Collage

Collage: IBM's declarative Programming Model for Compositional Development

Jaroslav Pullmann
FIT Fraunhofer

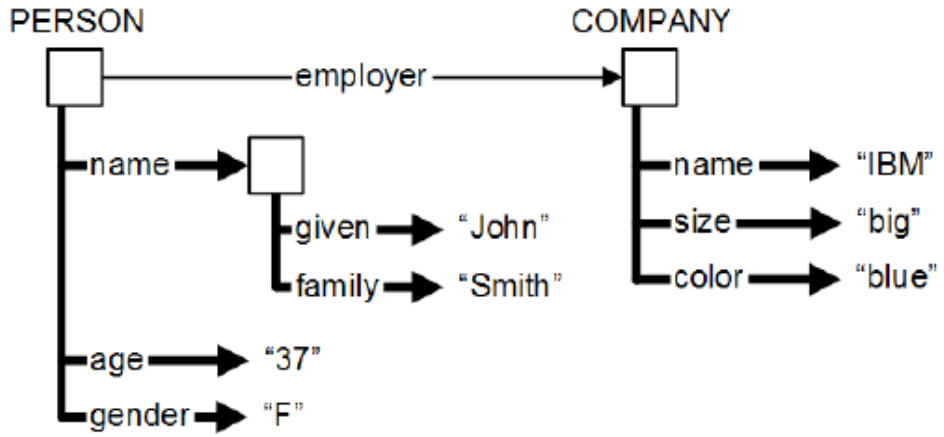
Fraunhofer Institut
Angewandte
Informationstechnik

RDF Data Model

- Unifying, distributed data model
- Resolvable via URLs
- RDF Classification
  - Explicitly through rdf:type assignment
  - Multiple classification
  - Dynamic classification: may change
  - Classifications may originate from disparate development (re)sources

RDF Data Model

– Separate relation graph and value tree
  - via specific subproperty “c:value”
  - XPath addressing
  - correspond to relationships within and between resources

