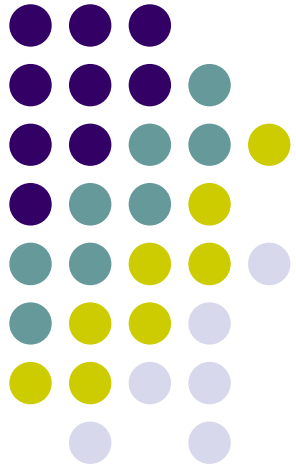
Java Integration of XQuery

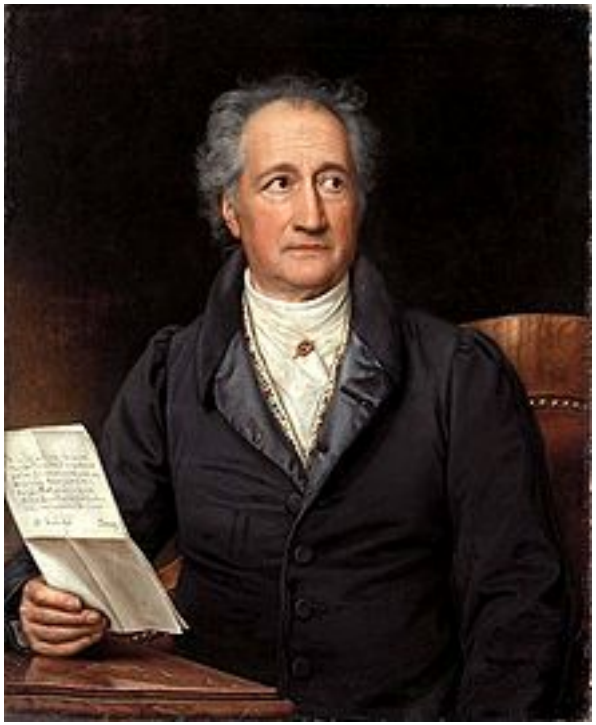
An Information Unit Oriented Approach

Presented at Balisage, August 3 – 6, 2010, Montreal, Canada

Hans-Jürgen Rennau
bits GmbH
Aachen, Germany



On Mastery



Goethe:

„In der Beschränkung zeigt sich der Meister.“

„It is in the limitation of means that mastery shows.“

The XDM

- Every value is a sequence of items
- An item is either a node or an atomic value
- Six node types (document, element, text, …)
- 45 atomic types (string, integer, date, …)

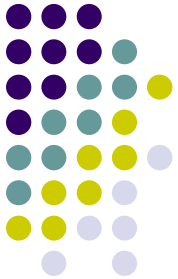
XQuery

- Everything is an expression
- An expression
  - consumes XDM values (operands)
  - generates an XDM value (expression value)

Inspired by nature?

- **Biochemistry**: *Energy* is generated, transferred and consumed in the form of ATP molecules
- **XQuery**: *Information* is generated, transferred and consumed in the form of XDM values

XQuery (continued)

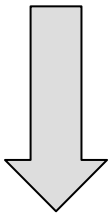


- A query‘s interface – also XDM
  - Input: context item and query parameters
  - Output: the query result

- But notice a subtle asymmetry
  - Input: a set of named XDM values
  - Output: a single XDM value

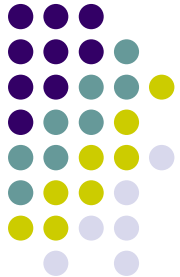
The XDM limitation – viewpoint of integration (1)

The output of a query is the input for another component integrating XQuery

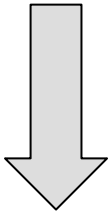


Integration calls for multiple, named output !

The XDM limitation – viewpoint of integration (2)



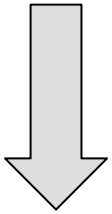
"Result = (node | atomicValue)*" - for the consumer a pointless limitation of granularity



Integration calls for non-XDM output !

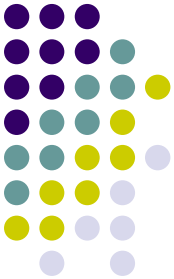
How tackle the XDM limitation?

