# 5. Unit: Web Services

**Exercise 5.1 (Web Services: tomcat)**

Install tomcat. You can do this on your own computer, and/or in the CIP Pool in your own account. Starting it makes it available under the URL localhost:8080 from the browser.

Copy the .war file for the XQuery Demo Web Service into tomcat/webapps and adapt the configuration (and restart tomcat again) to check whether it basically works.

**Exercise 5.2 (Web Service (Basics))**

Install the calendar application from the previous exercise as a Web-Service (to be accessible as `localhost:8080/calendar1/`).

a) Provide a (simple) Web interface as a form:
   - Show the calendar as an HTML page,
   - Load a calendar from the Web (a file calendar.xml in the public_html directory in your account is then accessible as `http://user.informatik.uni-goettingen.de/~`*username*`/calendar.xml`).
   - Insert entries. Preferably, the form should ask for date, time, duration, title.

b) Extend the java code with a method that generates for a given time interval (given by a start and end date) and a given duration (in hours) all collision-free appointments during weekdays between 8-18h, starting at full hours, ranging between 08:00 and 17:00). The result should be (i) primarily in form of an XML list of appointments in the same XML form as before.

c) Extend the Web interface with such that the method from (b) can be called, and the result list is shown in some useful HTML presentation.

**Exercise 5.3 (Web Service (Communication))**

For a simple scenario of communication between Web Services, just use a copy of the same code in the same tomcat server:

Install a second tomcat running under `localhost:8081` (adapt tomcat2/conf/server.xml) and do the same (if you have different calendar variants in your group, you might use these in the same tomcat).

The idea is now: User1 submits an enquiry as in (2b) "search for a possible slot of 2 hours [with the owner of Calendar2] in the next week" via the form of "his" Calendar1 (8080). Calendar1 collects his own possibilities and "asks" them via HTTP to Calendar2 (8081). Calendar2 reserves the first possible slot in itself, and returns this information to the Calendar1 (in the HTTP answer) that in turn also makes the reservation (and shows it as answer to User1).

a) For testing: feed both services with two different calendar1/2.xml.

b) Extend the code with a servlet method that is called via HTTP with a list of "proposals" as obtained from (2b), compares it with the own calendar, reserves the first possible slot and returns this information in the HTTP answer.

c) Extend the method from (2c) such that the result is not (only) shown as HTML page, but that a HTTP call to the other service's method from (3b) with the list of proposed appointments is executed, and that the answer is received and the reservation is made.

d) The simplest way is to read the HTTP XML message contents into a small DOM instance to process it.

e) [Optional] More "Web Service Feeling" can be obtained as follows: Reading the HTTP request in (3b) via SAX/StAX, checking each proposal immediately, and, if conflict-free, closing the reader and answering immediately.
   When submitting the HTTP message in (3c), sleep one second after writing each proposal (maybe it is necessary flush the writer), and choose a long interval to get a long list of proposals. Then, Calendar2 can already answer before Calendar1 has finished the generation of proposals (use logging, most simply by System.out.println(...) that can then be found in tomcat/logs/catalina.out.