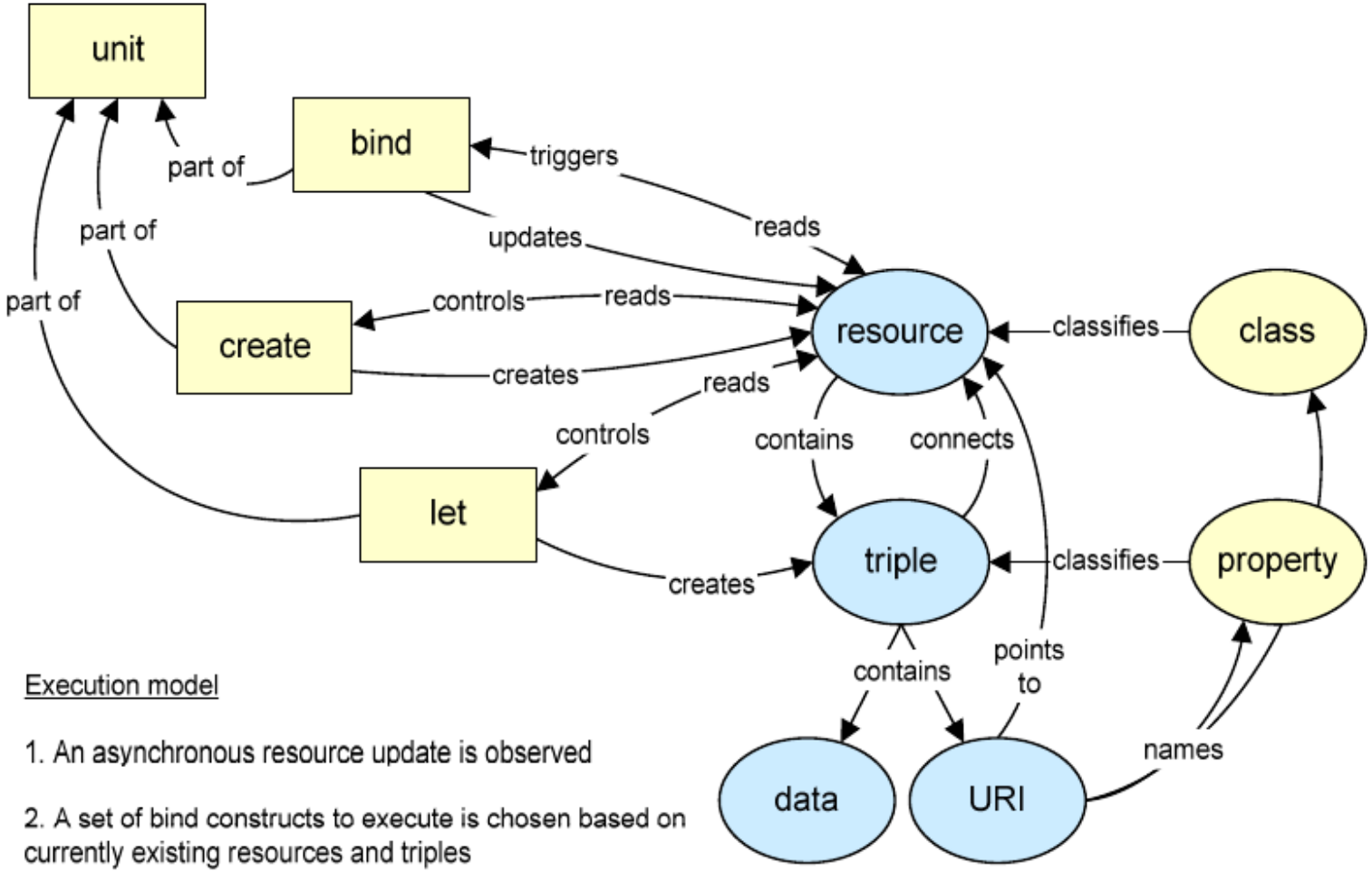


Execution model

- Data centric, reactive execution Model
  - driven by asynchronous external events
    - trigger an “initiating update”
    - recalculation of binds in “ensuing updates”
- Bind Construct
  - relation between one or more input-
  - exactly one output resource
  - function for updated output value
  - unidirectional, bidirectional and copying

Execution model

- Let Construct
  - creation of relations
    - classifies and adds properties
- Create Construct
  - creation of resources
    - subsumes <c:let> and creates new resources



Execution model

1. An asynchronous resource update is observed

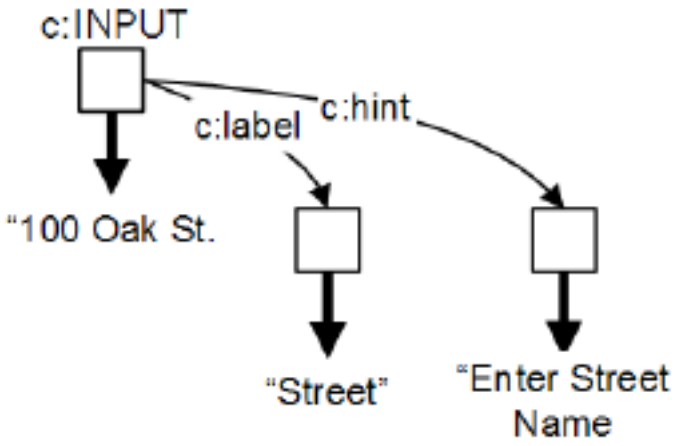
2. A set of bind constructs to execute is chosen based on currently existing resources and triples

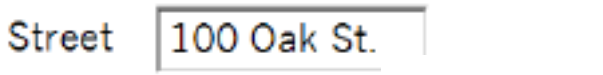
3. The binds are executed in dependency order