The need arises from an integration point of view



Need must be satisfied by the integration infrastructure / API

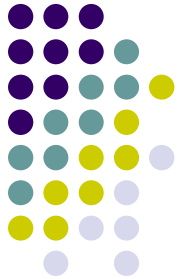
Proposal: two-layer API



- First-layer API delivers the literal result
- Second-layer API
  - Processes the literal result…
  - … producing a final result which
    - Associates partial results with names
    - Transforms items (XDM-defined granularity) into information units (API-defined granularity)

Literal result = one-to-one representation of the result sequence

XQJPLUS



XQJ

- The standard Java API of XQuery
- A first-layer API

XQJPLUS

- Built on XQJ
- XQJ implementation: Saxon processor
- A first-layer API (convenience wrapper around XQJ)
- A second-layer API
- A code generator

XQJPLUS – an example

Input: Water-quality data from USGS Web Services

„Description of measurement activities “

Output: Report with two named parts

Part name	*Part data type*	*Comment*
projects	SortedSet<String>	List of project names
quantities	Map<String,String[]>	Substance name → quantities

Glance at the input data…

```
<WQX xmlns="http://qwwebservices.usgs.gov/schemas/WQX-Outbound/2_0/">
   <Organization>
      <Activity>
         <ActivityDescription>
            <ProjectIdentifier>860700319</ProjectIdentifier>
         </ActivityDescription>
         <Result>
            <ResultDescription>
               <CharacteristicName>Depth</CharacteristicName>
               <ResultMeasure>
                  <ResultMeasureValue>0.60</ResultMeasureValue>
                  <MeasureUnitCode>ft</MeasureUnitCode>
               </ResultMeasure>
            </ResultDescription>
         </Result>
         <Result>
            <ResultDescription>
               <CharacteristicName>Nitrite</CharacteristicName>
               <ResultMeasure>
                  <ResultMeasureValue>0.020</ResultMeasureValue>
                  <MeasureUnitCode>mg/l</MeasureUnitCode>
               </ResultMeasure>
            </ResultDescription>
         </Result>
         <Result>
```

A glance at the results

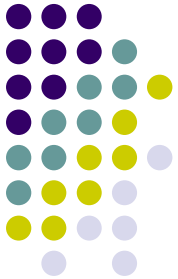
```
================================================
*** Part #1: projects
================================================
860700319
860700379
NAWQA

================================================
*** Part #2: quantities
================================================
Nitrate=
   0.28 [mg/l as N]
   0.36 [mg/l as N]
   0.37 [mg/l as N]
   0.40 [mg/l as N]
   0.41 [mg/l as N]
   1.24 [mg/l]
   1.58 [mg/l]
   1.66 [mg/l]
   1.77 [mg/l]
   1.82 [mg/l]
Algae, floating mats (severity)=
   0.0 [code]
Phosphorus=
   0.009 [mg/l]
   0.018 [mg/l]
   …
…
```

Java client code: three lines

```
…
// *** execute query
// ================
InfoTray tray = xq.execQuery2InfoTray(queryName, …);

// *** read results
// ================

SortedSet<String> projects =
   tray.getStringSortedSetObject("projects");

Map<String, String[]> quantities =
   tray.getMapString2StringsObject("quantities");
```

I get exactly **what** I need, and **how** I need it – no detours!

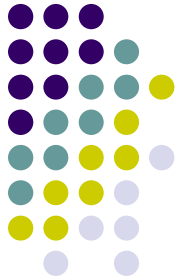
Concept: Information unit



- Convenient entity for reading, processing, delivering information

- A Java object; examples:
  - String, String[]
  - Element, Element[]
  - Map<String,Node> , Map<String,Node>[]

