Multimediale Informationssysteme 2  
VO/KU SS 2005 - 706.054, 706.055  
Denis Helic
Introduction

Denis Helic
Course

Multimediale Informationssysteme 2 (VO 706.054)

Multimediale Informationssysteme 2 (KU 706.055)

Elective (optional) course for Telematics Master

Subject area catalogue: Information Systems, Multimedia
Who is talking?

Name: Denis Helic

How can I reach him?

Office: IICM, Inffeldgasse 16c, 1st Floor, Room D.2.05
Office hours: Wednesday from 11 til 12
Phone: +43-316/873-5617
email: dhelic@iicm.edu
Language

Lectures in English

Communication in German/English

If in German: please informally (Du)!

Constructional Example: German/English

Examination: German/English
Organization of the course

Lectures: Thursday, 11:15 – 12:45, HS i12

Registration for Lectures (VO), Exercise (KU) (in TU Online System until 31.03.2006)

Lecture Slides, Links to external resources
[http://coronet.iicm.edu/lectures/mmis2](http://coronet.iicm.edu/lectures/mmis2).

The Web site will be updated a couple of times during the term

Newsgroup tu-graz.lv.mmis2 (news.tu-graz.ac.at)
In MMIS2 we deal with the Web as an application platform

**Goal:** To learn and understand the **specifics of Web as an application platform**

HTTP, URL, Client/Server, Data Formats, ...

**Goal:** To learn about the **technological trends**

What technology is currently used? (XML, (X)HTML, Web App Frameworks, Web Services, ...)

**Goals of the course (1/2)**
Goals of the course (2/2)

Goal: To learn about the **trends in development methods**

- What are the software engineering methods and how to develop Web apps?

Goal: To learn about the **software design trends**

- What are common problems and common design patterns in Web applications?
How we will achieve these goals?(1/2)

The theoretical background and overview of the current trends

- Lectures, slides, links to articles, ...

Practical implementation of a Web application

- Following a software (Web) engineering method
- Using some of the technologies that we have discussed
- Applying Web-oriented design patterns
Presentation of the achieved results
- All students participate in the presentation

Discussion of the results
- With the lecturer
- With the peers
Data Formats: Markup Languages (XML)

Document-oriented XML
- XML Publishing, XML-based Content Management Systems
- Using XSLT and XSL-FO for presentation

Data-oriented XML
- Data types
- Storing XML data, Native XML Databases

XML as a means for Declarative Web development
Traditional Page-oriented Web applications

- Server-side programming with Java (Ruby)

Design Patterns

- Model-View-Controller Pattern
- Web Application frameworks: Apache Struts, Ruby on Rails

Data Management in Web applications

- Object/relational mappings
- Frameworks and patterns: Hibernate, Data Access Object, ...
Topics of the course (3/3)

- Service oriented Web applications
- Web Services
  - SOAP, WSDL
  - REST architectural style
  - Web Services Frameworks: Apache Axis
- Rich Web Clients and Web Services
  - AJAX
09.03. Web Eng. Intro/Intro on frameworks for practical work
16.03. XML Publishing, XSL-FO, Cocoon
23.03. XML Data Management, Native XML Databases, XQuery
30.03. Model-View-Controller (Struts)
06.04. Data Management in Web Applications (O/R Mapping, Hibernate)
04.05. Model-View-Controller (Ruby on Rails)
11.05. Web services (SOAP, WSDL)
18.05. Web services (REST, AJAX)
01.06. Project presentations (XML Pub.)
08.06. Project presentations (MVC)
22.06. Project presentations (Web Serv.)
No lectures for KU

Registration: Required (in TU Online System)

Registration deadline: 31.03.2006
Goals:

- Implementing a Web application
- Learning about different technologies, methods
- Working in groups
- To try out something new
Prerequisites

- (X)HTML, CSS
- XML
- Java
Group work: groups of 3-5 people

Group registration with a short description
http://coronet.iicm.edu/denis/students/mmis/group.txt

Project plan (Time estimation, group members responsibilities, ...)
http://coronet.iicm.edu/denis/students/mmis/plan.txt
Documentation of Code and Design!

http://coronet.iicm.edu/denis/students/mmis/doc.pdf

KU example points to MMIS 1 (documents are same, the content is different)

Documentation is not equal to the source code

Documentation is a description of the system

Architecture, Installation, User Guide, etc.
Excercise (KU) - Organization of the course (6/7)

Presentation

- 15min + 5min discussion (if not so many groups: 20min+5min)

Argue about your development decisions

- Why did you take MVC architecture?

Point advantages, disadvantages
Excercise (KU) - Organization of the course (7/7)

Deadlines:

- Group building and preliminary description of the project (what we are going to do): 31.03.2006
- Project plan: 04.05.2006
- Completed project: one day before presentation
- XML Pub.: 31.05.2006
- MVC: 07.06.2006
- Web Ser.: 21.06.2006
Grading of the KU:

- Project plan: 25%
- Documentation: 25%
- Implementation: 50%

Project submission via e-mail: Denis Helic (dhelic@iicm.edu)
Grading of the lecture (Oral exam)

- Presentation 50%
- Answering questions 50%

Additional points if you participate in discussion

- E.g. you can improve your grade in this way
Important for the exam!!!

Show how you applied design patterns

Show how you applied a method

Argue why you did something!!!

Advantages/disadvantages!

Note that this is very similar to Sem/Proj or Master Exam
If you do not make a VO/KU combination

Only KU
  - If you are good at answering questions you can improve your grade

Only VO
  - Oral exam, with questions going more into details
  - Special exam dates in TUGOnline
Students can implement whatever Web application they like!

- Web shops
- Content management systems
- Collaborative systems
- Digital libraries
Three Java Frameworks + One Ruby Framework

In some cases something else can be done
  
  You need to discuss this with me first!

Native XML databases

Hibernate

REST + AJAX
Introduction to Web Engineering

Denis Helic
The Web started as a static information system

- Web servers served static HTML documents

Users wanted more interaction with the Web

- Interactive Web-based phone book

Introduction of HTML forms

- Need for a dynamic response from Web servers

Server-side programming as an answer: CGI, Perl, PHP, ...(MMIS1)
The Web and the Internet became ubiquitous

Faster than any other technological development in history

Manufacturing, travel, banking, education, government

All are Web-enabled

The Web is an application platform!
E-commerce crossing national boundaries

Porting of legacy systems to the Web

Wireless applications → mobile Web applications

We increasingly depend on a range of Web applications
We rely on Web applications → they need to be reliable and perform well

To build such applications we need a sound methodology

- Process, tools, guidelines

Web engineering

- Systematic, scientific, engineering and management approach
- Develop, deploy and maintain qualitative Web applications
Web software engineering is similar to traditional software engineering.

- The process follows a similar methodology.
- But at each step we need to take care about special issues related to the Web.

- Requirements engineering
- System analysis, system architecture and design
- Implementation
- Testing
New dimension of dealing with users

In traditional SE you know your users
  Easy to derive user requirements

In WE you don’t know your users
  Potentially, the whole world!
Even if you have a target group it can change in the future

Additionally, you need to compete with others over the users

→ Usability, accessibility, graphic design become very important

Popularity is important!

→ Google based its search engine on popularity → a large success
Web apps UR vs. traditional apps UR (3/3)

- One very popular approach to collect user requirements on the Web
- Start a basic version of a Web app, e.g., a beta version
  - Web 2.0 beta
- Build a user basis, collect the feedback
- Develop the app by taking into account the feedback
  - Social aspects of software development
- No need for usability tests, user simulations before the start of the app
Web apps SA vs. traditional apps SA (1/2)

- Generic architecture of user-oriented software applications
- Process (application) logic
- Data management
- User interface
A typical example

Managing student records in university

Process logic: add a student, delete a student, insert grades, exams, ...

Data management: tables of students, grades, exams (relational databases)

User interface: GUI with menus, buttons, text fields, etc.
Generic architecture of Web applications
Traditionally, developers deal mainly with process logic and data management.

User interface lately in the process and in a hurry.
- Poor quality of interfaces → users complain.

The Web is appealing because of its consistent user interface.
- HTML, platform independence.
- One of the reasons for the success!
- It is a huge advantage for developers!
Hyperlinks are a specific component of a Web-based user interface
URL/URI for addressing of Web pages (of modules of a Web app)
Important to provide meaningful URLs
Single URL (e.g. /app.cgi)
  No bookmarks, difficult for humans, bad user interface
  Problems with search engines
Multiple URLs

- /student/add /student/show
- /student/exam/add /student/exam/show

Meaningful, easier for humans

Bookmarks

Search engines can retrieve different parts and index it
Traditionally, app logic manages the app state

- E.g., the current state of the data, user inputs, etc.

Typically, Web browser supports only HTML and does not have direct connection to the app logic

- Communication over network and HTTP with the app logic

HTTP stateless (connection-less)

Web server needs to track users and sessions (cookies, URL rewriting)
However, Web server provides only low-level tracking

- Responsibility of the app logic to manage sessions
- Manage it within the application server

App server has other responsibilities as well

- Provides the basic app functionality
- Communicates with data management modules
- Communicates with external functionality (Web services)
We need to take care of separation of concerns

- Separate user interface and session management from the app logic
- Separate user interface and session management from data management

... 

Design patterns
Web server, application server, Web services

- All manage data → design patterns to separate concerns

Disparate and numerous data sources

- File system, databases, interlinked documents, document formats, metadata, etc..

- Design patterns to abstract access to data sources
Typically, Web applications deal with relational databases.

Need to manage relational data in object-oriented applications.

Use design patterns like Data Access Object (DAO).

Use object/relational mapping, like Hibernate framework.
Information retrieval

How to find what I’m looking for?

One approach are search engines with full-text processing

Another approaches manage links

Links in databases, or within documents

Mixed approach: full-text and links, e.g. Google
Yet another approach is managing metadata

Metadata is data about other data

On the web

- Tag information items (everything that you can access via URL) in a structured manner

- Search inside metadata

Data and document formats

XML!

- You should think about HTML as yet another application of XML
- XHTML

XML as a technology for managing documents

- For presentation we need style sheets (CSS, XSLT, XSL-FO)
XML as a technology for managing data
   We need storage tools, query languages (XPath, XQuery)

XML as a technology for exchanging data
   E.g. export/import data from databases

XML as a technology for distributed computing
   E.g. exchange messages between Web services
Further Readings (1/3)

General info on Web engineering

Web Engineering: An Introduction
isNumber=19981&prod=JNL&arnumber=923949&arSt=14&ared=18&arAuthor=Ginige%2C+A.%3B+Murugesan%2C+S.

Web Engineering: Creating a Discipline among Disciplines
isNumber=19845&prod=JNL&arnumber=917974&arSt=82&ared=87&arAuthor=Deshpande%2C+Y.%3B+Hansen%2C+S.
Both articles from special issues of IEEE Multimedia from 2001
http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=19981&puNumber=93
http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=19845&puNumber=93

Part of the IEEE Explore digital library

Free access for IPs from the TUG
Further Readings (3/3)

Article on architecture of Web applications

Grady Booch: The Architecture of Web Applications


Grady Booch is one of the creators of UML
Introduction to Frameworks for Exercises

Denis Helic
Web Application Frameworks

Three Java Frameworks + One Ruby Framework

- Cocoon for XML publishing [http://cocoon.apache.org/]
- Struts for MVC Web applications [http://struts.apache.org/]
- Rails for MVC Web applications [http://www.rubyonrails.com/]
- Axis for Web services [http://ws.apache.org/axis/]

(58/618)
Java Framework

Completely based on XML and XML processing

I.e., you can develop your example completely in XML without procedural programming

Based on principles of separating concerns

Complete separation of content, presentation, style, web-site management

Extensive use of XML stylesheet technologies, XSLT, XSL-FO
Pipeline-based processing of XML

XML Source → XML Transformation → XML Serializer

Source, e.g., an XML file

Transformation, e.g., an XSLT Transformation onto XHTML

Serializer sends back obtained XHTML
Many different combinations are possible

Alternative transformers

- XSLT transformation onto SVG (interactive multimedia presentations)
- XSLT transformation onto XSL-FO (formatting objects, “advanced CSS”)

Alternative serializers

- PDF serializer (in combination with XSL-FO)
- Office serializer, etc.
Alternative XML sources

- Native XML databases
- RDBMS, i.e., SQL sources

Multiple Sources

Multiple XML Sources → XML Aggregators → XML Transformation → XML Serializer

Aggregators are typically XSLT stylesheets that filter and combine XML data
Web-site management in a central file

Here you define all URLs of your Web application (matchers)

Using regular expressions to define sets of URLs, e.g. /student/*\.xml

Each matcher is related to an XML processing pipeline
Download [http://cocoon.apache.org/mirror.cgi](http://cocoon.apache.org/mirror.cgi)

Only available as source distribution

Compile and deploy it as a Tomcat webapp

Install instructions

Example: [http://coronet.iicm.edu/cocoon/mmis2/](http://coronet.iicm.edu/cocoon/mmis2/)
Struts Framework

- Java Server-Side Web Framework
- Follows a MVC design pattern to separate content, logic, presentation
- Controller in Struts is already available
  - You need to configure it
- Model is a collection of application specific Java classes
  - You need to implement them
- View is a collection of JSP pages
  - You need to implement them
Controller

The server receives different HTTP requests

- Requests can include different parameters submitted by the user
- The server dispatches the request to a specific handler (action)
- The server has a registry of mappings of parameters onto actions
- The registry is a Struts configuration file: struts-config.xml

Action that handles the request is a subclass of a Struts Action class

- You need to implement a number of action subclasses that handle requests
Model

Your application logic and application content is in a number of Java classes.

Additionally, you may connect to a database to manage the app data.

O/R mapping, data management patterns.
When the action is executed the request is forwarded to a specific JSP page.

JSP page reads the content from the Model (Java objects).

Produces valid HTML.

HTML is presented to the user.

Request → Find Action → Execute Action (Update Model) → Find View.
Download

http://struts.apache.org/download.cgi

You can download either binaries or sources

In the installation included: struts-blank.war

- You can use it for a quick start with Struts
- Only edit struts-config.xml and define your own actions
- Everything else is in place

Implement the model

Example: http://coronet.iicm.edu/struts-publicationdb
Ruby is an object-oriented dynamic programming language

- Similar to Smalltalk, Python


Rails is an MVC Web Framework

Principles behind Rails

- Less software (a lot of source code is generated - data driven)
- Convention over configuration (e.g., follow naming conventions and no need to configure anything)
Typical RoR workflow

Create a database schema
  Follow convention about naming tables and columns

Invoke a Rails script to generate model classes

Invoke a Rails script to generate controller classes

Start Rails server

Open your browser, access your app, and enjoy!
Ruby on Rails Framework (3/4)

- Automatically generated insert, update, list controllers, views
- Model classes directly connected to a RDBMS
- You can edit the generated source code and adjust it to your needs
  - e.g., create an association between two model classes (database relations)
- You can update the generated HTML templates
Ruby on Rails Framework (4/4)


Very productive

- E.g., a small Web app with 2 related tables only 50 LOC

Disadvantage: you need to learn Ruby

I personally recommend it: you will for sure have fun with Rails!

- Educational purpose: very clean implementation of MVC
Java Web Services Framework

Publishing of Web Services

- You have a number of Java classes (e.g., your app logic)
- You want to make a subset of that functionality available via Web
- Inform Axis about which methods you want to offer
- Automatically generate and publish it!
Clients communicate with services by exchanging messages in XML.

Axis supports also automatic generation of client Java classes.

Use this classes to work transparently with a Web service.

You write your client in Java and do not even know or care that the functionality is accessed over the Web.

You can write Rich Web clients, i.e., Java desktop apps, Java applets, etc.

You can also try to write an AJAX client.


Installation guide

Example: [http://coronet2.iicm.edu/axis/services/persondao?method=readAllPersons](http://coronet2.iicm.edu/axis/services/persondao?method=readAllPersons)
XML Validation

Denis Helic
XML well-formedness

- An XML document conforming to the basic XML syntax

Obviously, we need this to separate markup from content

XML validation

- An XML document conforming to a particular schema

- The schema can be written in different schema languages

- DTD, Relax NG, W3C XML Schema
Why do we need the validation (we already have the well-formedness)?

- To protect the system from data it cannot handle
  - e.g. a student matrikel number is equal to zero

- To enforce a contract between applications on the Web
  - For data exchange
  - Similar to an API, but here it is a contract for structuring data
  - Also similar to a database schema
The process of checking the conformance is validation.

There are different levels of validation:

- The validation of the markup (the content model validity)
  - The structure of an XML document
    - e.g. proper nesting of elements, proper attributes, etc.
The validation of the values (datatypes)

- Does the content of an element have the proper datatype?
- Does an attribute have the proper datatype?
- Are the values of an element in a valid range?