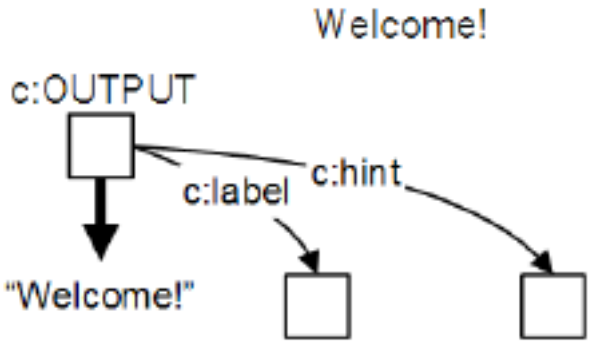
4. Repeat/use and let constructs are executed based on updated resources to create new resources and triples

Presentation layer

- simple text input C:INPUT
- and output: C.OUTPUT
  - hint, label attributes
- application window C:TOP







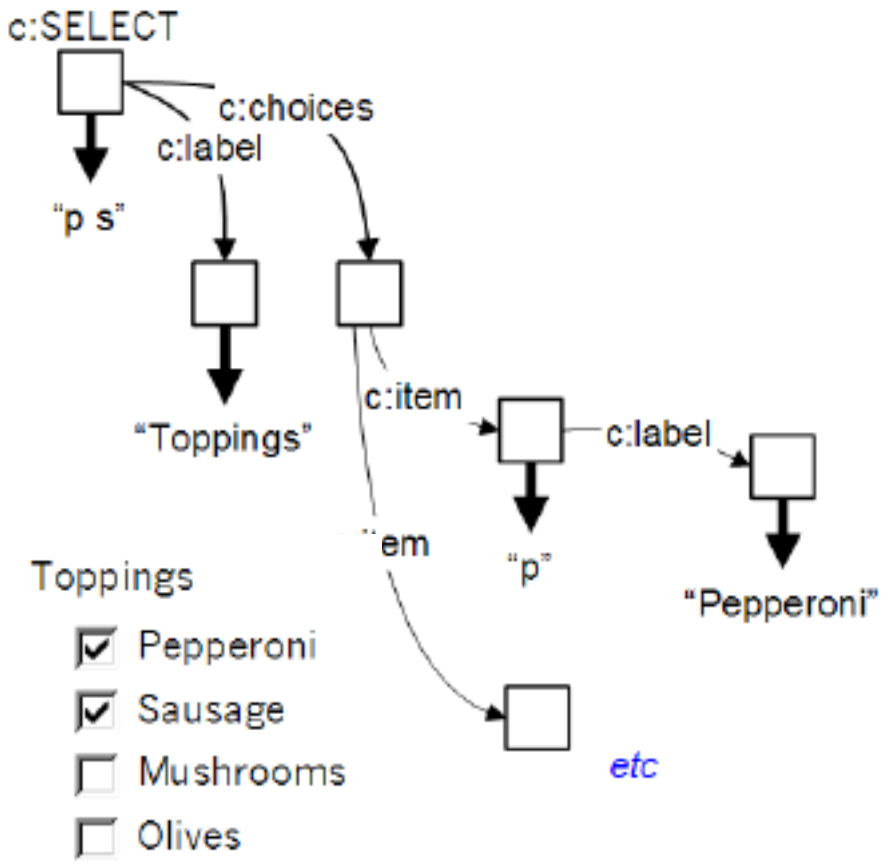

```
<c:create anchor="c:TOP" class="c:INPUT"
          property="in-field">
  <c:out path="c:label">Street</c:out>
  <c:out path="c:hint">Enter Street Name</c:out>
  <c:out>100 Oak St.</c:out>
</c:create>
```

Presentation layer

- Input abstractions
  - C:SECRET
  - C:TEXTAREA
  - C:TRIGGER
  - C:SELECT
- Layout container
  - C:VBOX, vertical flow layout
  - C:HBOX, horizontal flow layout
  - C:GRID, vertical flow layout

