Fortgeschrittenen-Praktikum Programming in the many

Alexander Derenbach, Harm Brandt, Peter Kauffels,
Mohammed Albari, Malte Tiedje, Oliver Wulf, Ulrich Schwarz

Christian-Albrechts Universität Kiel
Department of Computer Science and Applied Mathematics
Software Technology

8.2.05

Last compiled: 8th February 2005, 7:10 hrs

JCoMa

Java Conference Manager

Overview

Overview

Overview

Overview

Overview

Requirements

For JCoMa you need the following software:

- Apache Jakarta Tomcat >= 5.0
- Apache Ant >= 1.6
- Java >=jre1.5
- mysql >= 3.23.58

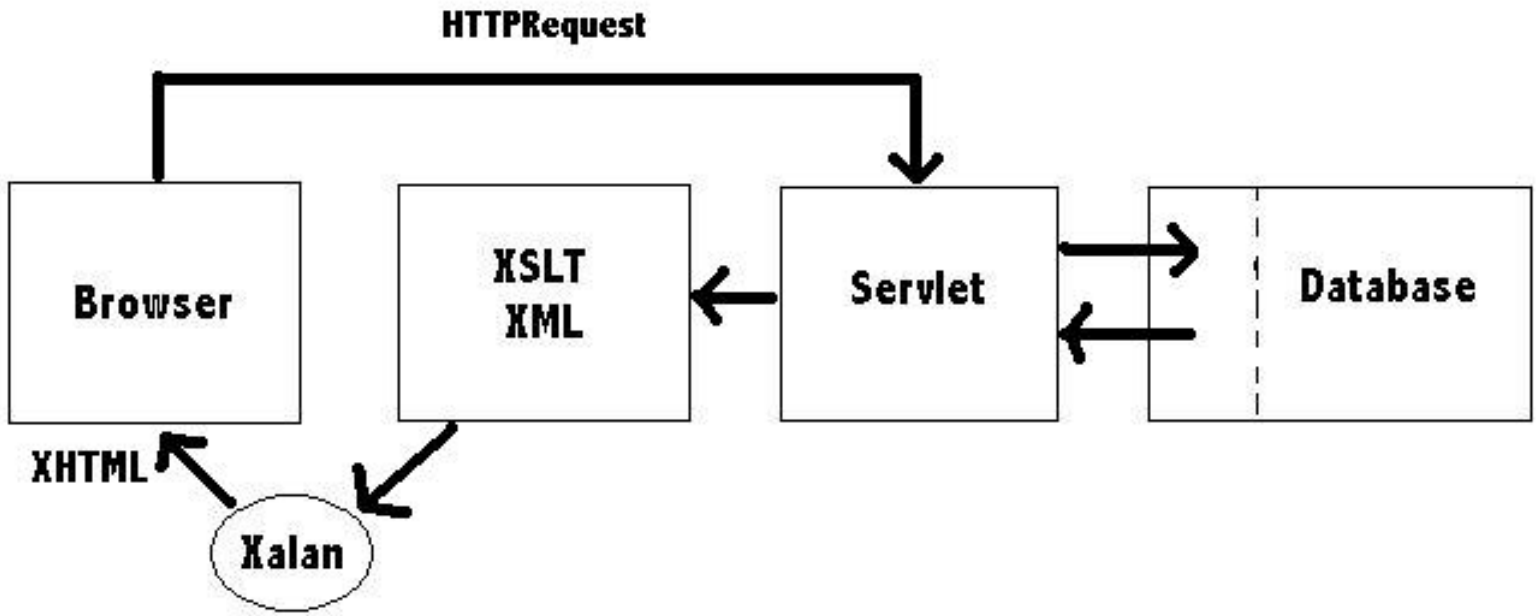
and jars

- xalan >= 2.6
- cos.jar from http://servlets.com/cos/
- log4j version 1.2.9
- mysql db driver Connector/J 3.1

Installation

1. extract jcoma.tar.gz
2. copy all *.jar in one directory
3. copy your tomcat servlet-api.jar to this directory
4. this is your external lib directory
   $\Rightarrow$ set the key "lib.home=" in build.properties
5. create a db in mysql
   $\Rightarrow$ set the db keys in build.properties
6. set the keys for tomcat,webapps in build.properties
7. set the keys for your conference admin in build.properties
8. be sure that Tomcat is running
9. run: ant remote-install
10. run: ant create-database
11. now you can log in with your admin email and password