The validation of integrity

- Links between nodes within a document (not that hard)
- Links between documents (very hard)
Up to here the validation is purely syntactical

- What are the names of elements and attributes?
- What order they can appear, where they can appear?
- What are the spelling rules for attribute values and element content (data types)?
The validation of the semantics - business rules

- If the element “foo” contains string “bar” then the document must contain a “fred” element

DTD, Relax NG, W3C XML Schema can not validate business rules

You need a separate semantics-enabled validator

- Your application logic in e.g. Java
- Rule ML (not part of the course)
- XSLT with XPath, e.g. Schematron (not part of the course)
Limitations of DTD

- Not in XML - need a separate parser
- Problems with namespaces, i.e., combining XML applications
- Poor support for datatypes, no integers, strings, etc.
  - No datatypes at all for elements
  - Especially important for data-oriented XML
- Syntactical limitations
  - A Calendar DTD with months and exact number of day-elements for each month!
Course XML document

- An XML document describing the MMIS2 course
- An extended version of the MMIS1 course example to depict validation
Course root element contains elements such as title, description, etc.

```xml
<course xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns="http://coronet.iicm.edu/courses">
  <title>Multimedia Information Systems 2</title>
  <description>Course on Web Engineering</description>
  <homepage xlink:href = "..."
    xlink:type="simple">MMIS2 homepage</homepage>
  <lecturer>Denis Helic</lecturer>
  ....
</course>
```
Content element can have a number of topic elements

```xml
<content>
  <topic>XML Validation</topic>
  <topic>Data-oriented XML</topic>
  ...
  <topic>Web Services</topic>
  <topic>Semantic Web</topic>
</content>
```
Schedule element contains a number of events

Each event has a date, time, duration, place, etc.
<event>
  <date type="ISO">
    <year>2005</year>
    <month>03</month>
    <day>10</day>
  </date>
  <time>11:15:00</time>
  <duration>45</duration>
  <place>HSi12</place>
  <title>XML Validation</title>
  <slides xlink:href = "...
    xlink:type="simple">xml_validation.pdf</slides>
</event>
<event>
  <date type="Euro">
    <day>17</day>
    <month>March</month>
    <year>2005</year>
  </date>
  <time>11:15:00</time>
  <duration>45</duration>
  <place>HSi12</place>
  <title>XML Publishing</title>
  <slides xlink:href = "...">
    xlink:type="simple">xml_publishing.pdf</slides>
</event>
We have content models

We have datatype checks

- Day values are integers, time values are in a specific time format
- Duration values are integers in a range from 45 to 90
- Place values are of the form “HSi” + two digit number
We have business rules (very simple)

Depending on the type of a date, sub-elements are structured differently

Complete document
http://coronet.iicm.edu/mmis2/examples/xml/validation/course.xml

DTD for the document (very limited functionality)
http://coronet.iicm.edu/mmis2/examples/xml/validation/course.dtd
Developed by the Web Consortium
http://www.w3.org/XML/Schema

Two parts of the recommendation

XML Schema Part 1: Structures
http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes
http://www.w3.org/TR/xmlschema-2/
A particular XML schema is a definition of a number of elements and attributes.

Each element or attribute has a name (tag-name) and a type.

Types can be complex types and simple types.

Complex types are used to define the content model.

Simple types are used to define datatypes of elements or attributes.
Basically, defining an XML schema is equal to defining types

Types can be then referenced in element or attributes definitions

You can think of a type as a class in object-oriented languages

XML Schema allows to sub-class types

In this way you can achieve more specific behaviour
Defining a content model (Course element)

```xml
<xs:element name="course">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="courses:title"/>
      <xs:element ref="courses:description"/>
      <xs:element ref="courses:homepage"/>
      ...
      <xs:element ref="courses:content"/>
      <xs:element ref="courses:goal"/>
      <xs:element ref="courses:schedule"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
The course element is a complex type that has a sequence of elements. Elements appear in that order and are repeated once. 

- minOccurs, maxOccurs attributes can be used

- Default value is 1, unbound for as many as you want

Other elements are referred to (reuse of types)
Defining datatypes

```xml
<xs:element name="description" type="xs:string"/>
<xs:element name="homepage">
  <xs:complexType mixed="true">
    <xs:attribute ref="xlink:href" use="required"/>
    <xs:attribute ref="xlink:type" use="required"/>
  </xs:complexType>
</xs:element>
<xs:element name="lecturer" type="xs:string"/>
```
The description element is of type string.

The homepage is a complex type because it has a structure.

- The structure is composed of a textual content and two attributes.

These elements are referred to from the course element.
Defining datatype restrictions

```xml
<xs:element name="time" type="xs:time"/>
<xs:element name="duration" type="courses:lectureDuration"/>
<xs:element name="place" type="courses:lectureRoomType"/>
```

The time element is of type xs:time

http://www.w3.org/TR/xmlschema-2/
The duration and the place element are user-defined types

<xs:simpleType name="lectureRoomType">
    <xs:restriction base="xs:string">
        <xs:pattern value="HSi\d{2}"/>
    </xs:restriction>
</xs:simpleType>

<xs:simpleType name="lectureDuration">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="45"/>
        <xs:maxInclusive value="90"/>
    </xs:restriction>
</xs:simpleType>
The previous example shows also sub-typing

The new type (e.g. lectureRoomType) inherits properties from its supertype

It is also a string

Additionally it restricts the supertype
What is with the business rules?

```xml
<xs:element name="date">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="courses:day"/>
      <xs:element ref="courses:month"/>
      <xs:element ref="courses:year"/>
    </xs:choice>
    <xs:attribute name="type" use="required"
                 type="xs:NCName"/>
  </xs:complexType>
</xs:element>
```
No means to check business rules in W3C XML Schema

Additional validator needed, e.g. XSLT and XPath

The Schematron

http://xml.ascc.net/resource/schematron/schematron.html

Complete schema

http://coronet.iicm.edu/mmis2/examples/xml/validation/course.xsd
W3C XML Schema - Analysis

- Considered complex
- Not that flexible (structurely)
- Not that intuitive (types-based not structure-based)
- Long time needed to learn all the features
- Tool supported not that good as for DTD
  - But it is fastly growing
Relax NG: Regular Language Description for XML Core - New Generation

Developed by OASIS (Organization for the Advancement of Structured Information Standards)

http://www.oasis-open.org/home/index.php

Developed as an ISO standard

Relax NG at Oasis

A schema written in Relax NG is close to a textual description of vocabulary.

E.g. The course element contains the description element, the homepage element, ...

E.g. The schedule element contains a number of event elements

   An event element contains the date, time, place...

Wrap this in an XML-based syntax → A Relax NG schema
Complete separation between structure (the content model) and datatypes

Compare with W3C XML Schema where the structure is a (complex) datatype

The structure defined by means of patterns

Similar to regular expression patterns

But here patterns are patterns of elements, attributes

Reusing of patterns is possible

Very intuitive!
How datatypes are handled?

- Specifies only few datatypes
  
  - You can plug-in any datatype system that you want
  
  - Mostly, W3C XML Schema datatype system
Defining a content model (Course element)

Just transform the above english sentences into XML

```xml
<element name="course">
  <ref name="title"/>
  <element name="description">
    <text/>
  </element>

  ...

  <element name="lecturer">
    <text/>
  </element>

  ...

</element>
```
Repeating elements

```xml
<element name="schedule">
  <oneOrMore>
    <element name="event">
      <element name="date">
        <attribute name="type">
          <choice>
            <value>Euro</value>
            <value>ISO</value>
            <value>US</value>
          </choice>
        </attribute>
      </element>
    </oneOrMore>
  </element>
</element>
```
Reusing patterns (the title element)

- Can be a part of the course and a part of an event

```xml
<define name="title">
  <element name="title">
    <text/>
  </element>
</define>
```

- This is not a type it is a pattern to look for in an XML document
Handling datatypes

<grammar xmlns:xlink="http://www.w3.org/1999/xlink"
    ns="http://coronet.iicm.edu/courses"
    xmlns="http://relaxng.org/ns/structure/1.0"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

    ...

    <element name="year">
        <data type="integer"/>
    </element>

    ...

</grammar>
Imposing restrictions on datatypes

Can impose everything that the plugged datatype system supports

```xml
<element name="duration">
  <data type="integer">
    <param name="minInclusive">45</param>
    <param name="maxInclusive">90</param>
  </data>
</element>

<element name="place">
  <data type="string">
    <param name="pattern">HSi\d{2}</param>
  </data>
</element>
```
Can not impose business rules (Interleave means in any order)

```xml
<interleave>
  <element name="day">
    <data type="integer"/>
  </element>
  <element name="month">
    <data type="string"/>
  </element>
  <element name="year">
    <data type="integer"/>
  </element>
</interleave>
```

Complete example

http://coronet.iicm.edu/mmis2/examples/xml/validation/course.rng
Relax NG - Analysis

- Very flexible (structurely) because of patterns
- Very intuitive (structure-based)
- Shorter time needed to learn all the features
- Tool supported not that good as for DTD or even XML Schema
  - But it is fastly growing ;)

(116/618)
**Schemas development tools (1/2)**

- **XMLSpy** (Altova from Vienna)
  
  [http://www.altova.com](http://www.altova.com)

  Validates with DTDs, W3C XML Schemas

- **Jing** by James Clark
  

  Validates with Relax NG

- **Schema converter Trang** by James Clark
  

  Converts in all possible directions between DTD, Relax NG, XML Schema
Using Trang to do Validation by Instance

First you write an XML document

You work with the document and improve it

Then you use Trang to automatically obtain a schema (e.g. Relax NG)

Then you optimize the schema manually

Very productive way to develop schemas
Further Readings

Schema articles on [www.xml.com](http://www.xml.com/pub/q/all_schema)

W3C XML Schema Tutorial

Relax NG Tutorial
XML Publishing

Denis Helic
Different data sources in Web applications

- Documents in file systems (HTML, XML, SVG, ...)
- Data in relational and XML databases
- Metadata in different formats (files, databases,...)
XML Publishing - Introduction (2/8)

- Web-based XML publishing approach
- Applies XML as a common denominator (gemeinsamer Nenner)
- Bring all data in at least XML
  - Dynamically
Three layers of data management

The first (low-level) layer deals with data in a storage-specific way

Data Access Layer

- Access files in a file system by means of File I/O
- Use SQL to access RDBMS, e.g., programmatically with JDBC in Java
Serialize the data as XML

Output of the first layer is input for the second layer

The second layer deals only with XML

Data Aggregation Layer
Validating XML (can be also done in the first layer)

- In the case of an RDBMS data validation already done by the RDBMS
- Programatically written → usually data is valid
- Validate once more and you can find errors in the program
Transforming data as XML

- Sorting data, filtering, transforming to another format, etc..

Integrating (aggregating) data from different sources

- Once when you have all data sources in XML → easy integration

Output of the second layer is input for the third layer
The third layer deals with presentation of XML

Data Presentation Layer

- Presenting in different formats such as PDF, XHTML, SVG
- Presenting for different devices such as desktop PC, PDA, mobile phone, etc.

Applying XSL-T and XSL-FO to do the job
We have a Web application offering information about courses and students.

Which courses do I have, which students registered for what course?

- An 1:n relation between courses and students

Information can be presented in HTML for viewing, browsing.

Additionally, presentation as PDF for e.g. printing.
Manage course information in an XML document

MMIS 2 course XML document

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/content/course.xml

Advantages of managing course information as XML

Single-source publishing is available

Manage document only in XML, publish in different formats
You can impose XML validation during content authoring

- Using an XML editor that supports validation
- Authors (lecturers) must create the documents that you expect!

The first layer only retrieves the document from the file system
Manage students data in a relational database

- Data is updated often (e.g. new students, new term, ...)

Query the database by means of SQL, e.g. in Java

Wrap the data into an XML document (the first layer)
The second layer has all the data available in XML. Validation, integration of the data in a single chunk of XML data. The third layer can present the data in XHTML or in PDF.
XML Publishing - Separation of Concerns

Still need to separate this (later in the course)

User Interface

Data Presentation Layer

Data Aggregation Layer

Data Access Layer

Data Management

Process Logic

User Interface

Data Presentation Layer

Data Aggregation Layer

Data Access Layer

File IO

SQL

XPath

XQuery

Metadata QL

Metadata DB

XML DB

XML

PDF

XHTML
An Open Source Apache project
http://cocoon.apache.org/

Web development framework supporting separation of concerns
http://cocoon.apache.org/2.1/introduction.html

Management of a Web-site
Coocon - Java-based XML Publishing Framework

Download [http://cocoon.apache.org/mirror.cgi](http://cocoon.apache.org/mirror.cgi)

Only available as source distribution

Compile and deploy it as a Tomcat webapp

- Some problems with Java 1.5
- With Java 1.4 no problems

Install instructions

Based on the concept of component pipelines

http://cocoon.apache.org/2.1/overview.html

A chunk of XML data is pushed through a pipeline

Matcher, generator, transformer, serializer

http://cocoon.apache.org/2.1/userdocs/concepts/index.html
Coocon - Java-based XML Publishing Framework (4/19)

Corresponds approximately the layers from above

- Transformer sometimes receives something else than XML
- Very often transformer produces the presentation
- Since everything in XML not so dangerous
SAX is Simple API for XML

- Originally Java, event-based
Matchers

Matchers

http://cocoon.apache.org/2.1/userdocs/matchers.html

Matches a URL with a specified pattern

- Wildcards, regular expressions

When the URL is matched it dispatches the request to a particular pipeline

- The pipeline process the XML document and produces a response
<map:match pattern="">
  ...
</map:match>

<map:match pattern="course.xml">
  ...
</map:match>

<map:match pattern="course.html">
  ...
</map:match>
Generators

Creates XML data from a data source

Files, directory lists, image lists, etc..

A lot of generators in Coocoon

http://cocoon.apache.org/2.1/userdocs/generators.html
<map:match pattern=""/>
  <map:generate src="examples.xml"/>
...
</map:match>

Default generator is the FileGenerator
Transformers

Transforms XML data coming from a generator into another XML

Using XSLT, XSLTTransformer is default transformer

A lot of transformers in Coocoon

http://cocoon.apache.org/2.1/userdocs/transformers.html
<map:match pattern=""">
<map:generate src="examples.xml"/>
<map:transform src="context://mmis2/style/xsl/html/simple-samples2html.xsl">
<map:parameter name="contextPath" value="\{request:contextPath\}"/>
</map:transform>
...
</map:match>
You can chain transformers to make more than one transformation:

```xml
<map:match pattern="students.html">
  <map:generate src="content/students.xml"/>
  <map:transform type="sql">
    <map:parameter name="use-connection" value="mmis2"/>
  </map:transform>
  <map:transform src="style/xsl/sql2xml.xsl"/>
  <map:transform src="style/xsl/students.xsl"/>
  <map:serialize type="html"/>
</map:match>
```
Note SQLTransformer

- Receives response from the RDBMS
- Transforms that response into Cocoon internal XML
- Chains another transformer to get it into XML format that you want
What is the generator for the SQLTransformer?

It is an XML file containing an SQL query

```xml
<page xmlns:sql="http://apache.org/cocoon/SQL/2.0">
  <content>
    <execute-query xmlns="http://apache.org/cocoon/SQL/2.0">
      <query>
        select * from registered
      </query>
    </execute-query>
  </content>
</page>
```
Serializer

 Writes the XML data in a specified format

 Default is HTML

 Many serializers in Cocoon

 http://cocoon.apache.org/2.1/userdocs/serializers.html
Web-site management

All pipelines defined in a central file called sitemap.xmap

Another XML file ;)

One sitemap.xmap per directory (similar to .htaccess)

You can access only the URLs defined in the sitemap.xmap

Top sitemap.xmap contains general configuration

Inherited by all sitemaps in sub-directories
<map:sitemap xmlns:map="http://apache.org/coocoon/sitemap/1.0">
  <map:pipelines>
    <map:pipeline>
      <map:match pattern="course.xml">
        ...
      </map:match>
      ...
    </map:pipeline>
    <map:match pattern="course.pdf">
      <map:generate src="content/course.xml"/>
      <map:transform src="style/xsl/course2fo.xsl"/>
      <map:serialize type="fo2pdf"/>
    </map:match>
    ...
  </map:pipelines>
</map:sitemap>
Where is the logic?

XSP (eXtensible Server Pages)

http://cocoon.apache.org/2.1/userdocs/xsp/logicsheet.html

It is an XSLT document that contains Java code

Actions (under development)

Java code

http://cocoon.apache.org/2.1/userdocs/actions.html

Within XSLT
MMIS 2 course and registered students in Cocoon
http://coronet.iicm.edu/cocoon/mmis2/

Samples XML document from Cocoon
http://coronet.iicm.edu/mmis2/examples/xml/cocoon/examples.xml

XSLT for transforming it into HTML already available
sitemap.xmap
http://coronet.iicm.edu/mmis2/examples/xml/cocoon/sitemap.xmap
Example has three parts

The first part: Course XML document

XSLT for producing HTML

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/xsl/course.xsl

CSS for formatting HTML

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/css/course.css

How to retrieve CSS in Cocoon: sitemap.xmap

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/sitemap.xmap
The second part: the student database

Registered students database from MMIS 1

Inserting students

http://coronet.iicm.edu/mmis/examples/java/mysql/form.html

Retrieving students

http://coronet.iicm.edu/mmis-servlets/Registration
XSLT for producing HTML

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/style/xsl/students.xsl

How to retrieve data from MySQL: sitemap.xmap

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/sitemap.xmap

SQL document

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/students/content/students.xml
Additionally, load the MySQL driver in web.xml

```xml
<init-param>
  <param-name>load-class</param-name>
  <param-value>
    org.gjt.mm.mysql.Driver
  </param-value>
</init-param>
```
Define SQL connection (Cocoon manages connection pool) in coo-
coon.xconf