Presentation layer

```
<c:create class="c:SELECT">
 <c:out path="c:label">Toppings</c:out>
 <c:create property="c:choices">
  <c:create property="c:item">
   <c:out path="c:label">Pepperoni</c:out>
   <c:out>p</c:out>
  </c:create>
  <c:create property="c:item">
   <c:out path="c:label">Sausage</c:out>
   <c:out>s</c:out>
  </c:create>
  <c:create property="c:item">
   <c:out path="c:label">Mushrooms</c:out>
   <c:out>m</c:out>
  </c:create>
  <c:create property="c:item">
   <c:out path="c:label">Olives</c:out>
   <c:out>o</c:out>
  </c:create>
 </c:create>
</c:create>
```



Presentation layer

– bind input and output together

```
<c:bind>
 <c:in path="in-field" variable="$input"/>
 <c:out path="out-field">{$input}</c:out>
</c:bind>
```

– use trigger to update passive input

```
<c:create class="c:TRIGGER" property="the-trigger">
<c:bind>
 <c:in path="in-field" variable="$input" passive="true"/>
 <c:in path="the-trigger"/>
 <c:out path="out-field">{$input}</c:out>
</c:bind>
```

–

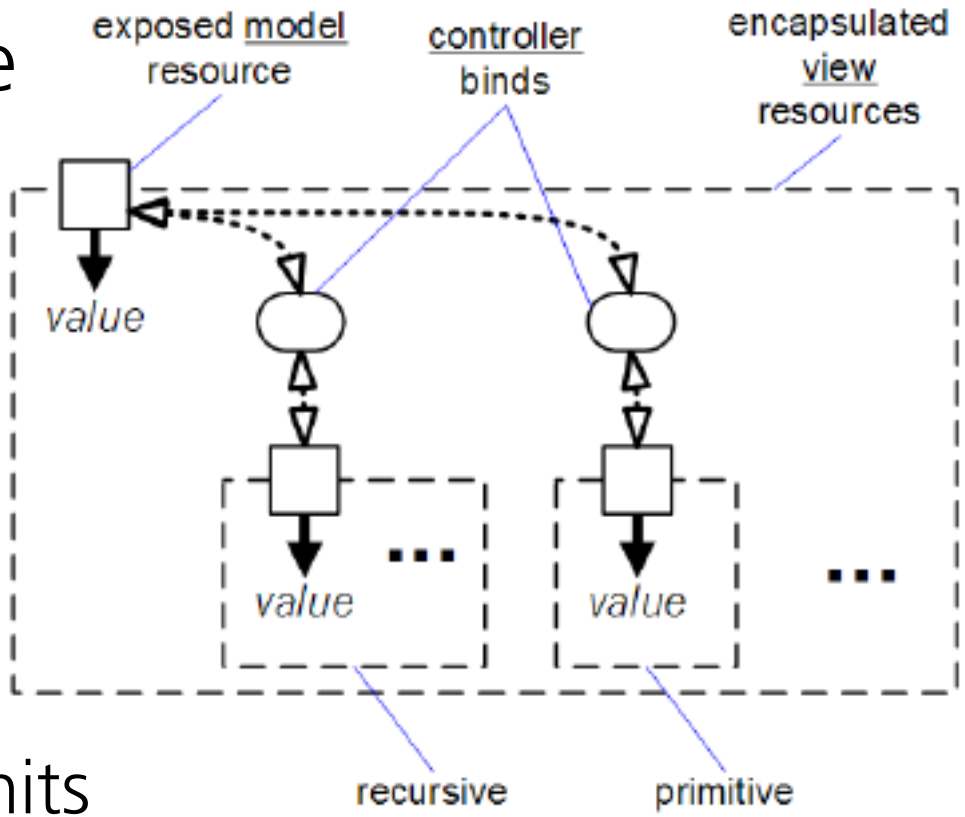
Data initialization

– "sending" output to initialize data model

```
<c:bind anchor="c:TOP" init="true">
<c:out>
 <foo>
    <bill>another value</bill>
    <bob>value of foo/bob</bob>
  </foo>
 </c:out>
</c:bind>
```

