

# Didax – a system for online testing: technical documentation

SANJA BABIĆ, CAMILLA BENGTTSSON & MATTIAS LINGDELL

**Reports from Uppsala Learning Lab —  
Digital Resources in the Humanities (DRHum) project  
Research reports (DRHumR)**



## DRHum — Digital Resources in the Humanities

“[T]he Wallenberg Global Learning Network [was] launched with the generous support of the Knut and Alice Wallenberg Foundation (KAW). In 1998, KAW donated \$15M over 5 years to Stanford University for the renovation of a campus building, Wallenberg Hall, and for a state-of-the-art center and network for global learning research associated with the Stanford Learning Lab. In 1999, this donation was supplemented with \$3M over 3 years for the establishment of a Swedish consortium of learning labs at Karolinska Institutet, the Royal Institute of Technology, and Uppsala University. These three institutions constitute the Swedish Learning Lab. The purpose of the network thus created around the Stanford Learning Lab and the Swedish Learning Lab is to promote learning across cultural and geographical bounds by developing human expertise and new learning technologies for education. [...]

The sub-project APE (Content archives, student portfolios & 3D environments) is an ongoing activity within the SweLL project "Meeting places for learning". The three tracks within APE:

Track A. Content and Context of Mathematics in Engineering Education (CCM),

Track B. Digital Resources in the Humanities (DRH)

Track C. 3D Communication and Visualization Environments for Learning (CVEL).”

(From the *Wallenberg Global Learning Network First Year Achievement Report*, 2001)

DRH—or DRHum, as we like to call it using a more easily pronounceable acronym (‘drum’)—consists of a set of interrelated activities investigating issues connected with the use of digital resources in humanities teaching and research at the university level. The members of the DRHum research team and their affiliations are:

PI: Lars Borin, Department of Linguistics, Uppsala University

PI: Jonas Gustafsson, Department of Teacher Education, Uppsala University

Karine Åkerman Sarkisian, Slavic Department, Uppsala University

Janne Backlund, Department of ALM, Aesthetics and Cultural Studies, Uppsala University

Camilla Bengtsson, Department of Linguistics, Uppsala University

Mattias Lingdell, Department of Linguistics, Uppsala University

György Nováky, Department of History, Uppsala University

John Rogers, Department of History, Uppsala University

Jan Sjunnesson, Department of Teacher Education, Uppsala University

We also collaborate with individuals and research groups inside and outside WGLN:

- Donald Broady, Director of Uppsala Learning Lab and scientific coordinator for APE
- Monica Langerth Zetterman, Uppsala member of the Swedish Learning Lab Assessment Team
- The Uppsala Learning Lab e-folio project led by Göran Ocklind
- The KTH Learning Lab Conzilla and Imsevimse APE CCM projects
- The LingoNet “web-based language laboratory” project at Mid-Sweden and Uppsala Universities
- The Nordic (Helsinki, Oslo, Stockholm/Uppsala) Squirrel project on corpus-based computer-assisted language learning

The main DRHum activities are:

- The development and evaluation of Didax, a web-based system for diagnostic language testing (Borin, Åkerman Sarkisian, Bengtsson, Lingdell)
- The use of digital picture archives and demographic databases in History courses (Nováky, Rogers)
- The use of biographical, historical and geopolitical databases and e-folios in teacher training (Gustafsson, Sjunnesson)
- The development of XML-based digital learning resources using emerging e-learning standards (Borin, Åkerman Sarkisian, Bengtsson, Lingdell, Backlund)

In the DRHumR (‘drummer’) research report series, the members of the DRHum team write about their work and their research findings. In the series, there will be status reports, technical documentation, evaluation reports, and preliminary versions of research articles which will appear elsewhere in a more polished format.



## Contents of this volume

Didax – a system for online testing: technical documentation <i>Camilla Bengtsson and Mattias Lingdell</i>	1
Didax – a system for online testing: technical documentation for the current implementation of teacher client 2 <i>Sanja Babić</i>	25



# Didax – a system for online testing: technical documentation

Camilla Bengtsson  
Mattias Lingdell  
Department of Linguistics,  
Uppsala University  
and  
Swedish Learning Lab

## 1 Introduction

Didax, the Digital Interactive Diagnostic Administering and Correction System, is a system for online testing. Didax uses the XML standard for describing questions and tests specified by the IMS Question & Test Interoperability Specification, <<http://www.imsproject.org/question/>>.

The current version of Didax allows students to take tests and teachers to grade tests. This version does not include the functionalities to create test questions and combine them into tests. These functions are intended for inclusion in future versions of Didax.