```xml
<datasources>
  ...
  <jdbc name="mmis2">
    <pool-controller max="10" min="5"/>
    <auto-commit>false</auto-commit>
    <dburl>jdbc:mysql://localhost/mmis</dburl>
    <user>student</user>
    <password>student</password>
  </jdbc>
  ...
</datasources>
```
The third part: data integration

How to integrate data: sitemap.xmap
http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/sitemap.xmap

XSLT for producing HTML
http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/combined/style/xsl/course.xsl

Reusing of stylesheets from before
XSL-FO for producing PDF (1/2)

- XSL Formatting Objects: XSL-FO
- The second part of XSL specification
  http://www.w3.org/TR/xsl/
- Similar to CSS
  - You can define font-family, font-weight
- An XML syntax and more powerful than CSS
Usually you will not create XSL-FO sheet manually

You use XSLT to produce XSL-FO from your input document

You need a processor (renderer) to translate an XSL-FO into PDF

Apache FOP

http://xml.apache.org/fop/

Integrated into Cocoon
MMIS 2 course and registered students in Cocoon
http://coronet.iicm.edu/cocoon/mmis2/

Produce PDF out of Course XML document

Some features of XSL-FO explained
Page properties

<fo:layout-master-set>
<fo:simple-page-master margin-right="2.5cm"
    margin-left="2.5cm" margin-bottom="2cm"
    margin-top="1cm" page-width="21cm"
    page-height="29.7cm" master-name="page">
    <fo:region-before extent="3cm"/>
    <fo:region-body margin-top="3cm"/>
    <fo:region-after extent="1.5cm"/>
</fo:simple-page-master>
</fo:layout-master-set>
Page numbering

<fo:page-sequence master-reference="all">
<fo:static-content flow-name="xsl-region-after">
<fo:block line-height="14pt" font-family="serif" font-size="10pt" text-align="center">
  page
  <fo:page-number/>
</fo:block>
</fo:static-content>
</fo:page-sequence>
Page flow within blocks

<fo:flow flow-name="xsl-region-body">
  <fo:block text-align="center"
    space-before.optimum="24pt"
    font-size="36pt">Multimedia Information Systems 2
  </fo:block>
...
  <fo:block text-align="center"
    space-after.optimum="24pt"
    space-before.optimum="12pt"
    font-size="14pt">General course information
  </fo:block>
.. 
</fo:flow>
Links

<fo:block text-align="justify"
    space-before.optimum="12pt"
    font-size="12pt">
    Homepage of the course:
    <fo:basic-link color="blue"
        external-destination
        ="http://coronet.iicm.edu/lectures/mmis2">
        MMIS2 homepage</fo:basic-link>
</fo:block>
Lists

<fo:list-block font-size="12pt">
<fo:list-item>
  <fo:list-item-label
    end-indent="0.5cm" start-indent="0.5cm">
    <fo:block>*</fo:block>
  </fo:list-item-label>
  <fo:list-item-body
    start-indent="body-start()">
    <fo:block>XML Validation</fo:block>
    <fo:block>XML Validation</fo:block>
  </fo:list-item-body>
</fo:list-item>
...
</fo:list-block>
Tables

<fo:table>
<fo:table-header>
  <fo:table-row>
    <fo:table-cell>
      <fo:block>
        Date
      </fo:block>
    </fo:table-cell>
  </fo:table-row>
</fo:table-header>

...</fo:table>
XSLT for producing XSL-FO

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/style/xsl/course2fo.xsl

Defining PDF conversion sitemap.xmap

http://coronet.iicm.edu/mmis2/examples/xml/cocoon/course/sitemap.xmap

Complete Cocoon example (unpack under cocoon webapp)

http://coronet.iicm.edu/mmis2/examples/xml/cocoon.zip
Further Readings

Style articles on www.xml.com
http://www.xml.com/style/

Java and XML articles on www.onjava.com
http://www.onjava.com/topics/java/java_xml

XML Publishing with Cocoon from onjava (Part 1)
http://www.onjava.com/pub/a/onjava/2003/07/02/cocoon.html

XML Publishing with Cocoon from onjava (Part 2)

Chapter 18 from XML Bible on XSL-FO
http://www.cafeconleche.org/books/bible2/chapters/ch18.html
XML Data Management

Denis Helic
How do we store XML data?

As XML documents in the file system

As data in RDBMS that is mapped onto XML documents

- Structured data → easy to separate content and structure (database schema)
- Each data item (e.g. row in a table) has the same structure
The MMIS 2 course example

Course XML document:

http://coronet.iicm.edu/cocoon/mmis2/course/course.xml

Structured data about registered students (mapped onto XML)

http://coronet.iicm.edu/cocoon/mmis2/course/students/students.xml

Very simplified example, since a single document and a single table in the database
How real-life Web applications look like?

Hundreds, thousands, even millions of documents

- Frequent management operations, e.g., add new, delete, update, query, ...

Multiple databases with hundreds or thousands of tables

- Again frequent management operations
What are the consequences of this situation?

Scalability, performance, reliability, data access, ...

DBMS excel regarding these issues

File system has tremendous problems
Representing structured data in XML

Each data item has the same structure

Important to be able to handle various data types

- Integers, strings, floats, dates, ...

Order of XML elements within a data item has no semantic significance

Registered students in MMIS 2

http://coronet.iicm.edu/cocoon/mmis2/course/students/students.xml
DBMS are very good at management of structured data

Mapping onto XML is straightforward

- Ordering is not important
- Map a table (or query results) onto single XML document

Always use DBMS (especially RDBMS) for management of structured data
Representing semi-structured (e.g. documents) data in XML

Data items can have different structure

- A list in XHTML has a different structure than a paragraph
- Even two paragraphs can have different structure (e.g. different sub-elements)

It is not important to handle different data types

Typically textual content

Ordering of elements has semantic significance

http://coronet.iicm.edu/lectures/mmis2/curriculum.html

XHTML is just a single document format, SVG, MathML, ...
Store XML documents in the file system

MMIS 2 course example

- Course XML document contains calendar of events
  [http://coronet.iicm.edu/cocoon/mmis2db/course/course.xml](http://coronet.iicm.edu/cocoon/mmis2db/course/course.xml)

- Resources XML document contains description and links to external resources
  [http://coronet.iicm.edu/cocoon/mmis2db/course/resources.xml](http://coronet.iicm.edu/cocoon/mmis2db/course/resources.xml)
A dynamically created description of the current event

The current event info

- Retrieve it from the course XML document

The description and links of resources for the current event

- Retrieve it from the resources XML document
Managing Document-centric XML: Alternative 1 (3/6)

We need to process both XML documents

Processing the documents includes

- Load the documents
- Parse the documents
- Retrieve the needed info with XPath
- Present it with XSLT

Scalability problems, performance problems, slow data access, ...
Example with Cocoon
http://coronet.iicm.edu/cocoon/mmis2db

XSLT for producing Course and combined HTML
http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/style/xsl/course.xsl

XSLT for producing Resources HTML
http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/style/xsl/resources.xsl

Sitemap Matchers
http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/sitemap.xmap
XSLT for querying documents

http://coronet.iicm.edu/mmis2/examples/xml/xmldb/course/style/xsl/event.xsl

...<xsl:param name="current_event"/>
...
<xsl:template match="courses:event">
  <xsl:if test="normalize-space(courses:title)=$current_event">
    <h2>
      Lecture on <xsl:value-of select="courses:title"/>
    </h2>
  </xsl:if>
</xsl:template>
...
How to pass request parameters to an XSLT style

...<map:transform src="style/xsl/event.xsl">
  <map:parameter name="use-request-parameters" value="true"/>
</map:transform>
...

Sitemap Matchers

http://coronet.iicm.edu/mmis2/examples/xml/xmlldb/course/sitemap.xmap
Managing Document-centric XML: Alternative 2

- Store XML data in a DBMS (e.g. RDBMS)
- How to map semi-structured data onto structured data storage?
- How to keep ordering of elements?
- Multiple tables, multiple joins to ensure correct mapping
- Bad performance, not reliable because of errors!
Working directly with an XML data model

What is an XML data model?

We can think of it as a tree data structure

- Nodes, sub-nodes, leaf nodes (no sub-nodes)
- e.g., an element is a node that has sub-nodes for attributes, content, sub-elements, etc.
Queries

Give me the third sub-element of the element with the name “schedule”

Give me the attribute value of the attribute with the name “href” of the first element

Give me the second processing instruction from this document
Updates

- Store these elements (in exactly this order) as sub-elements of the “course” element
- Insert this processing instruction into this document
- Add this document
- Delete this document
Supporting XML data model with database technology

To improve scalability, performance, data access, reliability

- Use database technology to support XML data model natively
- Native XML interface

Two ways to provide such a support

- XML-enabled databases (At some stage similar to Alternative 2)
- Native XML databases
Native XML databases - Definition (1/2)

Defines a (logical) model for an XML document

- Stores and retrieves documents according to that model
- Must include elements, attributes, PCDATA, and document order
- e.g. XPath, DOM, SAX, etc.
Native XML databases - Definition (2/2)

- Has an XML document as its fundamental unit of (logical) storage
  - e.g. a table (relation) in a RDBMS

- It is not required to have any particular underlying physical storage model
  - Built on relational, OO databases, indexed files!

- Definition comes from XML:DB initiative
Native XML Databases - Generic architecture

Applications/APIs (DOM, SAX, Apache)

Native XML interface

Query manager

Data manager

Index manager

Storage manager

Relational Object-oriented Object-relational File system
Native XML Databases - Collections

- Collections (sets) of documents
- Query all documents in the collection
- Namespace aware
- Indexed documents
  - Better performance, data access, etc.
XPath

Advantages
- If you know XSLT you know XPath
- Intuitive: file system wildcards, patterns, etc.

Disadvantages
- Grouping, sorting, cross document joins,
- Need to be a part of a more powerful language, e.g. XSLT
Recently XQuery

XQuery at W3C

http://www.w3.org/XML/Query

- Working draft status, recommendation expected

- Not only a query language but also a fully-fledged programming language

- Host language for XPath
Slightly similar to SQL

More procedural-oriented language than XSLT

More intuitive??

XSLT is a functional programming language with recursive function calls

In XQuery you make iterations instead of recursions
for $d$ in document("depts.xml")//deptno
let $e := document("emps.xml")//employee[deptno = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
  <big-dept>
    { $d,
      <headcount>{count($e)}</headcount>,
      <avgsal>{avg($e/salary)}</avgsal>
    }
  </big-dept>
So-called FLWOR ("flower") expression

For-Let-Where-Order-Return

With return statement you produce resulting XML

You can produce directly HTML

Hence no need for XSLT
Comparison with XSLT

- Matter of personal preference
- Procedural vs. functional programming
Some problems more easily solved with a recursion (e.g. a lot of mixed content)

```xml
<xsl:template match="resources:resource">
  <li>
    <xsl:apply-templates/>
  </li>
</xsl:template>
<xsl:template match="resources:link">
  <a>
    <xsl:attribute name="href">
      <xsl:value-of select="@xlink:href"/>
      <xsl:value-of select="."/>
    </xsl:attribute>
  </a>
</xsl:template>
```
for $resources in $resources_result/resources:resource
return
<li>
{
 for $nodes in $resources/node()
 return
 if ($nodes/@xlink:href)
 then
   <a href="{$nodes/@xlink:href}">{$nodes/text()}
 else
   $nodes/text()
}
</li>
Some problems more elegant with procedural programming (e.g. if-else-then statement)

```xml
<xsl:choose>
    <xsl:when test="courses:slides/@xlink:href">
        <a>
            <xsl:attribute name="href">
                <xsl:value-of select="courses:slides/@xlink:href"/>
            </xsl:attribute>
            <xsl:value-of select="courses:slides"/>
        </a>
    </xsl:when>
    <xsl:otherwise>
        <xsl:value-of select="courses:slides"/>
    </xsl:otherwise>
</xsl:choose>

Also because XSLT has an XML-based syntax
Weakness of native XML databases

Specific solutions by different vendors

Retrieve whole document, modify, insert again

XUpdate initiative
http://xml.db.org.sourceforge.net/xupdate/index.html

W3C works on update functionality for XQuery
  Will take some time
Example created with two native XML databases and Cocoon

Xindice from Apache

http://xml.apache.org/xindice/

- Query language: XPath

eXist

http://exist.sourceforge.net/

- Query language: XQuery
Xindice comes preinstalled with Cocoon

Installed as a Cocoon block (plug-in)

Data management with a command line tool

http://xml.apache.org/xindice/guide-tools.html

Uses XML-RPC to connect to the database servlet

We will talk about it in Web services lecture
Native XML databases - Example (3/14)

Example for MMIS 2 course installed in Cocoon

http://coronet.iicm.edu/cocoon/mmis2db/course

Specify a title of an event to retrieve the info on that event

The info comes from two documents stored in Xindice database

Course XML document: the info about date, place, ...

Resources XML document: the info about links
Query Course XML document

```xml
<map:match pattern="course_query.xml">
  <map:match type="request-parameter" pattern="xpath">
    <map:generate
      src="xmlmdb:xindice-embed://db/mmis2_course/#{1}"/>
    <map:serialize type="xml"/>
  </map:match>
</map:match>
```
Query both documents and integrate the results

```xml
<map:match pattern="event_query.xml">
  <map:match type="request-parameter" pattern="current_event">
    <map:aggregate element="aggregated-content">
      <map:part src="xmldb:xindice-embed://db/mmis2_course/#
        //courses:event[normalize-space(courses:title)='\{1\}']"/>
      <map:part src="xmldb:xindice-embed://db/mmis2_resources/#
        //resources:event[normalize-space(resources:name)='\{1\}']"/>
    </map:aggregate>
    <map:serialize type="xml"/>
  </map:match>
</map:match>
```
Query both documents, integrate the results, present as HTML

```xml
<map:match pattern="event_query.xml">
  <map:match type="request-parameter" pattern="current_event">
    <map:aggregate element="aggregated-content">
      <map:part src="xmldb:xindice-embed:///db/mmis2_course/#
        //courses:event[normalize-space(courses:title)='{1}']"/>
      <map:part src="xmldb:xindice-embed:///db/mmis2_resources/#
        //resources:event[normalize-space(resources:name)='{1}']"/>
    </map:aggregate>
    <map:transform src="style/xsl/event_query.xsl"/>
    <map:serialize type="html"/>
  </map:match>
</map:match>
```
XSLT for producing HTML

http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/style/xsl/event_query.xsl

<xsl:template match="courses:event">
  <h2>
    Lecture on <xsl:value-of select="courses:title"/>
  </h2>
  ...
</xsl:template>
eXist native XML database

- Does not come preinstalled with Cocoon

Install it as a Cocoon block

http://wiki.apache.org/cocoon/EXistAsBlock

Examples, administration available directly from Cocoon

http://coronet.iicm.edu/cocoon/samples/blocks/exist/index.xml

XQueryGenerator a special type of Cocoon generator
Example for MMIS 2 course installed in Cocoon

http://coronet.iicm.edu/cocoon/mmis2db/course

Both documents in one eXist collection

Query both of the documents with an XQuery

Retrieve the info from both documents, integrate it and present it
Simple XPath query submitted

xquery version "1.0";
declare namespace util="http://exist-db.org/xquery/util";
declare namespace request="http://exist-db.org/xquery/request";
<results>
{
    let $query := request:request-parameter("xpath", ())
    let $result := util:eval($query, "/db/mmis2")
    return $result
}
</results>

Complete XQuery file

http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/content/exist_query.xq
XQueryGenerator in sitemap.xmap

```xml
<map:match pattern="exist_query.xq">
  <map:generate src="content/exist_query.xq" type="xquery"/>
  <map:serialize type="xml"/>
</map:match>
```

Sitemap.xmap

```
http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/
sitemap.xmap
```
Querying both documents for an event

...

<Result>
{
...

... "//resources:event[resources:name = '$current_event', '$']
let $result := util:eval($query, '/db/mmis2')
return $result
}
</Result>

....

Complete XQuery file

http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/content/exist_event_query.xq
Presenting results as XML or as HTML

```xml
<map:match pattern="exist_event_query.xq">
  <map:generate src="content/exist_event_query.xq" type="xquery"/>
  <map:serialize type="xml"/>
</map:match>

<map:match pattern="event_query.xq">
  <map:generate src="content/exist_event_query.xq" type="xquery"/>
  <map:transform src="style/xsl/event_query.xsl"/>
  <map:serialize type="html"/>
</map:match>
```

XSLT for producing HTML

```
http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/
   style/xsl/event_query.xsl
```
Producing HTML directly from XQuery

http://coronet.iicm.edu/mmis2/examples/xml/xmlmdb/course/content/current_event.xq

<map:match pattern="event.xq">
  <map:generate src="content/current_event.xq" type="xquery"/>
  <map:serialize type="html"/>
</map:match>
Further Readings (1/2)

XML Data Stores: Emerging Practices

Introduction to Native XML Databases
http://www.xml.com/pub/a/2001/10/31/nativexmldb.html

Getting Started with XQuery (Part 1)
http://www.xml.com/pub/a/2005/03/02/xquery.html
Getting Started with XQuery (Part 2)

Database articles on xml.com
http://www.xml.com/databases/

Simple XQuery Tutorial
Web Application Frameworks - MVC

Denis Helic
Core design pattern of all software engineering

Easily supported by object-oriented software development

Definition by Edsger Dijkstra

We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day [...] But nothing is gained - on the contrary - by tackling these various aspects simultaneously. It is what I sometimes have called “the separation of concerns” [...] “On the role of scientific thought”
Edsger Dijkstra one of the most influential computer scientists

Paper: Go To Statement Considered Harmful

Algorithm: Shortest-path algorithm

Book: A discipline of programming

Award: The ACM Turing Award
You want to isolate different aspects of a software application from each other

- You can work on each aspect in details
- You can be consistent within each aspect
- At a single moment you work on a single aspect (e.g., you are not distracted with other aspects)
- Also, a basis for the team work (e.g., different teams work on different aspects)
Divide-and-conquer method of designing algorithms

Applied to developing software applications

OO principles easily support SOC

Classes, objects, encapsulation isolate one aspect from another

If interfaces are clearly defined you can work on different aspects in isolation
Separation of Concerns (5/5)

Benefits of SOC

- Improved comprehension of the application domain
- Reduced complexity
- Component (aspect) integration
- Reuse
- Adaptibility, customization (through component exchange)
User-oriented database applications - SOC(1/9)
The interface between UI and Data Management is crucial!

It is where the SOC is violated most

Why is this so?

An example: You have a database containing info about the students registered for a course

This info is composed of name, matricel number and study field
In your process logic you have a Student class

- The Student class has getter and setter methods
  - getName(), getStudyField(), etc.

In your UI script (e.g., a PHP script) you retrieve a list of Student objects

- You iterate through the list and use the getter methods to write info in an HTML table
Your customer tells you that there is another external student database

It contains contact info such as student addresses, e-mails, etc.

You customer wants to have students’ e-mail addresses in the list of the registered students

You are too lazy and want to accomplish it fast

In the UI script you use the matrikel number of each student

You connect to the external database and retrieve the needed info
You mix UI and data management

Consequences can be tremendous: different UI scripts need to be updated, maintained, etc.