Presentation layer – MVC

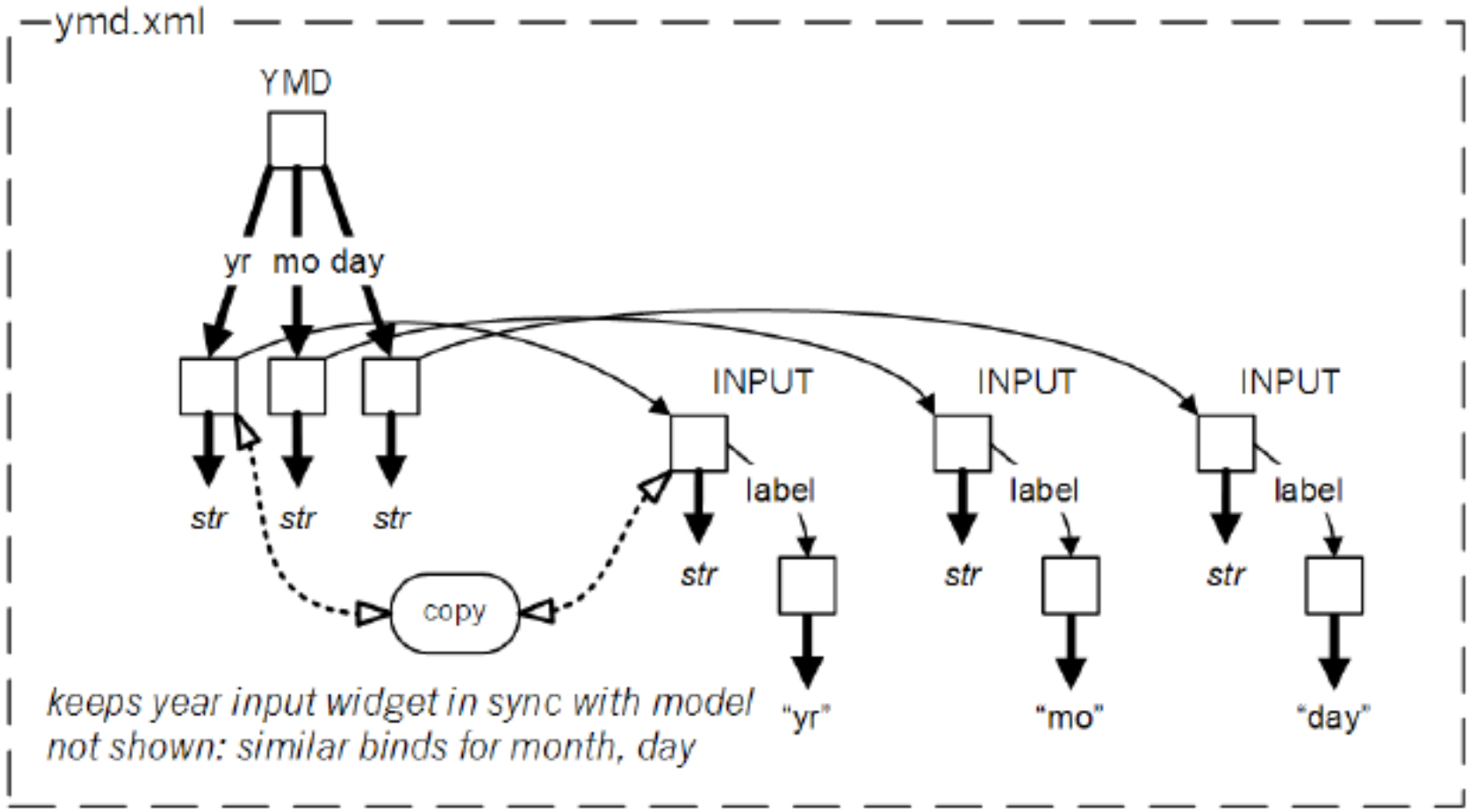
- Model: data resource
- Controller: binds
- View: associated visual resources
  - may itself serve as model
  - recursive MVC out of abstract interaction units