Concept: info tray



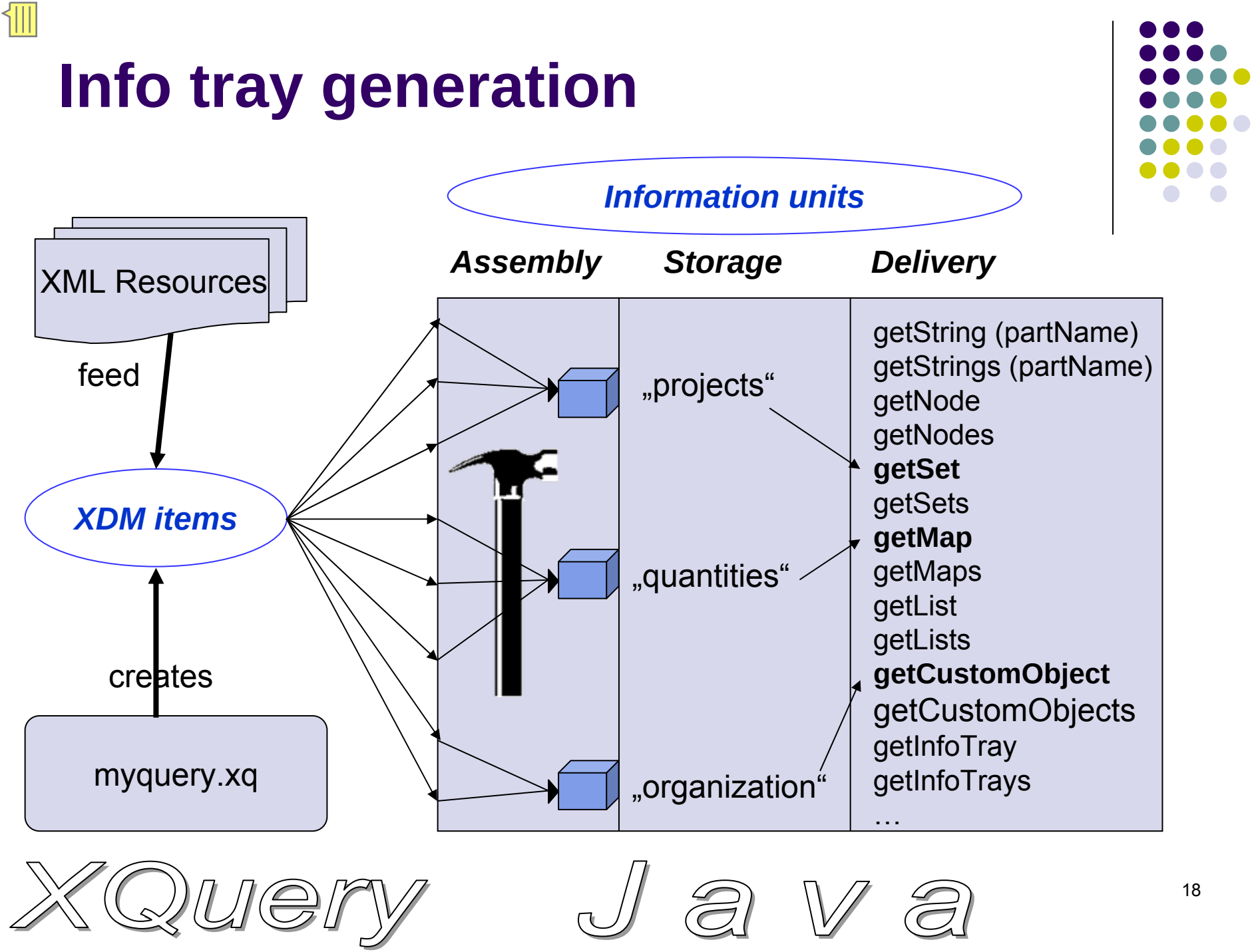
- Generic collection of information units
- Resembles a map: units are read by name
- Resembles a tree: units can be nested

Info tray – the interface

Name-based read methods:

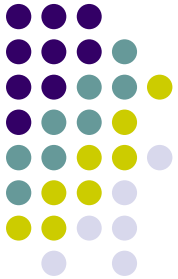
```
String                  getString  (String partName);
String[]                getStrings (String partName);

Node                    getNode    (String partName);
Node[]                  getNodes   (String partName);

Map<String,String>      getMapStringToStringObject  (String partName);
Map<String,String>[]    getMapStringToStringObjects (String partName);

InfoTray                getInfoTray  (String partName);
InfoTray[]              getInfoTrays (String partName);
…                       …
```