Proper way of implementing such changes

- Modify the Student class and extend it with `getEMail()`, `setEMail()`
- In the DM module connect to the external database and populate student objects with `setEMail()`
User-oriented database applications - SOC(6/9)

- SOC is supported by OO programming languages but it is not enforced!
  - Developers need to take care about this!
- Scripting languages are even more vulnerable
  - You do not need compiling, it is fast to make such changes!
- But, Java is vulnerable too, here you need to take care also!
- Recollect, the clear solution that we had with XML publishing
  - The UI gets an XML document that can only be presented with XSLT!
User-oriented database applications - SOC(7/9)
One way to improve the situation: a layered architecture

The UI communicates only with the PL module

Again, hard to enforce this

You can only hope that developers will follow the principle!
User-oriented database applications - SOC(9/9)
Model-View-Controller is a particular design pattern that supports SOC

It was invented in the early days of GUIs

To decouple the graphical interface from the application data and logic

Invented at Xerox Parc in the 70's
The first appearance in Smalltalk-80

One of the first OO languages

Pure OO language, i.e., everything is an object

MVC invented by Trygve Reenskaug

Original documentation

http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html
Model-View-Controller (3/11)
Controller

- Handles user input (e.g., mouse clicks, keyboard, ...)
- Updates the model
- Instructs the view to redraw itself
View

- Presents the model in a specific way
- Note different views for the same model
- Very important not only in GUIs but also in Web applications (e.g., XHTML, PDF, etc.)
Model

- Contains the data and application logic
Very easily accomplished with an OO programming model

- Objects encapsulate the data
- Objects implement behaviour (as methods)
- Interaction between different objects (i.e., invoking methods) supports the application logic
Where does the data come from?

If in memory → everything is covered

If in the file system or in a DBMS

We need a special Data Management module
A special case of the MVC uses Observer design pattern
When the model changes it notifies the views about the change

All views redraw as the result of the notification

Not applicable in a Web application

- Page-oriented applications
- You need a user request for each particular view (e.g., HTML, PDF,..)
MVC: An example with Observer pattern (1/11)

- A simple GUI Java application using MVC
- The example uses Observable/Observer pair from java.util
- Subclasses of the Observable class can notify observers about changes in their state
- Implementation classes of Observer need to implement update() method to react to the changes
- Observer is a Java interface
MVC: An example with Observer pattern (2/11)

Model

- SimpleModel holding a single integer value
- getValue() and setValue() methods to manipulate the model
- The value must be in the range [0,100]
final public class SimpleModel extends Observable {
...
    public void setValue(int value) {
        if ((value > 100) || (value < 0)) {
            throw new IllegalArgumentException("The value must be ..");
        }
        value_ = value;
        setChanged();
        notifyObservers();
    }
...
}

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleModel.java
MVC: An example with Observer pattern

**View**

- Holds a reference to the model
- Manages a GUI widget to represent the model value
- Reacts to change notifications from the model
- It is an abstract class, where subclasses manage different GUI widgets
public abstract class SimpleView implements Observer {
    protected JComponent widget_;  
    public void update(Observable observable, Object arg) {
        updateView();
    }
    abstract public void updateView();
...
}

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleView.java
final public class SimpleTextFieldView extends SimpleView {
    private JTextField value_field_ = new JTextField();
    public SimpleTextFieldView(SimpleModel model) {
        ...
        widget_ = new JPanel(new BorderLayout());
        ...
        widget_.add(value_field_, BorderLayout.SOUTH);
    }
    public void updateView() {
        value_field_.setText("" + model_.getValue());
    }
}

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleTextFieldView.java
public class SimpleSliderView extends SimpleView {
    private JSlider value_slider_ = new JSlider();
    public SimpleSliderView(SimpleModel model) {
        ...
        widget_ = new JPanel(new BorderLayout());
        ...
        widget_.add(value_slider_, BorderLayout.SOUTH);
    }
    public void updateView() {
        value_slider_.setValue(model_.getValue());
    }
}

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleSliderView.java
Controller

Java Event System follows already MVC, i.e., it is the controller

The event capturer forwards a captured event (e.g. a mouse click) to the event dispatcher

The event dispatcher manages a mapping of events to event listeners

The event dispatcher finds the proper listener and invokes its actionPerformed() method

We need only to write listeners, register them with GUI widgets and implement action() methods
MVC: An example with Observer pattern (9/11)

final public class SimpleTextFieldView extends SimpleView {
    public SimpleTextFieldView(SimpleModel model) {
        ...
        value_field_.addActionListener(new TextFieldControllerAction());
        ...
    }

    class TextFieldControllerAction implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            model_.setValue(Integer.parseInt(value_field_.getText()));
        }
    }
}

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleTextFieldView.java
public class SimpleSliderView extends SimpleView {
    public SimpleSliderView(SimpleModel model) {
        ...
        value_slider_.addChangeListener(new SliderControllerAction());
        ...
    }
}

class SliderControllerAction implements ChangeListener {
    public void stateChanged(ChangeEvent event) {
        model_.setValue(value_slider_.getValue());
    }
}

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/SimpleSliderView.java
MVC: An example with Observer pattern

```java
public static void main(String[] args) {
    SimpleModel model = new SimpleModel();
    SimpleView view = new SimpleTextFieldView(model);
    ...
    view = new SimpleSliderView(model);
    ...
    model.setValue(12);
}
```

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/simple/edu/iicm/mmis2/mvc/MVCMain.java

Complete source code

http://coronet.iicm.edu/mmis2/examples/mvc/simple.zip
The server receives different HTTP requests
- Requests can include different parameters submitted by the user
- On the basis of these parameters the server produces the response
- The server can dispatch the request to different handlers (actions)
- The server needs a registry of mappings of parameters onto actions

The server, registry, dispatcher and actions are the Controller
The Model is accessed from the actions

It is data and behaviour encapsulated within objects

Each action is associated with a View

When the action finishes the Controller invokes the View

The View accesses the Model, retrieves (!) the data and present it

The registry includes also associations between actions and views
Struts - Java-based MVC Web App Framework

An Open Source Apache project
http://struts.apache.org/

Web development framework supporting MVC and SOC
http://struts.apache.org/userGuide/introduction.html#mvc

It follows the design described above

The controller is the Struts ActionServlet + Actions

The model is your application data and logic

The views are JSP pages

Everything defined in struts-config.xml
Struts - Java-based MVC Web App Framework (2/8)
Struts - Java-based MVC Web App Framework

Download

http://struts.apache.org/download.cgi

You can download either binaries or sources

In the installation included: struts-blank.war

→ You can use it for a quick start with Struts

→ Only edit struts-config.xml and define your own actions

→ Everything else is in place
Defining actions in struts-config.xml has a number of steps

1. Defining a URL pattern for that action (pattern + .do)
2. Defining a so-called ActionForm which encapsulates the user parameters
   - There is a special JavaBean class that extends ActionForm class
   - This class provides getters and setters for the parameters (same names)
3. Defining a number of forwards to JSP pages to present the results
<action-mappings>
...

<action
    path="/search"
    type="edu.iicm.publication.struts.SearchAction"
    name="search_form"
    scope="request"
    validate="false"
    input="/search.jsp">
    <forward name="html" path="/search_results.jsp"/>
    <forward name="bibtex" path="/search_export.jsp"/>
    <forward name="rdf" path="/search_export.jsp"/>
</action>
...

</action-mappings>
If validate set to true then you can validate that the user parameters have the proper values

Need to implement the validate() method inside your ActionForm subclass

If validation fails

The user is redirected to JSP specified in the input attribute
Name parameter refers to the ActionForm subclass

```xml
<form-beans>
  ...
  <form-bean
    name="search_form"
    type="edu.iicm.publication.struts.SearchForm"/>
  ...
</form-beans>
```
Additionally, Java classes need to be implemented

ActionForm subclass encapsulating parameters

A subclass of the Action class which manipulates the Model objects

Here you also decide to which JSP page to forward

Finally, a number of JSP pages to present the results
Web-based database of scientific publications

- Used for managing of personal publication databases

Retrieving of publications in different formats

- HTML, BibTeX, XML

Searching for specific publications

- Type, year, title keywords
A special user as administrator
  - Can add authors
  - Can add publications
  - Delete, update!

Importing publications in different formats
  - BibTeX, XML
Managing of publications

Each publication has

- One or more authors
- Title
- Year of publishing
- Optionally URL
- Type (depends on format)
BibTeX format \url{http://www.din1505.informationskompetenz.net/}

Types such as Article, Conference, Book, etc

Depending on type different additional attributes

- Article has a journal
- Book has a publisher, etc.

BibTeX format mapped onto a relational database (e.g., a number of relations)
Here we just want to investigate the Struts part of the example

Search form, submitting queries

- Results in different formats

Administration interface

- Registering, validating inputs, etc.
Right now we have the Model

The Model includes classes for publications

The Model and the DM are separated

The lecture on O/R mapping will show how this is done

The Model

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/
Search functionality

...<action
    path="/search"
    type="edu.iicm.publication.struts.SearchAction"
    name="search_form"
    scope="request"
    validate="false"
    input="/search.jsp">
    <forward name="html" path="/search_results.jsp"/>
    <forward name="bibtex" path="/search_export.jsp"/>
    <forward name="rdf" path="/search_export.jsp"/>
</action>
...
ActionForm

<form-beans>
  ...
  <form-bean
    name="search_form"
    type="edu.iicm.publication.struts.SearchForm"/>
  ...
</form-beans>

SearchForm bean

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/struts/SearchForm.java
public final class SearchAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
        String type = ((SearchForm) form).getType();
        ...
        String format = ((SearchForm) form).getFormat();
    }
}
Iterator pubs = dao.readAllPubs(type, year, title);
while (pubs.hasNext()) {
    Publication pub = (Publication) pubs.next();
    
    String result = new String(buffer);
    request.setAttribute("result", result);
    return mapping.findForward(format);
}
JSP views

```jsp
<%@ include file="/jsp_layout/header.jsp" %>
<h2 class="hrthinbox">
All Publications
</h2>
<ul>
<%= request.getAttribute("result").toString() %>
</ul>
<div>
<a href = "search.jsp">Search again</a>
</div>
<%@ include file="/jsp_layout/footer.jsp" %>
```

Complete view

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/search_results.jsp.txt
The example

http://coronet.iicm.edu/struts-publicationdb

Search functionality

- You can select different criteria
- You can select different output formats
To have different output formats we use Visitor pattern

Publication is an abstract class with an abstract write method

Subclasses (Article, Book, ...) invoke an abstract method from an abstract Visitor

They pass the reference to itself to the visitor

Subclasses of Visitor implement the method

Write out the proper format
New format → new Visitor

Creation of Visitor with Abstract Factory

Allows you to use the same code to write out HTML, BibTeX, ...

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/writer
Administration functionality

- Adding, deleting, updating, importing publications

User management

- The user needs to log-in
- The user info need to be always available
- The user info should be used to check the rights
Inserting a publication requires a number of steps

Select a number of persons that are involved in the publication

Select the type of the publication

Enter the publication data (depends on the type)
Session management (continued)

- Each step is a separate HTTP request
- We need to manage the data over a number of HTTP requests
- The data is disparate (e.g., persons, pub type, etc.)
- We need special Java objects that encapsulate the needed info
HTML form validation

- At different steps different forms
- Different forms for different publication types
- A special login form

Apply form bean validation provided by Struts
User management

- Login with a special login form
- Logout
- User information should be available for all administration actions
- Store it in the session object
User info

public final class User {
    private String username_;  
    private String password_; 
    public User(String username, String password) {
        setUsername(username);  
        setPassword(username); 
    }
    ...
}

public final class User {

... public void setPassword(String password) {
    if((password == null) || (password.length() == 0)) {
        throw new IllegalArgumentException(...);
    }
    password_ = password;
}
...
}

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/struts/User.java
Login Action

```xml
<action
  path="/login"
  type="edu.iicm.publication.struts.LoginAction"
  name="login_form"
  scope="request"
  validate="true"
  input="/login.jsp">
  <forward name="success" path="/manage.jsp"/>
  <forward name="failure" path="/login.jsp"/>
</action>
```
Login form bean

...<form-bean
   name="login_form"
   type="edu.iicm.publication.struts>LoginForm"/>
...

Login form bean

```java
public final class LoginForm extends ActionForm {
    private String username_ = null;
    private String password_ = null;
    
    public String getPassword() {
        return password_;}
    }
    public void setPassword(String password) {
        password_ = password;
    }
    
    ...
}
```
Login form validation

```
public ActionErrors validate(ActionMapping mapping, 
HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();
    if((username_ == null) || (username_.length() == 0)) {
        errors.add("username",
            new ActionMessage("errors.username.required"));
    }
    ...
    return errors;
}
```

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/struts/LoginForm.java
Error messages handling

- All messages have keys, e.g. `errors.password.required`

... 
`errors.username.required=Please enter your username!`
`errors.password.required=Please provide your password!`

... 

- All keys and values stored in a special file

  [Source](http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/MessageResources.properties)
Login Action

Reads username and password from the LoginForm

- Everything should be available since we have validation

Compares username and password with a user database

- If authentication is OK creates and stores a User object
User Database

Encapsulated as a Java object

Public interface provides a single method: checkUsername()

A particular implementation manages usernames and passwords

In an XML file, a database, memory, etc
Very simple solution in the example

Hardcoded in a hash table

```java
public final class UserDatabase {
    private HashMap users_;
    ...
    private void init() {
        users_ = new HashMap();
        users_.put("dhelic", "dhelic");
    }
    ...
}
```
public final class UserDatabase {
    ...
    public boolean checkUser(String username, String password) {
        String db_password = (String) users_.get(username);
        if(db_password == null) {
            return false;
        }
        if(db_password.equals(password)) {
            return true;
        } else {
            return false;
        }
    }
    ...
}
We need very often a UserDatabase object

- Each time a user wants to log in
- We can use a shared single instance of the UserDatabase class
- More efficient (e.g. created and loaded only once)
- More secure because nobody can create another instance
Singleton design pattern

Managing and enforcing a single shared instance of a class

```java
public final class UserDatabase {
    private static UserDatabase instance_ = null;
    
    public static UserDatabase getUserDatabase() {
        if(instance_ == null) {
            instance_ = new UserDatabase();
        }
        return instance_;}
}
```
How to enforce that nobody creates another instance

Declare the constructor as private

```java
public final class UserDatabase {

... 

private UserDatabase() {
    init();
}

... 

}
```

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/struts/UserDatabase.java
public final class LoginAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                  HttpServletRequest request, HttpServletResponse response) {

        String username = ((LoginForm) form).getUsername();
        String password = ((LoginForm) form).getPassword();
        ActionMessages errors = new ActionMessages();
        if(!UserDatabase.getUserDatabase().checkUser(username, password)) {
            errors.add(ActionErrors.GLOBAL_MESSAGE,
                        new ActionMessage("errors.authentication.failed"));
        }

    }

    ...
}

Struts - An Example(34/42)
public final class LoginAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) {
        ...
        if(!errors.isEmpty()) {
            saveErrors(request, errors);
            return mapping.findForward(Constants.FAILURE);
        }
        ...
    }
}
public final class LoginAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        ...
        User user = new User(username, password);
        HttpSession session = request.getSession();
        session.setAttribute(Constants.USER_KEY, user);
        return mapping.findForward(Constants.SUCCESS);
    }
    ...
}

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/struts/LoginAction.java
What is BaseAction?

It is always good to create an abstract BaseAction as a subclass of Struts action

Declare an abstract execute() method

```java
public abstract class BaseAction extends Action {
    ...
    public abstract ActionForward execute(ActionMapping mapping,
            ActionForm form,
            HttpServletRequest request,
            HttpServletResponse response);
    ...
}
```
Advantages of an abstract base action

- You can implement a number of common methods here
- You can use these methods within all subclasses

User management as example

- All management actions need to check if user is authenticated
public abstract class BaseAction extends Action {
    protected boolean isUserAuthenticated(HttpServletRequest request) {
        HttpSession session = request.getSession();

        ... 

        User user = (User) session.getAttribute(Constants.USER_KEY);
        if (user == null) {
            return false;
        }
        return true;
    }
}

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/struts/BaseAction.java
Logout Action

...<action
  path="/logout"
  type="edu.iicm.publication.struts.LogoutAction">
  <forward name="success" path="/manage.jsp"/>
</action>...

...
The action removes the user object and invalidates the session

```java
public final class LogoutAction extends BaseAction {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                   HttpServletRequest request, HttpServletResponse response) {

        HttpSession session = request.getSession();
        session.removeAttribute(Constants.USER_KEY);
        session.invalidate();
        return mapping.findForward(Constants.SUCCESS);
    }
}
```
Example

http://coronet.iicm.edu/struts-publicationdb

Example Zip

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb.zip

Example WAR

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb.war
Ruby is a pure object-oriented programming language

Pure means everything is an object, e.g. the number 1 is an instance of class Fixnum

Interpreted scripting language

Dynamically, weakly typed

Single inheritance, but can be extended with so-called modules (similar to Java interfaces)

Rich text processing functionality (similar to Perl)
Rails is open source Web application framework

http://www.rubyonrails.org/

Supports development of database-backed application

User-oriented Web database applications

Follows MVC architecture and design pattern
Three main guiding principles

Model-driven (domain-driven) development
- You start with a data model and add the functionality, controllers, views on top of it

Convention over configuration
- Set of naming conventions (similar to JavaBeans but more in depth)

Less software, i.e., less code
- Generating default code that you adjust to fit your needs
Domain-driven development

Based on an ORM framework called ActiveRecord

http://wiki.rubyonrails.com/rails/pages/ActiveRecord

ActiveRecord is a generic ORM framework

Similar to Hibernate

Uses a naming convention to provide the default mapping

You can adjust the default mapping if you need to
ActiveRecord naming convention

- [http://ar.rubyonrails.com/classes/ActiveRecord/Base.html](http://ar.rubyonrails.com/classes/ActiveRecord/Base.html)

Names of classes and tables

- Give names to your classes as English singular, and to your tables as English plural
- Start the name of the class with an upper case, all other letters are lower case (Student)
- Table name is all lower case (students)
Student class

```ruby
class Student < ActiveRecord::Base
end
```

Student table

```sql
create table students (  
id int not null auto_increment,  
name varchar(80),  
study_field varchar(10),  
primary key(id)
);
```
Names of the table columns and instance variables

- Map 1-to-1, i.e., student.name maps to name column in students
- Primary key must be named id in the table

Immediately you can use all methods from ActiveRecord

```ruby
@students = Student.find_all
@student = Student.new
```
Internally, ActiveRecord uses a single table to map the whole class hierarchy.

**Single Table Inheritance**

To map associations a simple domain language-like set of macros is used.

For example, `belongs_to`, `has_many`, etc.