Recursive MVC Example

```
<c:with anchor="YMD">
  <c:create class="c:INPUT" ref="yr" property="yr-field">
    <c:out path="c:label">year</c:out>
  </c:create>
  <c:create class="c:INPUT" ref="mo" property="mo-field">
    <c:out path="c:label">month</c:out>
  </c:create>
  <c:create class="c:INPUT" ref="day" property="day-field">
    <c:out path="c:label">day</c:out>
  </c:create>
</c:with>
```

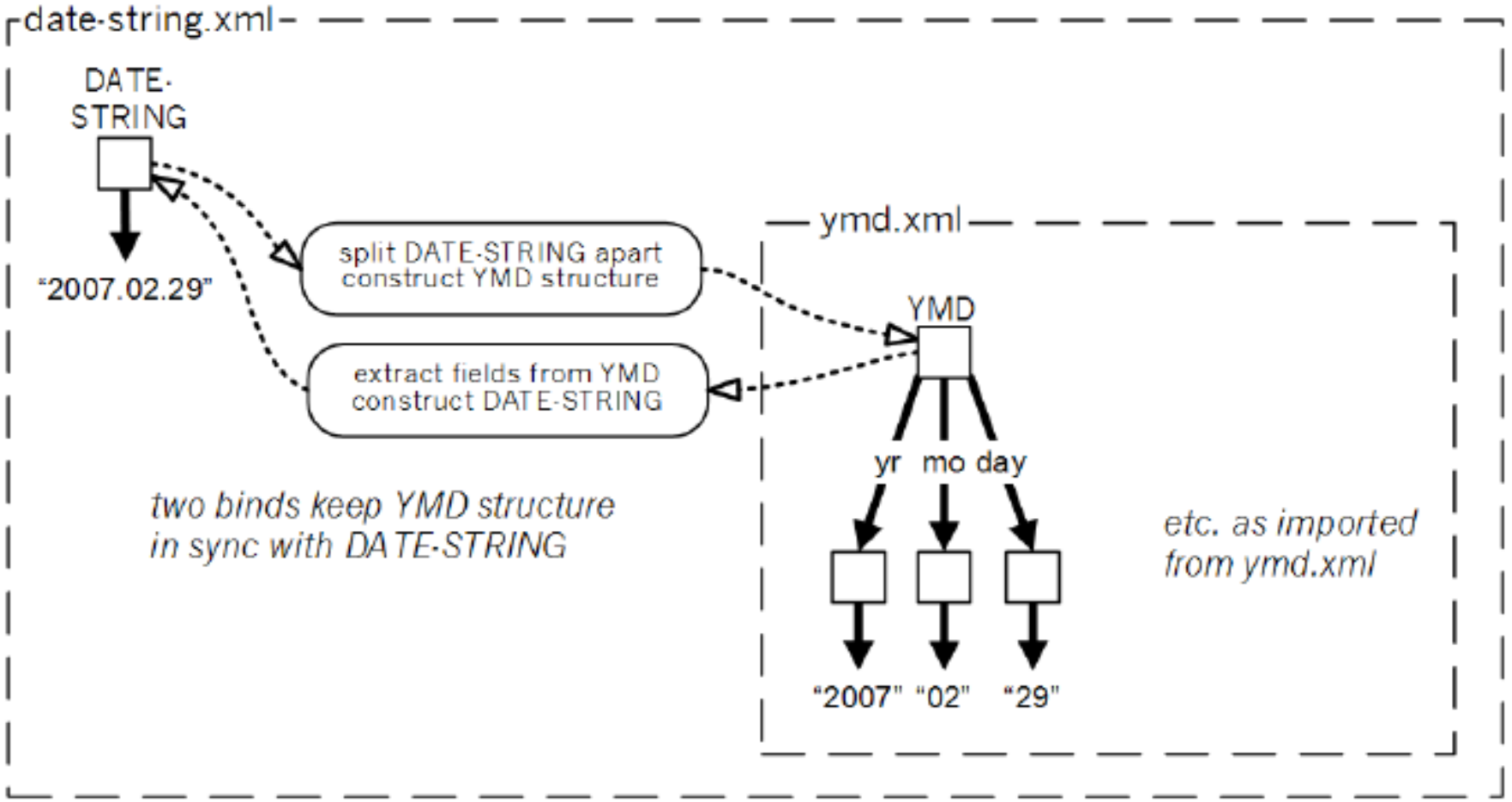
Recursive MVC Example



Recursive MVC Example

```
<c:with anchor="DATE-STRING">
  <c:create class="YMD" property="ymd"/>
  <c:bind>
    <c:in path="ymd" variable="$in"/>
    <c:out path=".">{$in/yr/text()}.{$in/mo/text()}.{$in/day/text()}</c:out>
  </c:bind>
  <c:bind>
    <c:in path="." variable="$in"/>
    <c:out path="ymd">
      <yr>{tokenize($in, '\.')[1]}</yr>
      <mo>{tokenize($in, '\.')[2]}</mo>
      <day>{tokenize($in, '\.')[3]}</day>
    </c:out>
  </c:bind>
</c:with>
```

Recursive MVC Example – request year

