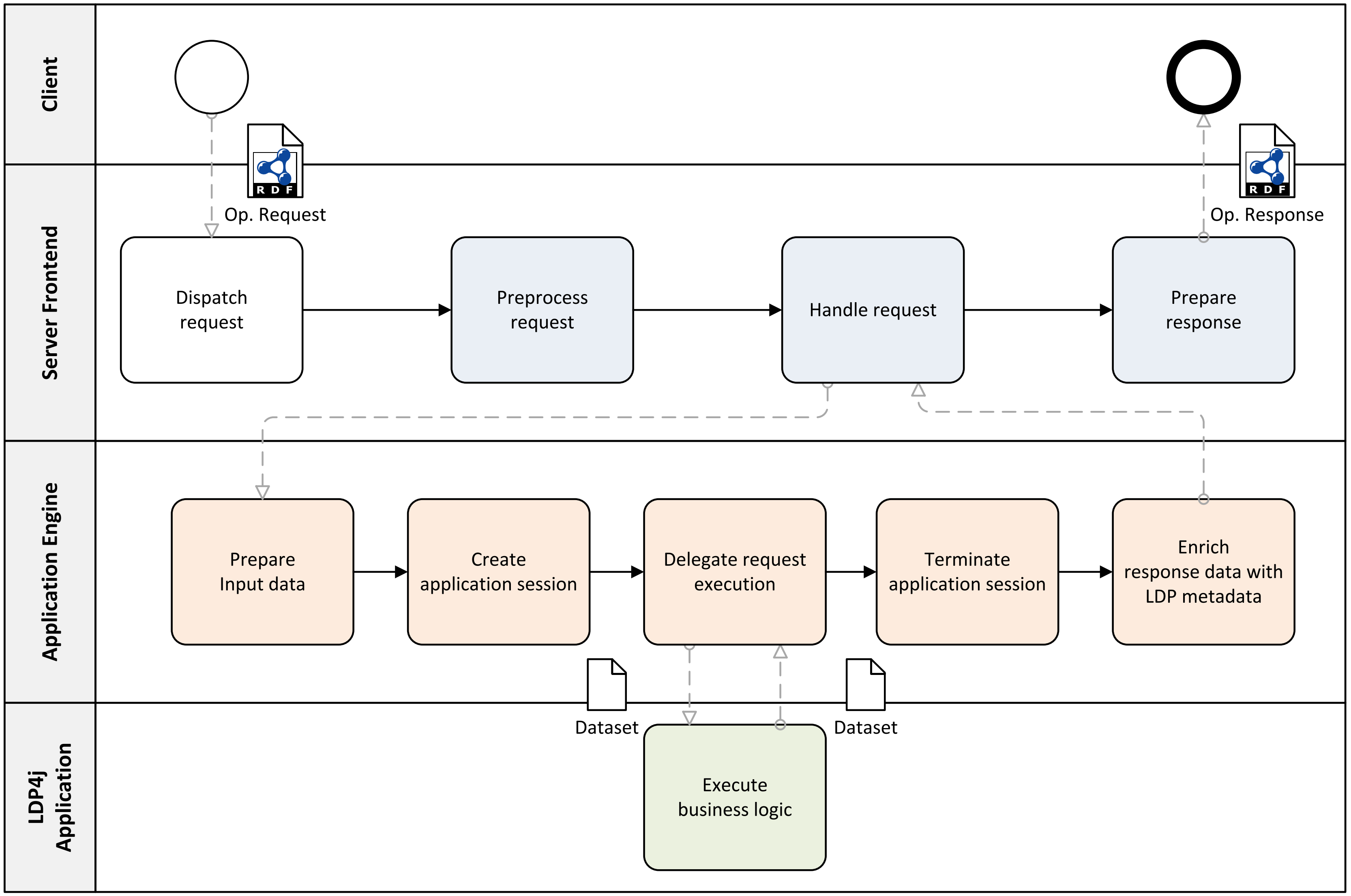
Application execution process



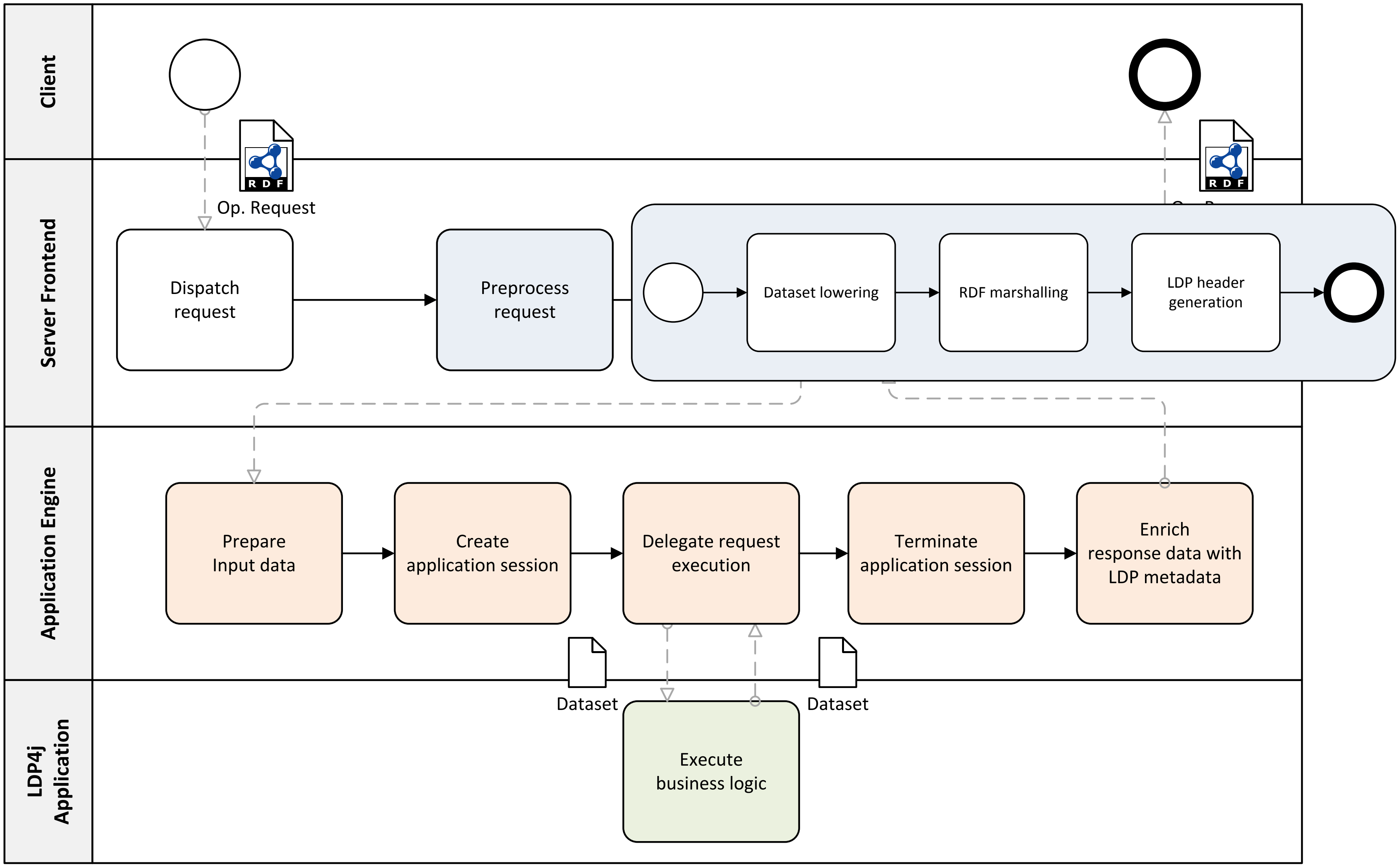
Application execution process



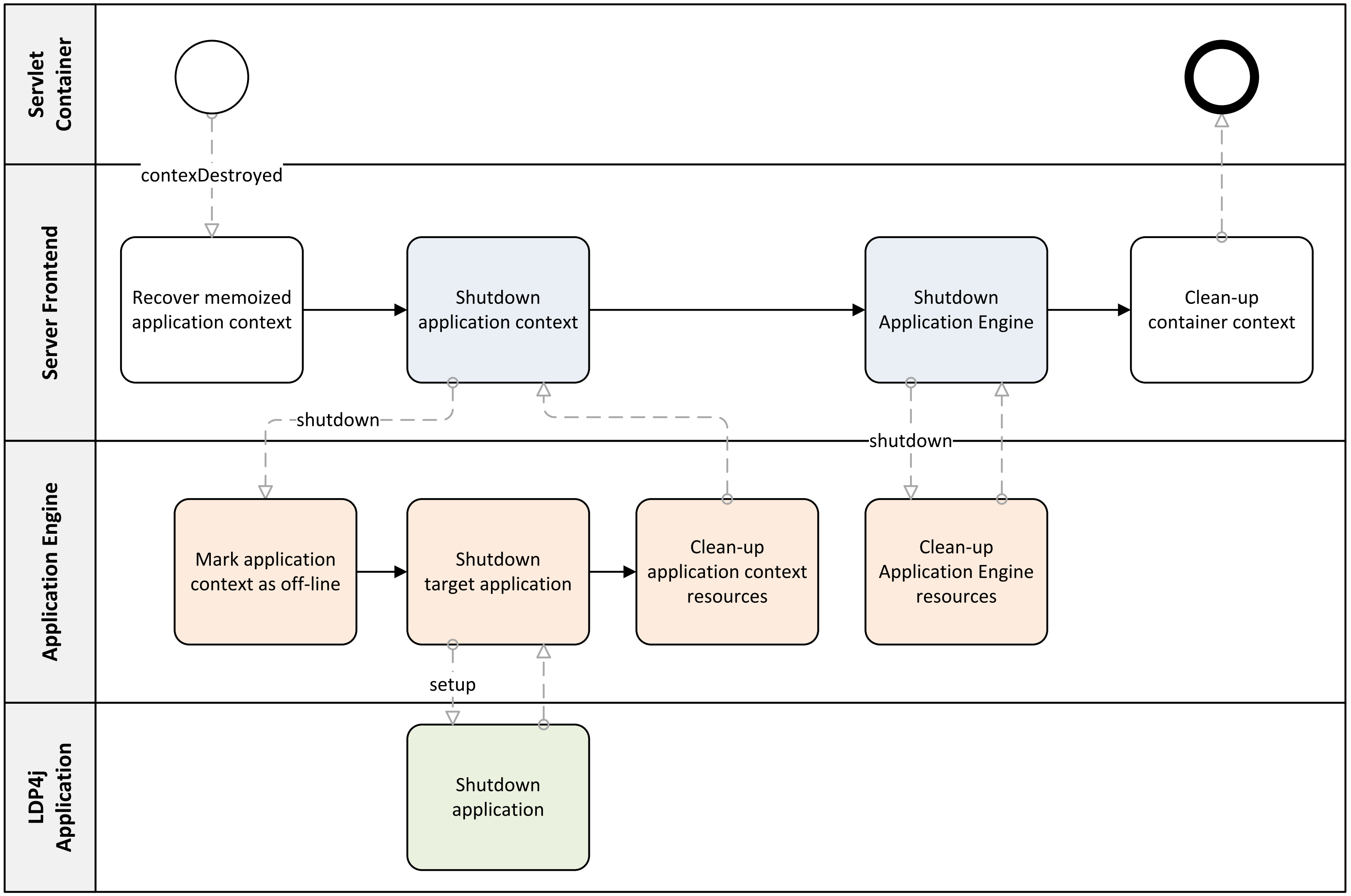
Application execution process



Application execution process



Application shutdown process


Servlet Container
Server Frontend
Application Engine
LDP4j Application
contexDestroyed
Recover memoized application context
Shutdown application context
Shutdown Application Engine
Clean-up container context
shutdown
shutdown
Mark application context as off-line
Shutdown target application
Clean-up application context resources
Clean-up Application Engine resources
setup
Shutdown application

Questions?

Miguel Esteban Gutiérrez,
Nandana Mihindukulasooriya,
Raúl García Castro,
Asunción Gómez Pérez
Center for Open Middleware / Ontology Engineering Group
Universidad Politécnica de Madrid,
Spain.