Info tray generation



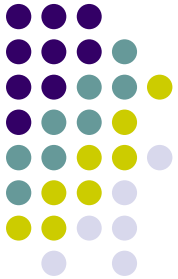
XQuery Java

XQuery code providing the data



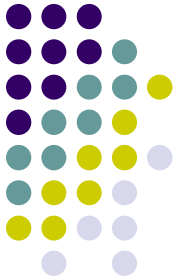
```
declare default element namespace "http://qwwebservices.usgs.gov/schemas/WQX-
Outbound/2_0/";

(: get project identifiers :)
for $n in distinct-values(//ActivityDescription/ProjectIdentifier) order by $n
return $n,

(: get substance-to-quantities map :)
<substances>{
   let $nameElems := //CharacteristicName[../ResultMeasure/ResultMeasureValue]
   for $name in distinct-values($nameElems)
   order by $name
   return
      <substance name="{$name}">{
         for $value in $nameElems[. eq $name]/../ResultMeasure/ResultMeasureValue
         order by number($value)
         return <quantity>{$value/concat(., ' [', ../MeasureUnitCode,
']')}</quantity>
      }</substance>
}</substances>
```

XQuery code adding *control information*



```
declare default element namespace "http://qwwebservices.usgs.gov/schemas/WQX-
Outbound/2_0/";

(: get project identifiers :)
<xqjp:part name="projects" type="string_sortedset_object"/>,
for $n in distinct-values(//ActivityDescription/ProjectIdentifier) order by $n
return $n,

(: get substance-to-quantities map :)
<xqjp:part name="quantities" type="map_string_to_strings_object"
           entryPath="*" keyPath="@name" valuePath="*"/>,
<substances>{
   let $nameElems := //CharacteristicName[../ResultMeasure/ResultMeasureValue]
   for $name in distinct-values($nameElems)
   order by $name
   return
      <substance name="{$name}">{
         for $value in $nameElems[. eq $name]/../ResultMeasure/ResultMeasureValue
         order by number($value)
         return <quantity>{$value/concat(., ' [', ../MeasureUnitCode,
']')}</quantity>
      }</substance>
}</substances>
```

XQuery result
= data items + control items



- Control items define information units
- Data items supply the data input

```
<xqjp:part
        name="quantities"
        type="map_string_to_strings_object"
        entryPath="*"
        keyPath="@name"
        valuePath="*"/>
```

Unit name and type

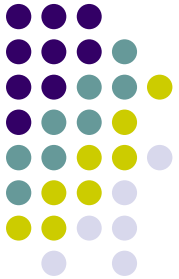
Control assembly

Control item

```
<substances>
   <substance name=„Alkalinity">
      <quantity>245 [mg/l CaCO3] </quantity>
      <quantity>245 [mg/l CaCO3] </quantity>
   </substance>
   <substance name=„Ammonia and ammonium“>
      <quantity>0.062 [mg/l as N] </quantity>
      <quantity>0.08 []mg/l NH4] </quantity>
   </substance>
</substances>
```

Data item

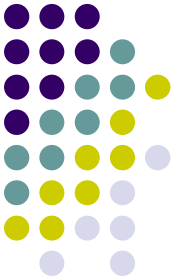
The assembly process



- The assembly process is controlled by the XQuery result
- The XQuery result consists of control items and data items; each item is either a pure control item or a pure data item
- The input for an information unit is (usually) one control item defining the unit, followed by one or more data items:

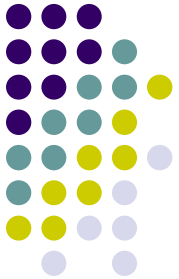
  CDDD-CD-CD-CDD-CDDDDDDDDDDDDD-CD

- There are also other types of control items – e.g. assigning data items to variables, or defining namespace context



Example summarized

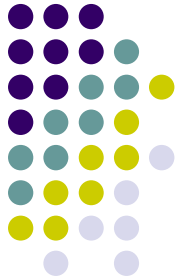
- Perspective #1: the API client (Java)
  - Receives an info tray and picks from it information units
- Perspective #2: the query (XQuery)
  - Produces data items, accompanied by control items
- Perspective #3: the API implementation (Java)
  - Constructs the info tray, directed by control items



Extending the concept –

tray schema & info shape

Info tray - problems



Info tray is convenient but allows usage errors:

- Wrong unit name
- Wrong combination of unit name and unit type

Wrong name!

```
String[] projects = tray.getStrings(„projeects“);

Node[] = tray.getNodes(„quantities“);
```

Wrong type!