**ActiveRecord::Associations module**

---

Ruby on Rails (8/14)
class Student < ActiveRecord::Base
  has_and_belongs_to_many :courses
end

Connects two classes via an associative table (many-to-many relation)
The name of the table: courses_students
The names of foreign keys: course_id, student_id
Convention over configuration

- A lot of examples in ActiveRecord
- Further examples in controller module of MVC

Controllers are modules that handle user requests

Convention on mapping of URLs onto methods in controllers

- Much easier than Struts configuration
URL = /controller_class_name/controller_method_name

For example, http://localhost:3000/test/index

Another example: http://localhost:3000/test/hello

class TestController < ApplicationController
  def index
    render_text "Wow, that was easy"
  end

  def hello
    render_text "Hello World"
  end
end
Whenever you have a model class, e.g., Student class you can use a so-called CRUD scaffold

create, read, update, delete methods

These methods are provided by the ActiveRecord

```ruby
class StudentController < ApplicationController
  scaffold :student
end
```
This single line embeds all of the CRUD methods into controller

Consequently, they are immediately visible through URLs

Read:  http://localhost:3000/student/show/1
Update:  http://localhost:3000/student/edit/1
List:  http://localhost:3000/student/list

Note how meaningful and consistent URLs are (we discussed this in the first lecture)
By scaffolding you also get default views

However, you can adjust them

For a particular controller method, e.g., show() you need to create show.rhtml

Another naming convention
Courses and students registered for courses

Two entities: Course and Student

Many-to-many relation between Course and Student
  - One student might register for a number of courses
  - For each course there is a number of registered students
First step with Rails: Create database (MySQL)

Follow the naming convention

```sql
create table courses (
    id int not null auto_increment,
    title varchar(200),
    url varchar(200),
    description text,
    primary key(id)
);
```
create table students (
    id int not null auto_increment,
    name varchar(80),
    study_field varchar(10),
    primary key(id)
);

create table courses_students (
    course_id int not null,
    student_id int not null,
    primary key(course_id, student_id)
);
Second step with Rails: Generate app skeleton

Invoke: `rails courses`

You will get a directory structure:

```
app/
config/
db/
doc/
lib/
log/
public/
script/
test/
vendor/
```
Within app directory you get something like:

```
app/
  controllers/
  helpers/
  models/
  views/
```

Edit config/database.yml

Start the app, invoke: ruby script/server
Third step: generate model and controller

- Invoke: ruby script/generate model Course
- Invoke: ruby script/generate model Student
- Invoke: ruby script/generate controller Course
- Invoke: ruby script/generate controller Student
Use the scaffold (the generated default controller and views)

class CourseController < ApplicationController
  scaffold :course
end

class StudentController < ApplicationController
  scaffold :student
end

Fourth step: Work (!) and adjust the app!
Adjusting the app

Create alternative views

- Update the method in controller, e.g. the show method
- Create the show.rhtml in views and adjust it
<p><b>Title</b><br>
  <%= @course.title %></p>

<p><b>URL</b><br>
  <%= link_to @course.url, @course.url %></p>

<p><b>Description</b><br>
  <%= @course.description %></p>
Implement the association

class Student < ActiveRecord::Base
  has_and_belongs_to_many :course
end
Adjust edit.rhtml and update method in controller

```ruby
<% for @course in @courses %>
  <% if @student.courses.include? @course %>
    <%= @course.title %>
    <input type="checkbox" name="<%= @course.title %>" value="0" checked="checked"/>
  <% else %>
    <%= @course.title %>
    <input type="checkbox" name="<%= @course.title %>" value="0"/>
  <% end %>
<% end %>
```
def update
  @courses = Course.find_all
  @student = Student.find(@params[:id])
  if @student.update_attributes(params[:student])
    @student.courses.clear
    for course in @courses
      if (@params[course.title])
        @student.courses<<course
      end
    end
  end
...

Example:  http://coronet.iicm.edu/mmis2/examples/mvc/rails/courses.zip
Ruby on Rails: Advances Features

- Defining layouts (headers, footers, ...)
- Modules for standard functionality, e.g., authentication
- Caching
- Validation and callbacks
- Transactions
- Testing
Further Readings

MVC Wiki with MVC History
http://c2.com/cgi/wiki?ModelViewController

J2EE Patterns: MVC

Introduction to Jakarta Struts Framework from onjava.com

Learning Jakarta Struts, part 2 from onjava.com
http://www.onjava.com/pub/a/onjava/2001/10/31/struts2.html
Learning Jakarta Struts, part 3 from onjava.com

Struts articles from JSP and Servlets from onjava.com
http://www.onjava.com/topics/java/JSP_Servlets

Struts Best Practices from javaworld.com

Struts Tutorial
Further Readings (3/4)

- Rolling with Ruby on Rails
  http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html

- Rolling with Ruby on Rails, Part 2
  http://www.onlamp.com/pub/a/onlamp/2005/03/03/rails.html

- Really Getting Started in Rails

- Rails Documentation
  http://www.rubyonrails.org/docs
Further Readings

Rails Wiki
http://wiki.rubyonrails.com/rails

Four Days on Rails
http://www.rails4days.pwp.blueyonder.co.uk/Rails4Days.pdf

Fast-track your Web apps with Ruby on Rails
Data Management in Web Applications

Denis Helic
Classifying data according to its structure

Structured data

- Clear separation between content and structure
- Each data item has same structure
- E.g., registered students, publications, etc.
Semi-structured data

- Separation between content and structure
- Structure can differ from a data item to a data item
- E.g., documents (XHTML, SVG, PDF, ...)
- E.g., a paragraph can include only text, but also links, lists, tables, etc..

Managing of semi-structured data - Native XML databases
Managing of structured data → DBMS, especially RDBMS

ACID databases test

- Atomicity (transactions, either all or none)
- Consistency (data integrity)
- Isolation (transactions independent on each other)
- Durability (permanent storage)
Declarative query language SQL

- You tell the system what do you want not how to do it!
- Less errors!

Maturity of the technology

- Mature products, know-how, experience, support, etc.
Architecture of User-oriented Database applications
Problem with OO systems and RDBMS (1/8)

- Process logic module (Model) has two submodules (OO technology)
- Application logic (behaviour) of the system
  - Methods you can invoke on the objects
  - Highly dynamic, objects sending messages to each other
- Data part
- RDBMS has only data
  - Tables, fields, relations between tables
Difference between OO paradigm and relational paradigm

OO paradigm models the real world as a dynamic entity
  - Object relations (inheritance, composition, etc.) model the real world
  - Through the object interactions this entity changes its state frequently

Relational paradigm models a single state of the real world using only data
  - With external (declarative) languages you can alter this state
The problem is how to persist objects, their relations, and dynamics within tables

E.g., how to persist inheritance?

E.g., when to persist (each time when something happens in the OO system)?

It is a question of mapping OO systems to RDBMS

This problem is known as Impedance Mismatch
Simple example from Agile Database Techniques by Scott W. Ambler

We have an order and each order has a number of order items

Order has attributes such as dates, taxes, etc.

Order item has the number of ordered items

Behaviour: you can cancel order, get total taxes, etc.
Problem with OO systems and RDBMS (5/8)

<<Class Model>>

Order
- dateOrdered: Date
- dateFulfilled: Date
- federalTax: Currency
- stateTax: Currency
- localTax: Currency
- subtotalBeforeTax: Currency

+ cancel()
+ getTaxes(): Vector
+ getTotal(): Currency
+ ship()
- calculateTaxes()
- scheduleShipment()

<<Physical Data Model>>

Order
- OrderId: INT24 <<PK>>
- DateOrdered: Date
- DateFulfilled: Date
- Tax: Float
- SubtotalBeforeTax: Float
- ShipToContactID: INT24 <<FK>>
- BillToContactID: INT24 <<FK>>
- LastUpdate: TimeStamp

OrderItem
- NumberOrdered: int
- ItemSequence: INT24 <<PK>>
- ItemNo: INT24 <<FK>>
- NumberOrdered: INT24
- LastUpdate: TimeStamp
The ideal solution is to have one-to-one mapping of classes onto tables.

In this simple case this is possible.

In a general case this is not always possible.

The second step is to map class attributes onto table fields.

dateFulfilled from the class onto DateFulfilled from the table.
Problem with OO systems and RDBMS(7/8)

However, even in this simple case there are differences

Several tax attributes in the class, a single field in the table
  Calculation is needed to implement this mapping

Primary keys, foreing keys in tables, references in objects
  To implement the mapping correctly objects need to know about keys

Different data types
  Currency type compared to float type
But why model it in this way?

Why do not we model it so that there is no such differences?

Legacy systems, there is already a database schema, and you are building an OO system around that schema.

If the system is new then we can take more care

The best way to build the OO schema first

Let the OO schema drive the implementation of the database schema

The OO model is the model that your application works with!
The mapping consists of three steps

Mapping attributes onto columns

An attribute maps onto zero or more columns in a table

Mapping classes onto tables

Handling of inheritance hierarchies

Mapping of associations, aggregation (composition)

By relating tables to each other
Mapping classes (and class hierarchies) onto tables

- Person
  - name_: String

- Student
  - matrikel_number_: int

- Teacher
  - salary_: int
Four possible ways to map a class hierarchy

- Using one table for an entire class hierarchy
- Using one table per subclass
- Using one table per class
- Using one table per superclass and one table for subclasses
Using one table for an entire class hierarchy

<table>
<thead>
<tr>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>peid</td>
</tr>
<tr>
<td>type</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>mat_number</td>
</tr>
<tr>
<td>salary</td>
</tr>
</tbody>
</table>

**Mapping objects onto relational databases**

(4/15)
Advantages

- Very simple
- Changing person roles is very simple
- Query performance is good (everything is in one table)

Disadvantages

- Adding a new attribute anywhere in the hierarchy → modifying the table
- A lot of empty fields in the table
- One person can only have a single type
Using one table per subclass

<table>
<thead>
<tr>
<th>Student</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid &lt;&lt;primary key&gt;&gt;</td>
<td>tid &lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>mat_number</td>
<td>salary</td>
</tr>
</tbody>
</table>
Mapping objects onto relational databases (7/15)

Advantages

- Simple
- Query performance is good (everything is in one table)

Disadvantages

- Adding a new attribute to the superclass → modifying all the tables
- Person changes roles → copying the data between the tables
- Person can have multiple roles, but hard to maintain integrity
Using one table per class

**Person**
- **peid** (primary key)
- name

**Student**
- **peid** (primary key) as foreign key
- mat_number

**Teacher**
- **peid** (primary key) as foreign key
- salary
Mapping objects onto relational databases (9/15)

Advantages

- Conforms to object-oriented concepts (the data stored in the tables that correspond to the types)
- Easy to extend the class hierarchy

Disadvantages

- Query performance is not that good (a lot of joins)
- You need a lot of SQLs to implement different views on the data
Using one table per superclass and one table for subclasses

<table>
<thead>
<tr>
<th>Person</th>
<th>PersonAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>peid</td>
<td>peid</td>
</tr>
<tr>
<td>type</td>
<td>&lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td>name</td>
<td>&lt;&lt;foreign key&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>key &lt;&lt;primary key&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>value</td>
</tr>
</tbody>
</table>
Mapping objects onto relational databases

Advantages

- Simple
- Query performance is OK (the data is in two tables)

Disadvantages

- A lot of data type conversion (different attributes are all mapped onto a single data type)
- One person can only have a single type (move this into another table, etc.)
Mapping of associations and aggregation (composition)

Implemented by relating tables to each other with primary and foreign keys

Aggregation (composition) represents whole/part relationship

  Composition a bit stronger (components belong to a single owner)
  Aggregation allows that components are shared

Associations just relate one object to another
Mapping objects onto relational databases

Mapping aggregation (composition)
- At database implementation level
- Everything that you do to the whole you always do to the parts
  - E.g., load, update, delete, etc.

Mapping associations
- It is different from a case to a case
Mapping objects onto relational databases (14/15)

Mapping of many-to-many associations

E.g., a teacher may work in a number of schools

E.g., a school has a number of teachers

[Diagram showing a many-to-many relationship between Teacher and School, with properties salary: int for Teacher and address: String for School]
The table in the middle is usually called the associative table.
Managing of publications

Each publication has
  
  One or more persons involved (authors or editors)

  Title

  Year of publishing

  Optionally URL

  Type (depends on format)
BibTeX format: [http://www.din1505.informationskompetenz.net/](http://www.din1505.informationskompetenz.net/)

Types such as Article, Conference, Book, etc.

Depending on type different additional attributes

- Article has a journal
- Book has a publisher, etc.
The Model

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/

Publication superclass

- Holds 4-5 attributes that are common for all publication types
- All publication types are subclasses of the superclass (14 types)
- Each publication type has approximately 3-4 additional attributes
Which alternative for mapping class hierarchies to choose?

Alternative 1: a single table
- We will have a lot of fields (approx. 4 + 14*3)
- For each record the most of the fields would be empty

Alternative 2: a table per subclass
- 14 tables is too much for such a simple model
Alternative 3: a table per class

- 15 tables is too much for such a simple model

Alternative 4: a table for superclass and a single table for all subclasses

- 2 tables are OK
- The superclass table holds the common attributes
- The subclasses table holds key-value pairs according to the type
- Seams reasonable, because no space is wasted
Publication table

```sql
create table publication (
    pid int not null auto_increment,
    title varchar(200) not null,
    url varchar(200),
    bibtype enum("Article", ...) not null,
    year varchar(4) not null,
    primary key(pid)
);
```
Publication attributes table

create table pubattr (  
    pid int not null,  
    name varchar(30) not null,  
    value varchar(200) not null,  
    primary key(pid, name)  
);
We have person table

create table person (  
  peid int not null auto_increment,  
  name varchar(80),  
  primary key(peid)  
);
Association between publication and persons is a many-to-many

- A publication can have many authors
- An author can work on a number of publications

Thus, we need to map this association using an associative table
create table pubperson ( 
    peid int not null, 
    pid int not null, 
    role enum("Author", "Editor"), 
    primary key(peid, pid) 
);
O/R Mapping - Implementation (1/4)
Interface between process logic and data management

- Programmatic (API) → no SQL in there

Another very important rule

- No SQL in the process logic
- Otherwise mixture of process logic and data management
- SQL only allowed through the implementation in the data management module
Implementation alternatives

In all cases we need JDBC

http://java.sun.com/products/jdbc/

However, we want to abstract JDBC and SQL from process logic

The implementation needs to hide it from process logic

Alternative 1: Implement the abstraction yourself

Using design patterns + JDBC
Alternative 2: Use an existing implementation and declare the mapping only

E.g., Java Data Objects from Sun
http://java.sun.com/products/jdo/

E.g., Enterprise Java Beans
http://java.sun.com/products/ejb/

O/R Mapping framework: Hibernate
http://www.hibernate.org/
Implementing the abstraction - JDBC issues (1/14)

SQL statement problems

JDBC API provides the Statement class

Statement objects usually applied in the following way

```java
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery(
    "select * from publication where title regexp " +
    request.getParameter("title"));
```
Implementing the abstraction - JDBC issues (2/14)

- Usually long statements → typing errors
- Security considerations
  - Suppose that users type “something ; select * from passwords;”
- Performance issues
  - Each time the statement is compiled again!
Implementing the abstraction - JDBC issues (3/14)

Solution: use PreparedStatement class

```java
select_pubs = connection.prepareStatement(
    "select * from publication where title regexp ?";
select_pubs.setString(1, request.getParameter("title"));
```

Implementing the abstraction - JDBC issues (4/14)

- No typing errors
  - No need to compose the query string
  - Types are checked, e.g. `setString()`, `setInt()`
- The security issue resolved
  - Parameter is used as a value of the query variable
- Performance improved
  - `PreparedStatement` is pre-compiled only once
Authentication and connection pooling

DB authentication usually implemented in the following way

```java
Connection connection = DriverManager.getConnection(
    "jdbc:mysql://localhost/publicationdb",
    "username", "password");
```

Problems

- Security, portability
Implementing the abstraction - JDBC issues (6/14)

A simple solution

- Declare username and password as init-param in web.xml

Another problem of the previous approach

- Performance since each request opens a new connection

A better solution

- Resolve authentication issue together with connection pooling
Implementing the abstraction - JDBC issues (7/14)

- Connection pools

- Broker class encapsulates access to the database connections with `get-Connection()` method

- Behind the scene broker manages a buffer of connections
  - `getConnection()` returns the first available connection
  - If no free connection enlarge the buffer
  - When a client finishes it frees the connection (broker notified)

- Optimizing the buffer size!
Usually JDBC drivers provide connection pooling

Apache Tomcat provides connection pooling

- Database Connection Pool (DBCP)
- Part of Jakarta Commons project

http://jakarta.apache.org/commons
Implementing the abstraction - JDBC issues (9/14)

- DBCP uses Java Naming Directory Interface (JNDI)
- JNDI Data Source
- Define JNDI resource reference in web.xml
- Map JNDI resource onto a real resource in server.xml
- Lookup JNDI data source in the code
Define JNDI resource reference in web.xml

```xml
<resource-ref>
  <res-ref-name>jdbc/publicationdb</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```
Map JNDI resource onto a real resource in server.xml

```xml
<Context path="/struts-publicationdb"
    docBase="struts-publicationdb"
    debug="0" reloadable="true" >
<ResourceParams name="jdbc/publicationdb">
    <parameter>
        <name>username</name>
        <value>publicationdb</value>
    </parameter>
    <parameter>
        <name>password</name>
        <value>mmis2004</value>
    </parameter>
</ResourceParams>
....
```
Map JNDI resource onto a real resource (continued)

```xml
<parameter>
    <name>driverClassName</name>
    <value>org.gjt.mm.mysql.Driver</value>
</parameter>

<parameter>
    <name>url</name>
    <value>jdbc:mysql://localhost/publicationdb</value>
</parameter>

</ResourceParams>
</Context>
```
Lookup JNDI data source in the code

Context init = new InitialContext();
Context ctx = (Context) init.lookup("java:comp/env");
DataSource ds = (DataSource) ctx.lookup("jdbc/publicationdb");
connection_ = ds.getConnection();
Be careful!

- Need to notify the broker when finished
- Do so by closing connection and all other DB resources