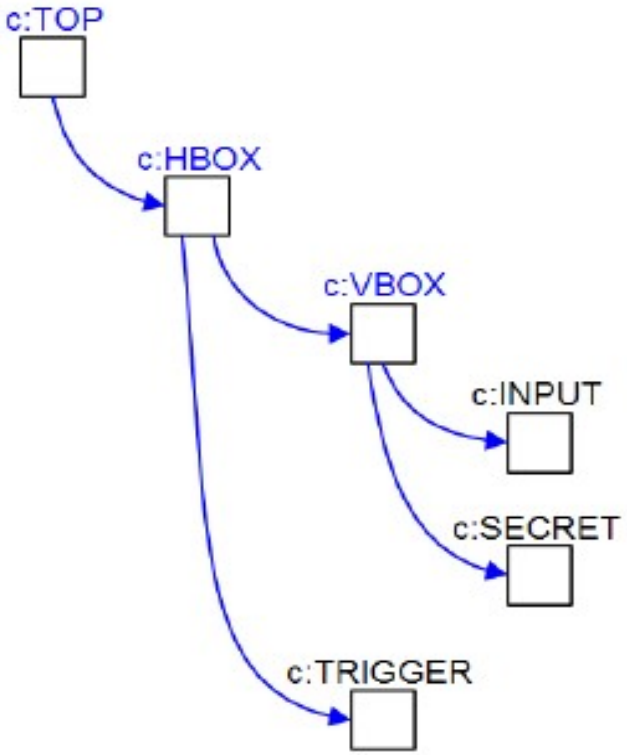


Abstract Layout Tree

– layout tree consists of
– a set of resources connected as a tree by the Collage layout containment property
– a root node (c:TOP) top-level user interaction window
– set of interior layout container
– a set of leaf nodes which
– are resources abstract interaction primitives (c:INPUT, c:OUTPUT, etc.).

```
<!-- fill a top-level window with a LOGIN-FORM -->
<c:create anchor="c:TOP" property="c:contains"
class="LOGIN-FORM"/>
<c:with anchor="LOGIN-FORM">
 <c:let class="c:HBOX">
  <c:create property="c:contains" class="c:VBOX">
   <c:create property="c:contains" class="c:INPUT">
    <c:out path="c:label">Username</c:out>
   </c:create>
   <c:create property="c:contains" class="c:SECRET">
    <c:out path="c:label">Password</c:out>
   </c:create>
  </c:create>
  <c:create property="c:contains" class="c:TRIGGER">
   <c:out path="c:label">Login</c:out>
  </c:create>
 </c:let>
</c:with>
```

Abstract Layout Tree



MBUI Layer Coverage