Tray schema - example

```
<ts:traySchema xmlns:ts="http://www.bits-ac.com/xqjplus/traySchema"
   xmlns:java="http://www.bits-ac.com/xqjplus/traySchema/java-binding">

   <!-- ============== -->
   <!-- tray structure -->
   <!-- ============== -->
   <ts:tray>
      <ts:part name="projects" type="string_sortedset_object" />
      <ts:part name="quantities" type="map_string_to_strings_object" />
   </ts:tray>

   <!-- ============ -->
   <!-- Java binding -->
   <!-- ============ -->
   <!-- implementation class -->
   <java:trayBinding className="QuantityReport"
                     package="com.bits_ac.xqjplus.appl" />
</ts:traySchema>
```

Tray schema

- An XML document describing a specific type of info tray
  - Names and types of information units
  - Constraints (cardinality and the use of meta data)
  - Code generation control

- Purposes
  - Documentation
  - Validation `infoTray.validate (traySchemaFile)`
  - Code generation

Code generation - principle

Unit names shifted into method names

- Tray

Method name generic

```
SortedSet<String> getStringSortedSetObject(String partName);
```

- Shape

```
SortedSet<String> getProjects();
```

Method name tray-specific

The result – an info shape

Generated from tray schema:

```
public class QuantityReport implements LoadableFromInfoTray, QuantityData {
   protected InfoTray dataSource;

   public void loadFromInfoTray(InfoTray source) {
      dataSource = source;
   }

//data access
   public SortedSet<String> getProjects() throws XQPException {
      return dataSource.getStringSortedSetObject("projects");
   }

   public Map<String,String[]> getQuantities() throws XQPException {
      return dataSource.getMapString2StringsObject("quantities");
   }
…
}
```

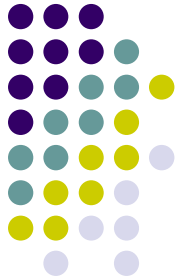
The shape is backed by a tray

Specific getters delegate to generic getters

Java code using info shape

```
…
// *** execute query
// ================
QuantityReport shape = xq.execQuery2InfoShape(queryName, …);

// *** read results
// ===============

SortedSet<String> projects = shape.getProjects();

Map<String, String[]> quantities = shape.getQuantities ();
```

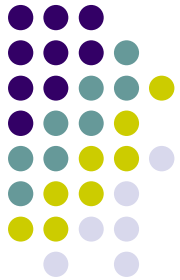
Mm - compiler-guarded access to structured XQuery results!



Adding details –

a closer look at information units

Information unit – properties



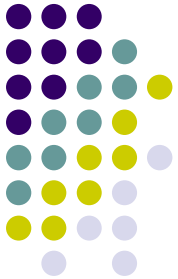
Q: Which Java types are possible?
A: list of physical types

- Data content (Java object)
- Name (QName)
- Meta data (name-value pairs)
- Type information (three QNames)

Information unit – data types

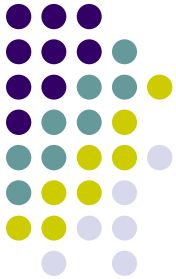
Category	Types (examples)
Nodes	Element, Document
Atomic values	String, Integer, Float, Duration
Maps	Map<String,String>, Map<String,Map<String,String>>
Collections	Set<String>, List<Node>
Miscellaneous	java.util.Properties
Custom types	mypackage.MyType
InfoTray	InfoTray, InfoTray[]