```java
try {
    ....
} finally {
    close(insert_person, result);
    closeConnection();
}
```
Design Patterns - Data Access Object (DAO)(1/5)

- Encapsulates all database access into a single class
  - CRUD interface (create, read, update, delete)
- Works with transfer objects
  - TOs reflect the model
Example of DAO CRUD interface

PersonDAO
....
public void storePerson(Person person);
public Iterator readAllPersons();
public Person readPersonWithId(int id);
....

Source code
http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PersonDAO.java
Similar classes for Publication

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publication/src/edu/iicm/publication/db/PublicationDAO.java

Code to access the DAOs and present the results (Search interface)

PublicationDAO dao = DAOFactory.createPublicationDAO();
...
StringBuffer buffer = new StringBuffer();
Iterator pubs = dao.readAllPubs(type, year, title);
while (pubs.hasNext()) {
    Publication pub = (Publication) pubs.next();
    buffer.append(pub.getStringRep(writer));
}
To decouple the storage from DAO class work with interfaces

Work with Abstract Factory Pattern to obtain proper DAO instances

Allows you to move to another storage, e.g., XML native database

Factory

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/db/DAOFactory.java
Two levels of implementation

An abstract class that implements the DAO interface

It declares a number of abstract methods to access the storage

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PublicationDAOImpl.java

A particular implementation implements the abstract methods

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PublicationJDBCDAOImpl.java
Using a mapping framework - Hibernate

Hibernate is an open source Java O/R mapping framework

http://www.hibernate.org

Hibernate allows you to simple declare the O/R mapping

- The implementation is already included in the framework

Currently widely used (the most popular O/R tool)

- It supports all major RDBMS
You develop the Model as Java classes, objects, etc.

For this OO model you declare a mapping onto relations

- Recollect the suggestion that the OO model should drive the mapping

The objects and classes from the Model need only to follow common Java idiom

- Java bean idiom
Java beans specification


Very simple specification

- For each class attribute you need getAttrName() and setAttrName()
- For each boolean class attribute you need isAttrName()
- You need an empty public constructor

Basically, it is a naming convention
How the Java Beans specification can be useful?

E.g., Struts form beans

Login form bean

...
Login form bean

```java
public final class LoginForm extends ActionForm {
    private String username_ = null;
    private String password_ = null;

    // Getters and setters
    public String getPassword() {
        return password_;}
    public void setPassword(String password) {
        password_ = password;
    }

    // Other methods...
}
```
Struts reads the parameters submitted with the HTML form

The names of parameters are used to create method names

  E.g., setPassword(), setUsername()

Struts invokes these methods on the instance of the form bean class

How Struts can invoke these methods in run-time?

  The invocations are not hard-coded!
Java Reflection API

http://java.sun.com/docs/books/tutorial/reflect/

This API allows you to investigate run-time properties of Java classes and objects

- e.g., what is the name of the superclass of this class?
- e.g., give me the list of public methods of this class?
- e.g., give me an instance of a method of an object!!

Method is a class that can have instances
Additionally, it allows you to instantiate objects, invoke methods, etc.

e.g., give me an instance of this Method, call the invoke() method on that instance

It is equivalent to a hard-coded method invocation

In this way Struts can call methods on the instance of the form bean class
Example of Java reflection

Publication attributes table

```sql
create table pubattr (  
   pid int not null,  
   name varchar(30) not null,  
   value varchar(200) not null,  
   primary key(pid, name)  
);
```
Depending on the publication type we have different additional attributes.

**Article type**

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/Article.java

In the pubattr table we have key-value pairs:

- e.g., (journal, “JUCS”)

The key of this pair should be used to obtain the name of the method:

- e.g., getJournal(), setJournal()
Using Java reflection we can then invoke the proper method and set the attribute

```java
final protected void populatePubAttributes(Publication pub, ... ... HashMap attributes) {
...

    // use Reflection API to invoke setters
    Class[] class_args = new Class[1];
    class_args[0] = String.class;

    Iterator attrs = attributes.keySet().iterator();
    while (attrs.hasNext()) {
        String key = (String) attrs.next();
        String value = (String) attributes.get(key);
        ...
```
// prepare arguments for reflection
key = key.substring(0, 1).toUpperCase() +
    key.substring(1, key.length());
Object[] args = new Object[1];
args[0] = value;

try {
    Method setter = pub.getClass().getMethod("set" +
        key, class_args);
    setter.invoke(pub, args);
} catch (SecurityException exc) {
...
}
The same approach is used by Hibernate

You need to specify what attributes of your Java objects you want to persist

Hibernate uses these names to invoke the proper getters and setters

e.g., for Publication class you specify “title” attribute

Hibernate will generate getTitle() and setTitle() and invoke them with reflection API

Additionally, you define how to map those attributes onto relations

Hibernate hides the JDBC and SQL issues from you
Publication example with Hibernate (1/13)

Model

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publication/src/edu/iicm/publication

All concerned classes are Java beans

We have inheritance (Publication and subclasses)

We have a many-to-many relation (Publication and Person)

For each class that we want to make persistent we need a declaration file
Publication example with Hibernate (2/13)

Person.hbm.xml

<hibernate-mapping>
  <class name="edu.iicm.publication.Person"
       table="person">
    <id name="id" column="peid">
      <generator class="increment"/>
    </id>
    <property name="name" column="name"/>
  </class>
</hibernate-mapping>

Source

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/src/edu/iicm/publication/Person.hbm.xml
Publication.hbm.xml

<hibernate-mapping>
  <class name="edu.iicm.publication.Publication"
        table="publication">
    <id name="id" column="pid">
      <generator class="increment"/>
    </id>
  </class>
  <property name="title" column="title"/>
  <property name="year" column="year"/>
  <property name="url" column="url"/>

  ...
</hibernate-mapping>
...<set name="persons" table="pubperson">
  <key column="pid"/>
  <many-to-many column="peid"
      class="edu.iicm.publication.Person"/>
</set>
...

<discriminator column="bibtype" type="string"/>
<subclass name="edu.iicm.publication.Article"
  discriminator-value="Article" />

....

<subclass name="edu.iicm.publication.Unpublished"
  discriminator-value="Unpublished" />
</class>

...
The mapping files should accompany class files

Hibernate resolves automatically associations, inheritance, etc.

e.g., proper subclasses will be created depending on the value of the “bibtype” field

Persons will be loaded whenever a Publication is loaded

You can also configure at what time the Persons should be loaded

e.g., eager load → immediately

e.g., lazy load → when a person is requested
HibernateDAO

Actually, we do not need this since Hibernate already abstracts JDBC and SQL

With DAO we can abstract Hibernate from the process logic ;)

Additionally, Hibernate configuration is accomplished by DAO

Belongs to the Data Management Module
PersonHIBERNATEDAO

... static {
    try {
        sessionFactory_ = new Configuration().addClass(Person.class).
            buildSessionFactory();
    } catch (Exception exc) {
        log_.error("Initial SessionFactory creation failed.", exc);
        throw new ExceptionInInitializerError(exc);
    }
}

...
public void storePerson(Person person) {
    Session session = session_factory_.openSession();

    Transaction transaction = session.beginTransaction();
    session.save(person);
    transaction.commit();
    session.close();
}
...
public Iterator readAllPersons() {
    Session session = session_factory_.openSession();

    String query_string = "from edu.iicm.publication.Person";
    Query query = session.createQuery(query_string);
    return query.iterate();
}
public Person readPersonWithId(int id) {
    Session session = session_factory_.openSession();
    String query_string = "select person from person 
    + "in class edu.iicm.publication.Person 
    + "where person.id=:id";
    Query query = session.createQuery(query_string);
    query.setInteger("id", id);
    return (Person) query.uniqueResult();
}

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PersonHIBERNATEDAOImpl.java
Publication example with Hibernate (12/13)

Similar for Publication DAO

Source code PublicationHIBERNATEDAO

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publication/
src/edu/iicm/publication/db/PublicationHIBERNATEDAOImpl.java

Stand-alone demonstration

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publication/
src/edu/iicm/publication/test/PublicationDBTest.java
Hibernate uses HQL: The Hibernate Query Language

HQL Manual


Very similar to SQL

But is fully OO

You search in classes, and object collections, not in relations
Further Readings (1/2)

Mapping objects to relational databases

The Object-Relational Impedance Mismatch
http://www.agiledata.org/essays/impedanceMismatch.html

Mapping Objects to Relational Databases: O/R Mapping In Detail
http://www.agiledata.org/essays/mappingObjects.html

Agile Database Techniques
http://www.ambysoft.com/agileDatabaseTechniques.html

Database articles on
http://onjava.com
http://onjava.com/topics/java/jdbc_sqlj
Further Readings (2/2)

Core J2EE Patterns - Data Access Object
http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

Hibernate Your Data
http://www.onjava.com/pub/a/onjava/2004/01/14/hibernate.html

Hibernate Reference Manual

Get started with Hibernate

Struts Recipes: Hibernate and Struts
Web Services

Denis Helic
One of the basic principles of Software Engineering - Separation of Concerns

We separate different aspects of the system into different modules

Each module supports a single part of the functionality

The modules interact with each other to provide the needed functionality

How do we connect modules, i.e., how they interact with each other?
Service-Oriented Architecture (2/15)

The communication through interfaces

We can observe interfaces from different view points

From the implementation point of view

- Programmatic (procedural) interface (API)
- Declarative interfaces, e.g., XML, SQL, etc.
- Hybrid, i.e., API + Declarative
Service-Oriented Architecture (4/15)

Still need to separate this (later in the course)

User Interface

Data Presentation Layer

Data Aggregation Layer

Data Access Layer

Data Management

Process Logic

File IO

SQL

XPath

XQuery

Metadata QL

File System

RDBMS

XML DB

Metadata DB
Programmatic interfaces

- Process Logic and Data Management Module with DAO

Declarative interfaces

- Data Integration and Data Presentation Layer within XML Publishing

Hybrid interfaces

- Process Logic and Data Management Layer with Hibernate
Another way of observing interfaces is the runtime environment

- All modules in one OS process (modules = objects)
- Modules distributed in a number of processes on the same machine (modules = components)
- Modules distributed over the network
Finally, we can observe interface concerning the way of coupling of modules

- Tightly coupled
- Loosely coupled
- Decoupled (if not connected to each other)
If a modification to one module leads to a modification of another module than tightly coupled

- Otherwise loosely coupled

If operation of a module depends on operation of another module than tightly coupled

- Otherwise loosely coupled

If decoupled the modules are totally independent

- They can still communicate through a middleware module
Service-Oriented Architecture (9/15)
Usually, APIs are closer to tight coupling

- e.g., a modification to DAO interface requires a modification in the process logic
- Without the modification the system can not function anymore

Typically, declarative interfaces are closer to loose coupling

- e.g., adding a new XML element to an XML document processed by an XSLT
- The system can function but the new functionality requires an extended XSLT
Service-Oriented Architecture (SOA) is an architectural style

The goal of this style is to achieve loose coupling among services interacting via declarative interfaces over the network

A service is a piece of functionality

- e.g., DAO service provides CRUD interface to the underlying database
SOA imposes two architectural constraints to achieve loose coupling:

- Simple, ubiquitous and generic declarative interfaces
  - e.g., only specification of communication between services
  - e.g., request, response, how to send information (messages), etc.
  - e.g., no information on what the service needs to do or how to do it
Service-Oriented Architecture (13/15)

- Descriptive (declarative) messages constrained by a schema
  - The semantics is in messages

- The schema prescribes the structure and the vocabulary of messages
  - e.g., header, body of the message

- No (or minimal) system behaviour is included in the messages

- The schema is extensible
  - No extension breaks the previous versions of services
Example, DAO interface as a service

The DAO service returns OO descriptions of data contained in a database. Such descriptions can be used by the process logic to instantiate objects.

The DAO service receive messages containing only identifiers of the data.
Request - response cycle of the DAO service

Request message containing “Person” and the name of the person

The DAO service access the database and retrieves the data

Internally, it creates a Person object

Response message contains a description of this object
Web Services are a specific application of SOA

Obviously, the network infrastructure is the Web

Other important features of SOA are also mapped onto the Web by Web Services

e.g., using of HTTP, XML, etc.

Web Services standardization by Web Consortium

http://www.w3.org/2002/ws/
How Web Services comply with the SOA constraints

Generic communication interfaces

- Interfaces based on Internet and Web protocols such as HTTP, SMTP, FTP, etc.

  - e.g., HTTP request (GET, POST)

  - e.g., HTTP response

  - e.g., HTTP messages with HTTP header, HTTP body, etc.

  - e.g., no specification what to do or how to do something!
Messages are encoded as XML

- e.g., no predefined semantics (each application defines its own semantics)
- e.g., well-formedness of XML
- e.g., XML schemas for XML validity
- e.g., schemas are extensible
Where is the message structure
  i.e., message header, message body?
Defined by a particular style of Web Services
SOAP Web services
  SOAP = Simple Object Access Protocol
SOAP is an XML vocabulary that allows you to encode messages sent between Web services.

- It defines the basic message structure, i.e., message header, message body.
- You can add any other XML into a message body.

SOAP is developed by the Web Consortium: [http://www.w3.org/2000/xp/Group/](http://www.w3.org/2000/xp/Group/)

A Web service is a piece of functionality.

- Typically, SOAP messages used for Remote Procedure Call.
<SOAP-ENV: Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

    <SOAP-ENV:Header>
        <t:Transaction xmlns:t="some-URI"
            SOAP-ENV:mustUnderstand="1">
        </t:Transaction>
    </SOAP-ENV:Header>

    <SOAP-ENV:Body>
        <m:GetBookPrice xmlns:m="some-URI">
            <title>My Life and Times</title>
            <author>Felix Harrison</author>
        </m: GetBookPrice>
    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
In addition to SOAP, Web services introduce WSDL

- WSDL = Web Services Description Language

WSDL is also developed by the Web Consortium
http://www.w3.org/2002/ws/desc/

With WSDL you describe how the message body looks like
- A kind of Web services schema (similar to XML - XML schemas)
- Which methods you can invoke and what are the arguments
- Method declarations
A Web service is a piece of functionality

It is offered for public by a service provider

We have a server where we host the service

I.e., the service has its URL

e.g. http://localhost:8080/axis/services/EchoService
Simple example of a service

The service prints out the string it receives (echo)

```java
public class EchoService {
    public String echo(String msg) {
        return msg;
    }
}
```

Source code

http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/ws/EchoService.java

e.g., HTTP GET

Invoking a Web service using encoded URL over HTTP GET

Encoded URL will be transformed into a SOAP message

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <echo>
      <msg>Hello World!</msg>
    </echo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Very simple encoding

- echo element is the method
- msg subelement is the argument

The Web service guesses the proper datatype conversions

- In this case very simple because the argument and the return value are simple strings
Frameworks provide abstractions of SOAP and WSDL!

- Basically, they allow you to do RPC transparently
- You do not even need to know about SOAP or WSDL

.NET from Microsoft
http://www.microsoft.com/net/

WebSphere from IBM

J2EE from Sun
http://java.sun.com/developer/technicalArticles/J2EE/j2ee_ws/
Apache Axis

http://ws.apache.org/axis/

Open source Java-based Web services framework

Started as a SOAP framework

Evolved to support WSDL, etc.

Download, the latest version is 1.2 RC2

http://ws.apache.org/axis/releases.html

Installation guide

http://ws.apache.org/axis/java/install.html
WSDL is a schema language for Web services

WSDL document describes

- Datatypes that a Web service uses (e.g., PurchaseOrder schema)
- Operations that a Web service provides
- The location of a Web service, i.e., its URL
WSDL of the EchoService

...<wsdl:binding name="EchoServiceSoapBinding" type="impl:EchoService">
  <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="echo">
    <wsdlsoap:operation soapAction=""/>
    <wsdl:input name="echoRequest">
      ...
    </wsdl:input>
    <wsdl:output name="echoResponse">
      ...
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
...
<wsdl:message name="echoResponse">
  <wsdl:part name="echoReturn" type="soapenc:string"/>
</wsdl:message>

<wsdl:message name="echoRequest">
  <wsdl:part name="msg" type="soapenc:string"/>
</wsdl:message>

...
WSDL is a structured description of a Web service

It is written in a machine-understandable way

- e.g., a client accessing the WSDL of a service can invoke the service as described

Automatization of the process of Web services invocation
Automatization of creation of clients

- Automatically generate clients from WSDL descriptions
- Abstract the SOAP/WSDL completely and hide it behind an API

Automatization of creating services

- Use your existing functionality
- Create automatically WSDL
Java2WSDL tool creates automatically WSDL from Java interfaces

- It creates descriptions of all public methods
- It creates proper datatypes
  - It needs to be able to access these Java classes
  - The classes should be JavaBeans
WSDL2Java tool creates Java classes from WSDL descriptions

Client side

A Java interface which corresponds to the operations defined in WSDL

For echo service this would be Echo interface with a single public method

A so-called stub, which is the implementation of the generated interface

It uses Axis API (e.g., Service, Call) and SOAP to communicate with the Web service

A ServiceLocator class which retrieves instances of the generated interface

Basically it is an Abstract Factory
Server side

- An empty implementation of the generated interface
- You should modify this empty implementation
- Forward the calls to the proper methods of your implementation
PersonDAO interface from the Publication Database

Basically, we want to offer the access to the database through a Web service

- The usual way of providing Web services
- I.e., publishing already existing functionality as Web services
PersonDAO interface

....

public void storePerson(Person person);
public Iterator readAllPersons();
public Person readPersonWithId(int id);
....

Source code

http://coronet.iicm.edu/mmis2/examples/mvc/struts-publicationdb/
src/edu/iicm/publication/db/PersonDAO.java
Using Java2WSDL

org.apache.axis.wsdl.Java2WSDL
-o persondao.wsdl
-l "http://coronet2.iicm.edu/axis/services/persondao"
-n urn:coronet.iicm.edu/persondao
-p "edu.iicm.publication.db" coronet.iicm.edu/persondao
edu.iicm.publication.db.PersonDAO

Source code

http://coronet.iicm.edu/mmis2/examples/ws/persondao.wsdl
SOAP and WSDL in Axis - An Example (4/12)

Using WSDL2Java

org.apache.axis.wsdl.WSDL2Java
-o src
-d Session
-s
-p edu.iicm.publication.db.ws
persondao.wsdl
Client side

PersonDAO interface for the client