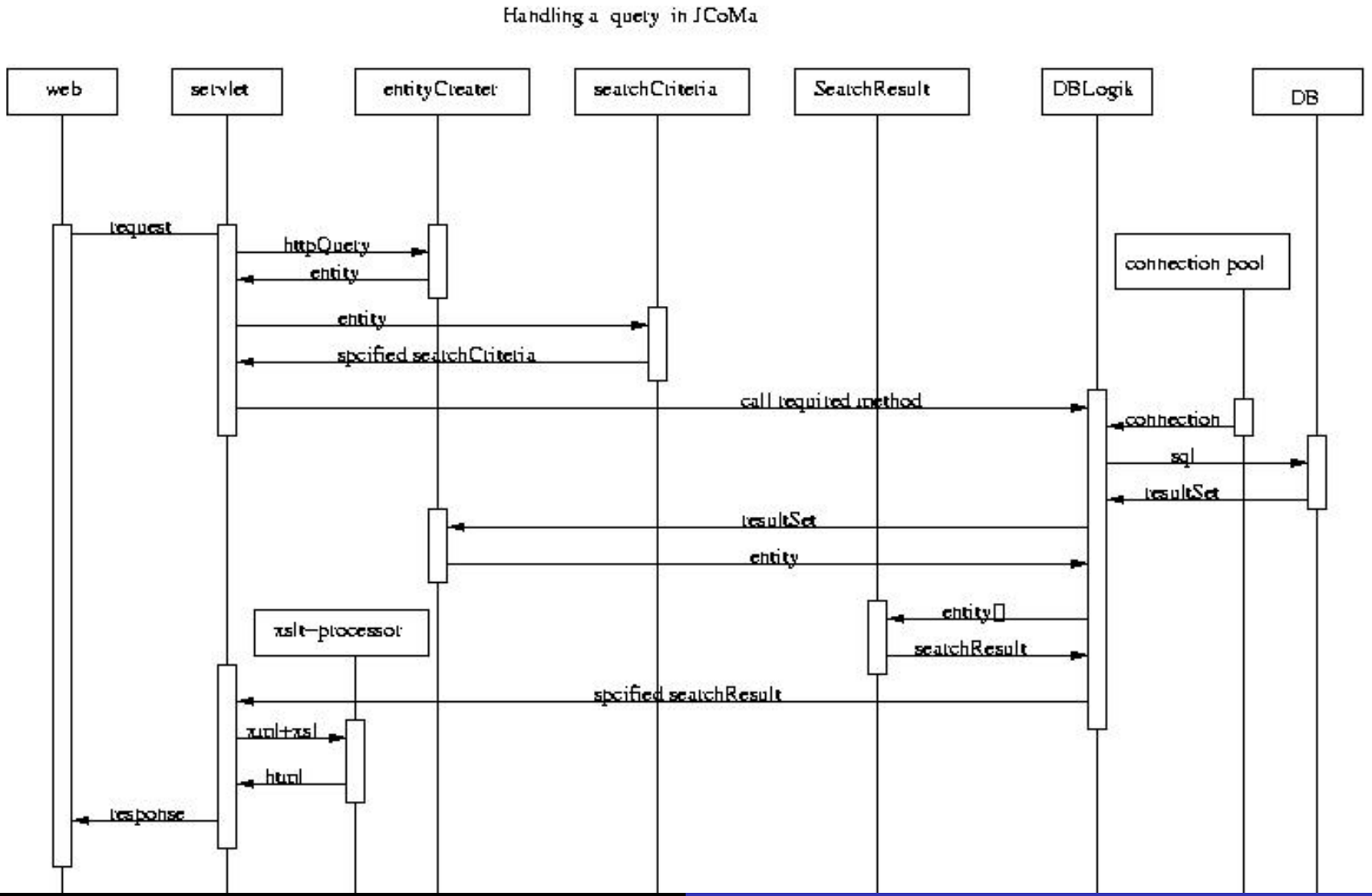
Layered architecture



Responsibilities

ald	hbr pka	mal	mti owu	ums
allocate- algorithm	Admin Chair	database setup	Author Login Subscribe	Reviewer Subscribe

Handling requests in JCoMa



Programming language: Java.

+ Type system: many bugs could be found at compile time.
+ Many speling error(sic!)-type bugs in user communication (names of form elements etc.) could be avoided through use of constant values. (But: couldn't reasonably be used in XSL.)
+ object structure made output easy (all "entities" provide a method `toXML()` in two flavours of verbosity. $\Longrightarrow$ no further problem with output, anywhere).

Programming language: Java.

- Much work until the first page is ever generated.
- Much error-prone work to get low-level typed DB data into high-level Java objects.
- very slow during debugging (add debugging statement, recompile, restart Tomcat, Tomcat memorial minute, navigate to problem, shut down Tomcat (write cache of log files!), look at new debugging, start over again)

XSLT output

+ Easier programming on Java level ("let's dump all info, the XSL style sheet can sort it out")
+ Easy extensibility: could theoretically[1] change the language of the user interface without touching the Java code.
– without possibility of validation: debugging difficult ("I wonder if it's `<person>` or `<Person>`...")
– yet another language to learn

[1] we haven't tried

Things we do better ... next time

- a better spec
- integrate more horizontal layers
- uniform exception handling
- more efficient meetings

We learned something!

All of us have gained experience in

- programming Servlets
- using XSLT
- using SQL
- using new functions in Java5