Information units can be trees, and even forests



- Data type *InfoTray* → a tree of units
- Data type *InfoTray[]* → a forest of units
- The leaves of such trees are information units – can be documents, maps, collections …

Information units can be custom objects



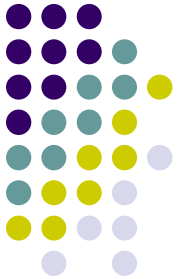
- Custom types must implement
  `void loadFromXML(Node xmlData)`

- Query control item specifies type name

```
<xqjp:part name="activityResults"
                type="custom_objects"
                customType="com.bits_ac.xqjplus.appl.UsgsResult" />,
```

- Query data item provides the input data

Data types - summary



- Currently: 70 data types supported
- All data types in pairs – single object & array
- Besides standard types also custom types
- Units can be trays, so units can be nested

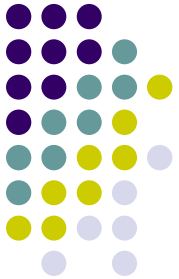
Meta data

- XQuery control items can assign metadata to an information unit

```
<xqjp:part name="projects"
           type="string_sortedset_object"
           k:crdate="20100715"
           k:department="AC01">
   <im:source><![CDATA[729<<40172”]]></im:source>
</xqjp:part>,
```

- Java reads them from the tray

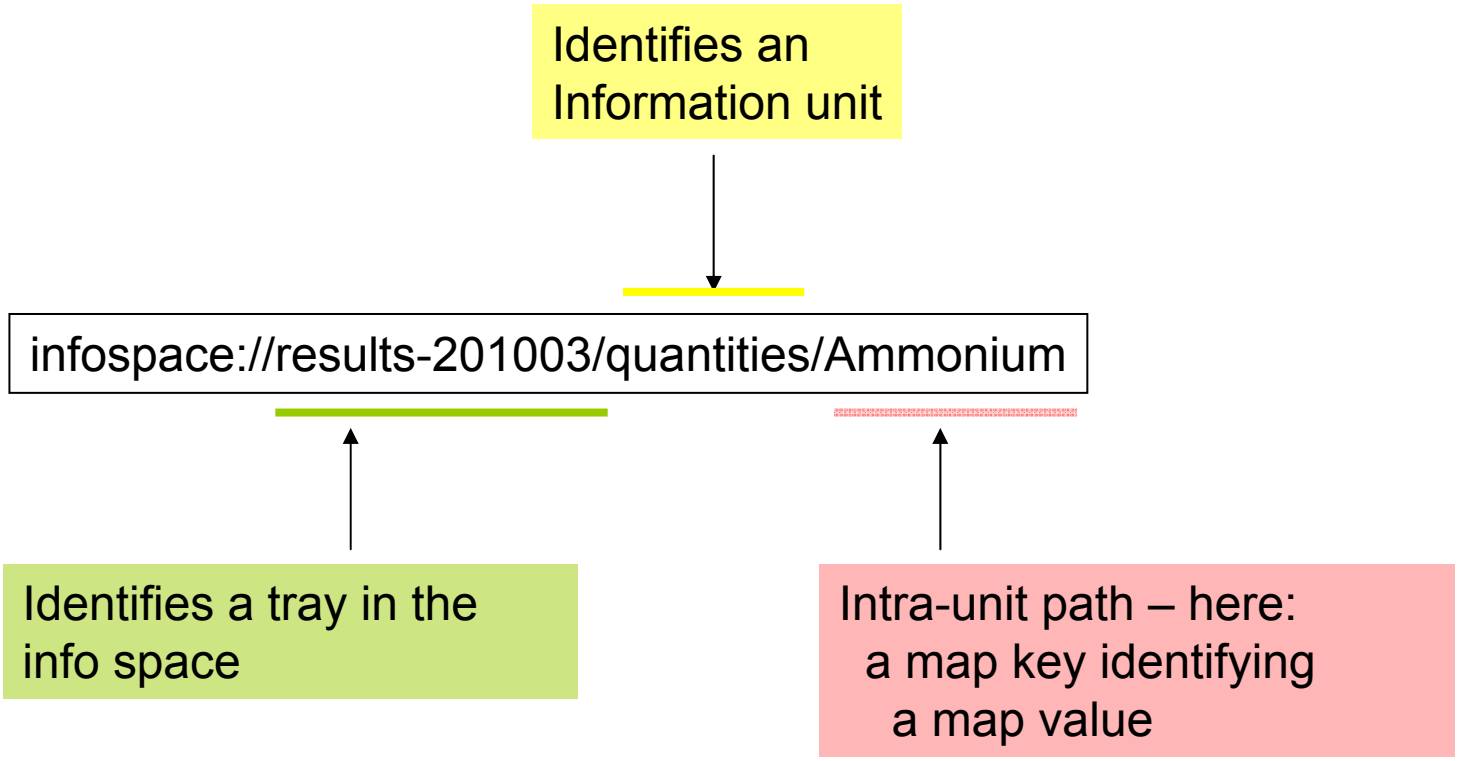
```
Map<String,String> meta = tray.getPartMeta("projects");
System.out.println(meta.get("{www.kant.de/ns}crdate"));
     // → 20100715
```