```java
package edu.iicm.publication.db.ws;

public interface PersonDAO extends java.rmi.Remote {
    public void storePerson(edu.iicm.publication.db.ws.Person in0)
            throws java.rmi.RemoteException;
    public java.lang.Object readAllPersons()
            throws java.rmi.RemoteException;
    public edu.iicm.publication.db.ws.Person readPersonWithId(int in0)
            throws java.rmi.RemoteException;
}
```

Source code

http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/
publication/db/ws/PersonDAO.java
Person class created from WSDL

```java
package edu.iicm.publication.db.ws;
public class Person implements java.io.Serializable {
    private int id;
    ...
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    ...
}
```

Source code

[http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/Person.java](http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/Person.java)
SOAP and WSDL in Axis - An Example(7/12)
PersonDAOStub
Implemets PersonDAO interface and hides the SOAP from the client

public java.lang.Object readAllPersons()
throws java.rmi.RemoteException {
...
org.apache.axis.client.Call _call = createCall();
_call.setOperation(_operations[1]);
_call.setUseSOAPAction(true);
_call.setSOAPActionURI("");
_call.setSOAPVersion(org.apache.axis.soap.SOAPConstants.SOAP11_
...

(475/618)


_call.setOperationName(
   new javax.xml.namespace.QName(
       "urn:coronet.iicm.edu/persondao",
       "readAllPersons"));
setRequestHeaders(_call);
setAttachments(_call);
java.lang.Object _resp =
   _call.invoke(new java.lang.Object[] {});
...

Source code

http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/
publication/db/ws/PersondaoSoapBindingStub.java
Service side

PersonDAOImplementation

... public edu.iicm.publication.db.ws.Person readPersonWithId(int in0) throws java.rmi.RemoteException {
edu.iicm.publication.db.PersonDAO dao = DAOFactory.createPersonDAO();
...
...
edu.iicm.publication.Person person =
dao.readPersonWithId(in0);
edu.iicm.publication.db.ws.Person out0 =
    new edu.iicm.publication.db.ws.Person();
out0.setId(person.getId());
out0.setName(person.getName());
out0.setRole(person.getRole());
return out0;
}
...

Source code

http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/
publication/db/ws/PersondaoSoapBindingImpl.java
Client code

... 

PersonDAOServiceLocator locator = new PersonDAOServiceLocator();

try {
    PersonDAO dao = locator.getPersonDAO();
    Person person = new Person();
    person.setName(name);
    dao.storePerson(person);
    log_.debug("Person stored");
...
**SOAP and WSDL in Axis - An Example**

Source code

http://coronet.iicm.edu/mmis2/examples/ws/edu/iicm/publication/db/ws/client/PersonDAOClient.java

Example with Eclipse

SOAP Monitor

http://localhost:8080/axis/SOAPMonitor

HTTP GET

http://localhost:8080/axis/services/persondao?method=readAllPersons
The basic problem: it is exclusively used for Remote Procedure Calls

RPC implies an API

Rcollect that APIs tend to enforce tight coupling of modules and systems

It is somewhat confusing

We use declarative XML to describe APIs

Regardless if you use Java interfaces or XML it is still an API
Problems of SOAP/WSDL Protocol Stack (2/2)

- APIs rely on a close world assumption
- You have modules and systems that are tightly coupled
  - You need to know all of your users, their systems, tools
  - You need to enforce a particular API
  - Everyone needs to know about the changes in the API
- Scalability problem!
But what do we have on the Web

Obviously it is not a closed world

Impossible to know all of your users

Impossible to enforce API or changes in API

The only possibility is to have a very simple basic and extensible framework
But we already have that

A standardized, extensible application protocol: HTTP

A standardized, extensible message format: HTML

A single unifying namespace of URLs

  Allows to create links that make the Web actually

  URLs keep the Web together as a single huge application
Now, let us extend a notion of URLs (URIs)

URLs identify resources: IICM, Homepage of IICM, ...

Representations of resources are delivered over the Web by means of HTTP

Most notably, HTML representations

But other representations are possible, such as GIFs, JPEGs, etc.
But we are interested in declarative representations

E.g., XML-based representations

That particular extension is the basis for resource-oriented Web Services

HTTP, URIs, XML

It is extremely loosely coupled (as the Web itself)

We do not care what is on the other side (a file, a method, a module, etc.)

As long as we can use simple HTTP GET + URI and get XML out of it
Note here the difference between RPC and this new approach.

To get an XML representation out of a RPC service you need to know:
- Its URL
- Its WSDL

Then you need to build a SOAP message accordingly.
Some advantages of Resource-oriented Web Services

Extensibility

- You organize a web service around URLs
- That service can automatically integrate with other web services through links
- Similar to how HTML pages are linked together

Performance

- Anything that is retrieved with HTTP GET and URL can be cached
- Caching is already built in into HTTP

These two advantages were the reasons for the success of the Web
Representational State Transfer: REST(1/4)

A particular architectural style for Web Services

Based on the premises from above

- HTTP, URLs, XML

Use HTTP to its full extent

- GET, POST, PUT, DELETE

Manage and manipulate resources on the Web

- Concentrate on data (XML) rather than on software (RPC)
How to manipulate resources

CRUD interface

Create = PUT or POST

Read = GET

Update = POST or PUT

Delete = DELETE

We have everything!

Just need to create XML vocabularies to represent resources
Comparison with SOAP/WSDL

The majority of SOAP/WSDL based services

- Simply provides CRUD interface
- Why then take SOAP/WSDL and add one level of complexity

Now we can think of a Web server as a huge information repository, a database

The Web is a huge distributed hypermedia information repository that links all Web servers into a single system
Everything is already there

- Everything has been there for almost ten years

No need to wait for standards, implementations, etc.

Just use HTTP, URLs, XML to:

- Publish information on the Web
- Retrieve information from the Web
- Link information on the Web
To create a REST Web service the following steps should be accomplished:

1. Define URLs
2. Define XML vocabulary
3. Define which methods are available at each URL
4. Define HTTP status codes
How to create a REST Web service (2/6)

Defining URLs

Recollect the clean URLs that we have discussed before

Course with ID 1:  http://localhost:3000/courses/1

Course with ID 2:  http://localhost:3000/courses/2

All courses:  http://localhost:3000/courses/

Note that URLs are regular, i.e., we can automatically construct them
Defining XML formats

Single course

```xml
<?xml version="1.0" encoding="UTF-8"?>
<course>
  <title>MMIS2</title>
  <url>http://coronet.iicm.edu/lectures/mmis2</url>
  <id type="integer">1</id>
  <description>Advanced course on multimedia information systems</description>
</course>
```
All courses (collection)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<courses>
  <course>
    <title>MMIS2</title>
    <url>http://coronet.iicm.edu/lectures/mmis2</url>
    <id type="integer">1</id>
    <description>Advanced course on multimedia information systems</description>
  </course>
  <course>...</course>
</courses>
```
How to create a REST Web service (5/6)

Defining methods

GET and POST

GET retrieves the list, POST adds another course to the list

GET, PUT, DELETE

PUT updates the course

Additionally, send OPTIONS at each URL to see what is available
Defining HTTP status codes

GET: HTTP/1.1 200 OK

POST: HTTP/1.1 201 Created

PUT, DELETE: HTTP/1.1 204 No Content

Error code: HTTP/1.1 500 Internal Server Error
We need a Web server that can dispatch URLs and HTTP methods to appropriate code.

Within that code we manipulate resources using their XML representations.

With Cocoon we can easily dispatch URLs (recollect sitemap).

With Rails we can also dispatch URLs easy.

Recollect that Rails dispatch to a class and a method within that class.

We need to dispatch to a class and according to the HTTP method.
There is a Rails plugin that does exactly that

RESTful Rails controller

http://rubyforge.org/projects/restful-rails/

class CourseController < ApplicationController
  includeRestController::Base
  ...
end
The plugin allows you to introduce resources and within resources handle HTTP methods.

```ruby
resource :collection do |r|
  conditions << @courses = Course.find_all

  r.post do
    ...
  end
end
```

GET method is automatically handled and connected with appropriate XML-based view.
collection.rxml in views directory

xml.instruct!
xml.courses do
  xml.title 'Courses'
  @courses.each do |course|
    xml.course do
      xml.id   course.id
      xml.title course.title
      xml.url  course.url
      xml.description course.description
      ...
    end
  end
end
Handling POST to the collection (adding a course)

```ruby
r.post do
  @course = Course.new params[:course]
  if @course.save
    render_post_success :action => 'by_id', :id => @course
  end
end
```
Handling of single courses (introduce a new rule by_id)

resource :by_id do |r|
  conditions << @course = Course.find(params[:id])

  r.put do
    @course.attributes = params[:course]
    if @course.save
      render_put_success
    end
  end
end
Deleting a resource...

r.delete do
  if @course.destroy
    render_delete_success :id => nil
  end
end
...

GET XML view

xml.instruct!

@course.to_xml
Demo: [http://coronet.iicm.edu/mmis2/examples/ws/rest/rest_courses.zip](http://coronet.iicm.edu/mmis2/examples/ws/rest/rest_courses.zip)

curl http://localhost:3000/courses

curl http://localhost:3000/courses/1

curl -i -X POST -d '<course><title>Web Information Systems</title></course>' http://localhost:3000/courses

curl -i -X PUT -H 'Content-Type: application/xml' -d '<course><title>Web Information Systems</title><description>Very advanced course</description></course>' http://localhost:3000/courses/4

curl -i -X DELETE http://localhost:3000/courses/4
Ajax = Asynchronous JavaScript and XML

- Nothing new except the famous XMLHttpRequest object
- Supported in all browsers
- Allows you to send any HTTP request to a Web server from JavaScript
- When the response arrives you can do something with the received data
The intent is to make web pages feel more responsive

By exchanging small amounts of data with the server behind the scenes

The entire web page does not have to be reloaded each time the user makes a change

Meant to increase the web page’s interactivity, speed, and usability
But it is still JavaScript with all of its advantages/disadvantages

Usability issues, compatibility issues, performance

Interactivity is increased

Many successful applications, gmail, digg, flickr, etc.
A possibility to build Rich Web Clients

Javascript widget libraries

It can be very powerful when combined with REST services

I hope to see some demonstrations during your presentations ;-)
Further Readings (1/3)

What is Service-Oriented Architecture?

A Web Services Primer

Top Ten FAQs for Web Services
http://webservices.xml.com/pub/a/ws/2002/02/12/webservicefaqs.html?page=1

Service-Oriented Architecture and Web Services: Concepts, Technologies, and Tools

Articles on http://webservices.xml.com/ from O’Reilly
Further Readings (2/3)

- SOAP Primer from W3C
  http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

- SOAP Tutorial
  http://www.w3schools.com/soap/default.asp

- Java and XML: SOAP
  http://www.onjava.com/pub/a/onjava/excerpt/java_xml_2_ch2/

- WSDL Tutorial
  http://www.w3schools.com/wSDL/default.asp

- Apache Axis User Guide
  http://ws.apache.org/axis/java/user-guide.html
Further Readings (3/3)

REST definition (PhD Thesis of Roy Fielding)

The Restful Web: Column on xml.com
http://xml.com/pub/at/34

Putting REST on Rails
http://www.xml.com/pub/a/2006/04/19/rest-on-rails.html

REST vs SOAP: Second Generation Web Services
http://webservices.xml.com/pub/a/ws/2002/02/06/rest.html
Semantic Web and RDF
Introduction
Denis Helic
Web today: more than 10 billions of Web pages

How to discover information on the Web (very hard!)?

- Browsing
- Searching

Very often results are disappointing!
How do we find things in real world?

Example 1: The Library

You want to find a book on Relational Databases

Use the computer in the library to list books by author, title, subject, etc.

Identify the book you want and look in the list where the book is
Example 2: The Video Store

You want to find the latest Star Wars movie

Use the computer in the store to list books by director, title, actors, etc.

Identify the video you want and look in the list where the video is

Both examples based on *metadata*
Metadata is data about data

Example 1: Data about books

- Author: Nick Scerbakov
- Subject: Databases
- Title: From Databases to Hypermedia

...
Example 2: Data about videos

- Director: George Lucas
- Title: Star Wars - Revenge of the Sith
- Actor: Ewan McGregor

...
Metadata comes in key-value pairs

Key: Author

Value: Nick Scerbakov

For different applications we use different sets of metadata

Library: author, title, subject, ...

Video store: director, title, actors, ...
We can use metadata for other purposes as well!

- Cataloging
- Printing
- Sorting, etc.
Use metadata to describe Web resources

- Web sites, Web pages, etc.

We can apply metadata on the Web in the same way as in the real world

- searching, cataloging, printing, etc..

We can apply metadata in a more Web-specific way

- provide a summary of a Web site, describe intellectual property rights of a Web page, etc.
Metadata is very useful for multimedia resources
  - images, videos, music, etc.

We can try to solve problems with searching for multimedia resources

Example: Describing digital images with metadata
  - what objects do I have on an image (e.g. dog, cat, etc.)
Current situation with metadata on the Web

Not that much metadata on the Web

Especially, not that much in HTML pages

<meta> element in HTML

used to raise search engine rankings!

search engines do not query <meta> elements directly

Some systems apply metadata for more than 10 years now! (e.g. Hyperwave)
What are the challenges for metadata on the Web?

Web comprises many different applications

- Many libraries on the Web
- Many video stores on the Web
- ...

Applications use different sets of metadata, but the principles are the same (i.e. metadata comes always in key-value pairs)
What do we need to use metadata effectively on the Web?

Technology:

- That supports the basic metadata principle
- That allows to use different sets of metadata
- That is compatible with current Web standards
What do we need to use metadata effectively on the Web (continued)?

Standard (standardized technology):

- We want to exchange metadata with other people
- We want software to process it automatically

We need to agree on a standard for metadata!
Resource Description Framework (RDF) is such standardized technology

Developed by Web Consortium (Recommendation)

http://www.w3.org/RDF

Part of larger initiative called Semantic Web

http://www.w3.org/2001/sw
New initiative by Web Consortium

Definition (from W3C): The Semantic Web is the representation of data on the World Wide Web

Description of Web resources in the form of metadata
Example: Web resource described with some metadata

- Web resource: http://coronet.iicm.edu/lectures/mmis2
- Creator: Denis Helic
- Administrator: Denis Helic
- Title: Multimedia Information Systems
- ...
Design ideas behind Semantic Web

Based on the current Web technologies, such as URL (URI) and XML

Extension of the current Web

Make it very simple to create metadata! (recollect the reason for the success of the Web)

Make it possible to introduce different metadata sets

Anyone, anywhere, anytime can provide metadata for a resource by addressing it with its URI (similar to linking in HTML)
Goals of Semantic Web

- Web of descriptions and resources - Semantic Web
- Enable effective information discovery (e.g. search)
- Enable automation (e.g. software agents)
- Enable effective integration (e.g. ftp and http)
Resource Description Framework - RDF

Resource Description Framework is a recent development of Web Consortium

http://www.w3.org/RDF

Purpose of RDF is to provide a standard for exchanging metadata on the Web

The basic RDF specification consists of:

RDF Data Model: http://www.w3.org/TR/rdf-concepts - status Recommendation

Design goals for RDF data model are:

- To match the basic metadata principle (i.e. to describe Web resources by applying simple key-value pairs)
- A simple data model
- Scalable (it has to scale over the whole Web!)
- Anyone can provide metadata for any resource
- Independent (e.g. anyone can invent metadata sets for their purposes)
- Interchangeable (e.g. easy to format it as XML)
RDF Data Model is based on the following rules:

- A **Resource** is anything that can have a URI (e.g. all Web pages, all Web images, all files accessible through ftp, etc.)
  
  http://coronet.iicm.edu/lectures/mmis2

- A **Property** has a name and describes some relationship (e.g. Creator, Title, Subject, etc.)

- A **Statement** consists of a combination of a Resource, a Property and a value
  
  ”The creator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic”
Statement may be represented as a triple (N3 notation):

<http://coronet.iicm.edu/lectures/mmis2> <creator> <Denis Helic>.

<Resource> <Property> <Value>.

<Subject> <Predicate> <Object>.
Statement may be also represented graphically

http://coronet.iicm.edu/lecture/mmis2 --> http://coronet.iicm.edu/mmis2/examples/rdf#creator

Denis Helic
A particular description of a resource includes a number of statements:

"The creator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic"

"The administrator of http://coronet.iicm.edu/lectures/mmis2 is Denis Helic"

"The title of http://coronet.iicm.edu/lectures/mmis2 is Multimedia Information Systems 2"
<http://coronet.iicm.edu/lectures/mmis2> <creator> <Denis Helic>.
<http://coronet.iicm.edu/lectures/mmis2> <administrator> <Denis Helic>.
Graphically

http://coronet.iicm.edu/mmis2

http://coronet.iicm.edu/mmis2/examples/rdf#creator

http://coronet.iicm.edu/mmis2/examples/rdf#administrator

http://coronet.iicm.edu/mmis2/examples/rdf#title

Denis Helic

Multimedia Information Systems
RDF graph consists of nodes and arcs

Oval nodes represent resources

Square nodes represent values

Arcs (arrows) represent properties
A value might be a resource

"The homepage of http://coronet.iicm.edu/lectures/mmis2 is http://coronet.iicm.edu"

Advanced concepts in RDF

- A Property is also a Resource
- A Statement is also a Resource

Both can have their own properties (e.g. when it was created?)
How this model meets the design ideas?

- To match the basic metadata principle - each triple is a combination of a URI and a key-value pair
- A simple data model - just triples
- Scalable - easy to handle even large number of triples
- Anyone can provide metadata for any resource - if you know its URI!
- Independent - no prescribed sets of metadata
- Interchangeable - we need to specify XML syntax for it!
RDF/XML Syntax Specification (1/11)

Defines how to encode an RDF graph into a valid XML

We need to represent nodes and arcs as XML elements, attributes, element content and attribute values

A graph is a collection of paths of the form Node, Arc, Node, ...