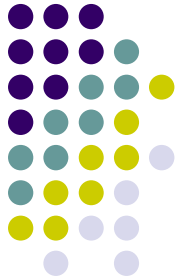
Data integration –

info space & info path

Example of an info path

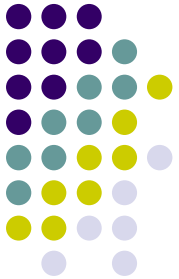


The concept of an info path



Evaluation context: info space (a set of named info trays)

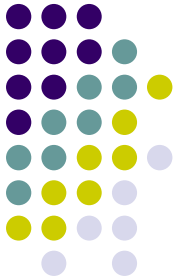
1. Within space: tray identified by a QName
2. Within tray: a unit identified by QNames(s)
3. Within unit: data identified by intra-unit path



Integration of technology –

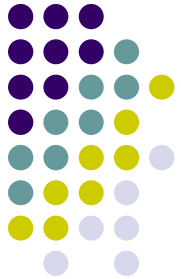
based on info tray

Info tray – independent of XQuery



- XQuery can be used to construct info trays
- Their use is independent of how they were constructed
- Other modes of construction are possible
  - Using other constructor languages (e.g. XSLT)
  - Building units „by hand“

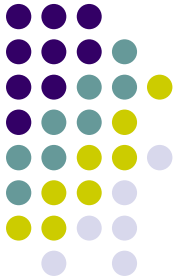
A new design pattern for general purpose languages?



- Pure information processing
  - Side effect free
  - Determined by immutable information sources

- W3C languages designed for this purpose: XSLT & XQuery

- Design pattern:
  1. Factor out pure information processing
  2. Delegate it‘s execution to specialists (XQuery, …)
  3. Pick up the results and go on…

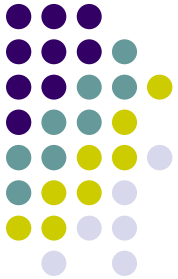


Integration of skills –

based on tray schema

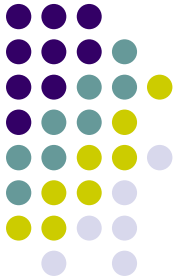
The cooperation between Java and XQuery developers

The cooperation between Java and XQuery developers

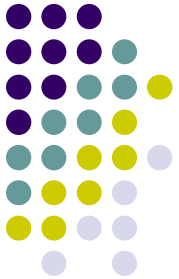


1. They discuss the information to be created
2. They formalize the result as a **tray schema**
3. Tray schema is compiled into an info shape
4. ***XQuery dev.*** writes a prototypic query
5. ***Java dev.*** starts to code against info shape
6. ***XQuery dev.*** incrementally completes query

XQJPLUS - summary



- Integrates XQuery into Java, supporting free granularity and name-based delivery
- Defines and implements six key concepts:
  - Information unit and info tray
  - Tray schema and info shape
  - Info space and info path
- A structured approach to integrate information processing into host languages
- Might be modified by replacing host language (Java) or processing language (XQuery) by other languages



Thank you!