In RDF/XML these turn into sequences of nested elements which alternate between elements for Nodes and Arcs
Example 1

http://coronet.iicm.edu/lectures/mmis2

http://coronet.iicm.edu/mmis2/examples/rdf#creator

Denis Helic

http://coronet.iicm.edu/mmis2/examples/rdf#administrator

Denis Helic

http://coronet.iicm.edu/mmis2/examples/rdf#title

Multimedia Information Systems
First path: creator

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
</rdf:Description>
```
Second path: administrator

```xml
<rdf:Description rdf:about =
   "http://coronet.iicm.edu/lectures/mmis2">
   <administrator>Denis Helic</administrator>
</rdf:Description>
```

Third path: title

```xml
<rdf:Description rdf:about =
   "http://coronet.iicm.edu/lectures/mmis2">
   <title>Multimedia Information Systems 2</title>
</rdf:Description>
```
Example 1 (complete)

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
  <administrator>Denis Helic</administrator>
  <title>Multimedia Information Systems 2</title>
</rdf:Description>
```

Example 1:

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf
Example 2
Example 2 (continued)

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
  <creator>Denis Helic</creator>
  <administrator>Denis Helic</administrator>
  <title>Multimedia Information Systems 2</title>
  <homepage>
    <rdf:Description rdf:about = "http://courses.iicm.edu">
      <creator>Hermann Mauerer</creator>
      <administrator>Karl Trummer</administrator>
      <title>Courses offered by IICM</title>
    </rdf:Description>
  </homepage>
</rdf:Description>
```
To complete the example we need declaration!

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns = "http://coronet.iicm.edu/mmis/examples/rdf#">
    ...
</rdf:RDF>
```

Example 2:

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_complete.rdf
Striping pattern!

```
<rdf:Description rdf:about = "http://coronete.iicm.edu/lectures/mmis2">
  ...
  <homepage>
    <rdf:Description rdf:about = "http://coronet.iicm.edu">
      <creator>Hermann Mauerer</creator>

  ...
</rdf:Description>
</homepage>
</rdf:Description>
```
Gray (Resource) - white (property) stripes

```
<rdf:Description rdf:about = "http://courses.iicm.edu/mmis">
  <homepage>
    <rdf:Description rdf:about = "http://courses.iicm.edu">
      <creator>Hermann Mauerer</creator>
    </rdf:Description>
  </homepage>
</rdf:Description>

<Resource-A>
  <property-A>
    <Resource-B>
      <property-B>value</property-B>
    </Resource-B>
  </property-A>
</Resource-A>
```
Check that the RDF/XML document validates

W3C Validator:
http://www.w3.org/RDF/Validator

Namespaces included into graph

Needed because everybody can define properties
To represent collections we use rdf:Bag, rdf:Seq or rdf:Alt

```xml
<rdf:Description rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
   <topics>
      <rdf:Bag>
         <topic>Internet</topic>
         <topic>XSLT</topic>
         <topic>RDF</topic>
         <topic>Cocoon</topic>
      </rdf:Bag>
   </topics>
</rdf:Description>
```
RDF Advanced Features (2/8)

- rdf:Bag is unordered collection
- rdf:Seq is sequence of properties
- rdf:Alt is set of alternate properties (e.g. represent title in different languages)
Query language for RDF

```
SELECT ?x
WHERE (<http://somewhere/res1>, <http://somewhere/pred1>, ?x)
```

RDQL, Part of Jena RDF framework:

```
http://jena.sourceforge.net/
```

More on the Jena framework in the next lecture
Anyone can provide metadata for any resource - if you know its URI!

The first description:

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf

An additional description:

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_add.rdf
<rdf:Description
    rdf:about = "http://coronet.iicm.edu/lectures/mmis2">
...
</rdf:Description>

Refer to resource with the URI!
An aggregator tool collects all metadata about a resource

We can query all metadata about the specific resource

Metadata might be distributed all over the Web, i.e., we have a distributed network of metadata

Basics of Semantic Web!
Anybody can define metadata sets

RDF Schema: defines all properties

Example Dublin Core Metadata Schema (Library background)

<table>
<thead>
<tr>
<th>Content</th>
<th>Intellectual Property</th>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Creator</td>
<td>Date</td>
</tr>
<tr>
<td>Subject</td>
<td>Publisher</td>
<td>Type</td>
</tr>
<tr>
<td>Description</td>
<td>Contributor</td>
<td>Format</td>
</tr>
<tr>
<td>Language</td>
<td>Rights</td>
<td>Identifier</td>
</tr>
<tr>
<td>Relation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dublin Core: [http://dublincore.org/](http://dublincore.org/)

In many cases you need just Dublin Core

FOAF Project

Friend of a friend

Creating RDF descriptions of people, contacts, homepages, etc.

To support communication, networking between people on the Web

URL: http://www.foaf-project.org/
Further examples on http://www.rdfdata.org/

Open directory RDF dump
http://rdf.dmoz.org/

US State and Regions
http://www.daml.ri.cmu.edu/ont/USRegionState.daml

AI bibliography
http://www.isi.edu/webscripter/planning.scheduling.daml
Expressing bibliographies in RDF

Allows you to exchange bibliographic information more easily

PublicationDB

http://coronet.iicm.edu/struts-publicationdb

Publications

http://coronet.iicm.edu/denis/pubs/list.rdf
Applications using RDF and other Semantic Web technologies

Easiest way to implement Semantic Web applications

Using a Semantic Web framework

Jena Framework

Open source, Java-based, implemented by Hewlett Packard

Project homepage

http://jena.sourceforge.net/
Jena framework provides

- A programmatic environment for RDF and other Semantic Web technologies
- RDF API, i.e., manipulating RDF graphs using an API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- In-memory and persistent storage (with RDBMS)
- RDQL - a query language for RDF
- API for other Semantic Web technologies
Build a server-side application for manipulating RDF graphs (models)

Basic RDF Server (BRDFS)

- Implemented as a master thesis at IICM
- Implemented using Jena Framework

I.e., Jena framework accessible through Web interface
Publishing RDF graphs on the (Semantic) Web

Requirements for BRDFS

- Running as a Web application, i.e., available through HTTP
- Persistent data
- Can create, delete, update models and statements
- Can merge models, query models, etc.
- Easy to use for humans and software agents
BRDFS implementation

As a Web application using Tomcat

Server is an action framework, i.e., similar to Struts

You can issue commands to the server using the action framework

HTML user interface
Published RDF graphs on the (Semantic) Web (4/10)

BRDFS Run-time

At start time the server is connected to a MySQL database

After that every RDF model operation is persisted in that database

Jena provides DB Connection

Jena abstracts the database layer

We just deal with models
Create a new model

```java
ModelMaker maker = ModelFactory.createModelRDBMaker(jena2_dbconnection_.getJena2DBConnection());
Model m = maker.createModel(modelname);
m.close();
```

Model name comes from the request

Source code

http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/edu/iicm/brdfs/action/NEWMODEL.java
Delete a model

```java
ModelRDB m =
    ModelRDB.open(
        jena2_dbconnection_.getJena2DBConnection(),
        modelname);

m.remove();

m.close();
```

Model name comes from the request

Source code

Merge two models

ModelMaker maker =
    ModelFactory.createModelRDBMaker(
        jena2_dbconnection_.getJena2DBConnection());
Model target = maker.openModel(modelname);
target.begin();
...
    Model merge_model = maker.openModel(tomerge_values[i]);
    target.add(merge_model, true);
    merge_model.close();
target.commit();
target.close();

Source code
http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/
edu/iicm/brdfs/action/MERGEMODELS.java
Add statements
ModelMaker maker =
    ModelFactory.
    createModelRDBMaker(
        jena2_dbconnection_.getJena2DBConnection());
Model model = maker.openModel(modelname);
model.begin();
Resource resource = model.createResource(subject);
Property property = model.createProperty(namespace, propertyname);
resource.addProperty(property, object);
model.commit();
model.close();

Source code

http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/
edu/iicm/brdfs/action/ADDDSTATEMENTS.java
Query a model

```java
ModelRDB model = ModelRDB.open(
    jena2_dbconnection_.getJena2DBConnection(), modelname);
Query query = new Query(querystring);
query.setSource(model);
QueryExecution qe = new QueryEngine(query);
QueryResults results = qe.exec();
response_writer_.writeQueryResult(results);
results.close();
```

Source code

http://coronet.iicm.edu/mmis2/examples/rdf/brdfs/src/edu/iicm/brdfs/action/QUERYMODEL.java
RDQL is a query language for querying RDF models

RDQL Tutorial

http://jena.sourceforge.net/tutorial/RDQL/index.html

Similar to SQL

    But operations on triples not on relations

SELECT ?x
WHERE (?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith")
BRDFS Simple Example (1/3)

BRDFS accessible from

http://coronet.iicm.edu/brdfs

We upload two MMIS2 course RDF files from the last lecture

Intellectual property rights of the MMIS2 homepage

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2.rdf

Information about topics of the course

http://coronet.iicm.edu/mmis2/examples/rdf/mmis2_add.rdf

We merge the models and query the final model

- Shows also how metadata integration is achieved
Get creator of the MMIS2 homepage

SELECT ?creator
WHERE (<http://coronet.iicm.edu/lectures/mmis2>
  <http://coronet.iicm.edu/mmis2/examples/rdf#creator> ?creator)
Get the course topics

SELECT ?topic
WHERE (<http://coronet.iicm.edu/lectures/mmis2>
 <http://coronet.iicm.edu/mmis2/examples/rdf#topic>
 ?topic)
Another master thesis project at IICM
Implement OAI provider for the JUCS journal
JUCS journal
http://www.jucs.org/
You can retrieve BibTeX reference for each of the articles
Open Archives Initiative - OAI

http://www.openarchives.org/

Providing metadata about publications in digital libraries

Metadata format is not specified

Should support at least Dublin Core format

Defines also a number of protocols for accessing and retrieving the data
OAI provider for JUCS built on the top of BRDFS

Metadata managed in RDF

- Different output formats available
- BibTeX format encoded as RDF, i.e., as XML
- Dublin Core format encoded as RDF, i.e., as XML
Automatic adding of references for new issues of the journal

Automatic conversion of BibTeX text format onto BibTeX RDF

Adding of statements to an already existing RDF model

HTML format for people

- Supports browsing

- Automatic conversion of XML formats using XSLT on the client side
Administration interface

http://coronet.iicm.edu/brdfs

Uploading of new BibTeX reference files

Retrieving of the model

Querying of the model
Select IDs and titles of all articles

SELECT ?id, ?title
WHERE (?id <http://coronet.iicm.edu/bibtex#journal> "Journal of Universal Computer Science")
(?id <http://coronet.iicm.edu/bibtex#title> ?title)
Select titles and authors

SELECT ?title, ?author
WHERE (?x <http://coronet.iicm.edu/bibtex#journal> "Journal of Universal Computer Science")
(?x <http://coronet.iicm.edu/bibtex#title> ?title)
(?x <http://coronet.iicm.edu/bibtex#author> ?author)
Use abbreviations for namespaces

SELECT ?title, ?author
WHERE (?x bibtex:journal
   "Journal of Universal Computer Science")
(?x bibtex:title ?title)
(?x bibtex:author ?author)
USING bibtex FOR <http://coronet.iicm.edu/bibtex#>
Select titles and authors for specific issues

SELECT ?title, ?author
WHERE (?x bibtex:journal "Journal of Universal Computer Science")
(?x bibtex:title ?title)
(?x bibtex:author ?author)
(?x bibtex:volume ?volume)
AND ?volume == 1
USING bibtex FOR <http://coronet.iicm.edu/bibtex#>
Add another issue to the database

Shows the integration of metadata

HTML user interface

http://coronet.iicm.edu/brdfs/oai/

With XSLT on the client side
RDF is very simple

It provides a basic set for information modelling

Because of this simplicity it is a perfect assembly language

It is possible to build other information modelling languages on top of it

RDF Schema is one such language

http://www.w3.org/TR/rdf-schema/

RDFS is developed by the Web consortium
RDF Schema is an object-oriented modelling language

- It allows you to define classes
- It allows you to define relationships between classes
  - E.g., subclass relationship
- It allows you to define properties of classes
How does RDFS relate to RDF?

- Classes are resources
  - Each class has a URL, so we can define properties for it
- If two classes are related with subClassOf property then we build class hierarchies
- Simple properties of a class are same as properties that have simple values in RDF
RDFS Example

At the top of the class hierarchy we have WaterSource

Two subclasses of WaterSource are Stream and BodyOfWater

- Stream has subclasses such as River, Creek, etc.
- BodyOfWater has subclasses such as Lake, Sea, etc.

There is a relationship between Stream and BodyOfWater

- The relationship is called emptiesInto
Definition of classes and relationships in RDFS

```xml
<rdfs:Class rdf:ID="WaterSource" />

<rdfs:Class rdf:ID="Stream">
  <rdfs:subClassOf rdf:resource="#WaterSource"/>
</rdfs:Class>

<rdfs:Class rdf:ID="BodyOfWater">
  <rdfs:subClassOf rdf:resource="#WaterSource"/>
</rdfs:Class>
```
Definition of classes and relationships in RDFS (continued)

```xml
<rdf:Property rdf:ID="emptiesInto">
  <rdfs:range rdf:resource="#BodyOfWater"/>
  <rdfs:domain rdf:resource="#River"/>
</rdf:Property>
```
Using RDF schema with RDF

Basically, we describe now real Web resources according to the developed schema

...<River rdf:ID="Yangtze"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.geodesy.org/water/naturally-occurring#">
<emptiesInto
  rdf:resource="http://www.china.org/geography#EastChinaSea"/>
</River>

...
What are the advantages of applying RDF Schema to this small RDF graph?

Inference!

We can infer new facts from already existing facts and classes and relationships defined in the RDF schema.

- River is a subclass of Stream → Yangtze is a stream
- Stream is a subclass of WaterSource → Yangtze is a WaterSource
- River emptiesInto BodyOfWater → EastChinaSea is a BodyOfWater
The newly inferred facts can be also used to improve information discovery.

For example, you are searching for streams that empty into EastChinaSea.

You get the info about Yangtze.
OWL is built on the top of RDFS

i.e., it extends RDFS

Work prior to OWL

DAML (developed by DARPA)

http://www.daml.org/

OIL

http://www.ontoknowledge.org/oil/index.shtml

OWL developed by the Web consortium (Recommendation)

http://www.w3.org/2004/OWL/
OWL is an extension of RDFS

It allows you to define richer relationships than RDFS

Inferencing capabilities are more powerful

Consequently, you can improve information discovery even more
Using OWL to define rich properties

```xml
<owl:ObjectProperty rdf:ID="connectsTo">
  <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"
  />
  <rdfs:domain rdf:resource="#WaterSource"/>
  <rdfs:range rdf:resource="#WaterSource"/>
</owl:ObjectProperty>
```
connectsTo relationship is symmetric

i.e., if A connectsTo B, then B connectsTo A

<River rdf:ID="Yangtze"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns="http://www.geodesy.org/water/naturally-occurring#">
   <connectsTo>
      <River rdf:about="http://www.china.org/rivers#Wu"/>
   </connectsTo>
</River>

Give me all rivers that connectTo Wu → Yangtze
Transitive property

<owl:ObjectProperty rdf:ID="containedIn">
  <rdf:type
      rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Sea"/>
  <rdfs:range rdf:resource="#BodyOfWater"/>
</owl:ObjectProperty>

i.e., if A is containedIn B and B is contained in C then A is contained in C
<Sea rdf:ID="EastChinaSea"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.geodesy.org/water/naturally-occurring#">
  <containedIn>
    <Sea rdf:about="http://www.china.gov#ChinaSea"/>
  </containedIn>
</Sea>
<Sea rdf:about="http://www.china.gov#ChinaSea"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.geodesy.org/water/naturally-occurring#">
<containedIn>
  <Ocean rdf:about="http://www.geodesy.org#PacificOcean"/>
</containedIn>
</Sea>

EastChinaSea is containedIn PacificOcean

subClassOf is a transitive property
Other properties of relationships

FunctionalProperty, there is at most one value for the property

InverseProperty

containedIn → contains

DatatypeProperty, the value of the property is a simple datatype

XML schema datatypes
Using OWL to improve information discovery while browsing

We have a typical directory of information

  e.g., similar to Yahoo categories

We can browse through categories and find the information we need
We can model categories as OWL classes

OWL classes are related with subClassOf relation

subClassOf relation is transitive

e.g., Class A is a subClassOf class B

i.e., an instance of class A is also an instance of class B

e.g., if a document belongs to a subcategory it also belongs to its supercategory
We had a lot of learning resources on Computer Science

We wanted to categorize them and improve information discovery

Firstly, we created an ontology of Computer Science

i.e., categorization of computer science fields

e.g., information systems, algorithms, etc.

Finally, we added resources to the categories

When you open a category you see all resources including resources from subcategories
“SuchAlgorithmus” is a subClassOf “Algorithmus”

“BinäreSuche” is an instance of “SuchAlgorithmus”

“BinäreSuche” is therefore an instance of “Algorithmus”

Additionally, a category can be a subclass of more than one superclass

Example

http://coronet.iicm.edu/navig
Using the system to personalize the information discovery

We define a personal class for each user

What happens if we declare the “Personal” class as a superclass of some class in Computer Science
e.g., Personal is a superclass of "Algorithmus"

All instances of "Algorithmus" are instances of "Personal"

We access these instances directly

I.e., we declare our interest in algorithms

A new instance in algorithms $\rightarrow$ we see it immediately

Example

http://coronet.iicm.edu/navig
Protege Editor
http://protege.stanford.edu/

IsaViz, visual authoring tool for RDF
http://www.w3.org/2001/11/IsaViz/

Jena RDF Framework, Java parser, Java API, RDQL
http://jena.sourceforge.net

W3C RDF Validator
http://www.w3.org/RDF/Validator/

As editor: any XML editor, emacs, etc.
Further Readings

- RDF Tutorial
  http://www710.univ-lyon1.fr/~champin/rdf-tutorial/

- Intro to RDF
  http://www.dlib.org/dlib/may98/miller/05miller.html

- RDF Tutorial and examples
  http://www.dstc.edu.au/Research/Projects/rdf/

- Semantic Web articles from
  http://www.xml.com
  http://www.xml.com/semweb/

- What is RDF?
  http://www.xml.com/pub/a/2001/01/24/rdf.html

- Metadata articles from
  http://www.xml.com
  http://www.xml.com/metadata/