This document is the technical documentation for the current implementation of Didax.

### 1.1 Structure of this document

Section 2 describes the use cases in the Didax system, section 3 describes the overall design of Didax, section 4 describes the design of the server part of Didax, section 5 covers the client side of Didax and section 6 specifies the Didax-specific XML formats used by the system.

## 1.2 Abbreviations used in this document

CGI	Common Gateway Interface
Didax	Digital Interactive Diagnostic Administering and Correction System
HTML	Hypertext Markup Language
JDBC	Java Database Connectivity
JSDK	Java Servlet Development Kit
QTI	Question & Test Interoperability
XML	Extensible Markup Language

## 2 Use cases

The current implementation of Didax, as described in this report, defines three different use cases:

- The student takes a test. Consists of the following steps:
  1. The student launches the web browser.
  2. The student logs into the Didax system through the Login page with his/her username and password.
  3. The student's course information page is shown in the browser.
  4. The student chooses to take a test by clicking the appropriate link.
  5. The test is shown to the student.
  6. The student navigates between the questions and gives his/her answer by filling in HTML forms.
  7. The student submits his/her answers. Automatic correction of multiple choice questions and fill-in-blank questions is performed.
  8. The student logs out from Didax.
- The teacher grades the answers given by the student. Consists of the following steps:
  1. The teacher launches the web browser.
  2. The teacher logs into the Didax system through the Login page with his/her username and password.
  3. The teacher's course information page is shown in the browser.
  4. The teacher chooses a test to grade by clicking the appropriate link.
  5. The answers given by the students is shown in the browser.

6. The teacher grades and comments the students' answers by filling in HTML forms.
  7. The teacher returns to the course information page and marks that the test has been graded by clicking the appropriate link, thereby making the grades and comments available for the students to view.
  8. The teacher logs out from Didax.
- The student views his/her results.
    1. The student launches the web browser.
    2. The student logs into the Didax system through the Login page with his/her username and password.
    3. The student's course information page is shown in the browser.
    4. The student chooses which for test he/she wants to see the results by clicking a link.
    5. The grades and comments given by the teacher are shown in the web browser.
    6. The student views the results by navigating between the questions.
    7. The student logs out from Didax.

Other use-cases are intended to be included in the system but have not yet been implemented. These include for instance creating test questions and combining these into tests. These tasks are currently done manually, e.g. the task of converting XML to HTML.

### **3 Overall design of Didax**

Didax's main design is a client-server model. The server part is a database containing the information used in the system. Section 4 describes the server part of Didax. The client part consists of a number of java servlets that interact with the database through JDBC and with the users through a web browser. The server part of Didax is described in section 5.

### **4 Server-side design**

The server side of Didax consists of a database where all information used by the system is stored. The database model used is mainly a relational model except for the use of so-called binary large objects (BLOBs) which is not purely relational.

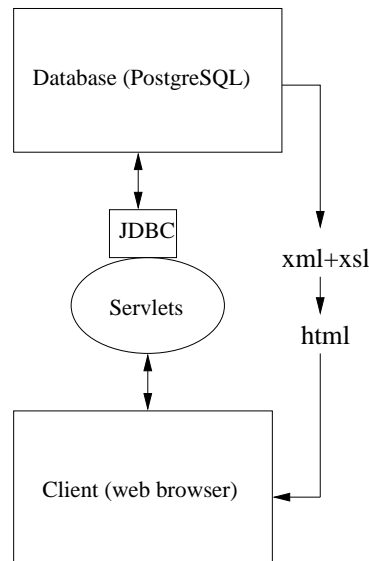


Figure 1: Overview of the design of the Didax system

The use of a database ensures that the data is stored in an efficient way and that searching can be done easily and efficiently.

#### 4.1 Technical information

The server part of Didax uses PostgreSQL, version 7.0, for database management. PostgreSQL is an open-source object-relational database management system. Psql, a terminal-based front-end to PostgreSQL, has been used to create and maintain the tables in the database.

The connection between the server (database) and the clients (servlets and web browser) is made with JDBC (Java Database Connectivity). JDBC is a Java API for communicating with databases. It provides the possibility to connect to a database, execute SQL commands and queries and to interpret the results. Functionality to handle blobs, i.e. binary large objects, is provided by the large object manager included in the PostgreSQL distribution.

#### 4.2 Database design

The database consists of 12 different tables: answers, courses, grades, groups, items, links, multimedia, resprocessing, test\_instances, tests, user\_courses and users.

The next sections describe the 12 tables in the Didax database regarding functionality and design. Each table is described and its attributes are listed in table form. The tables with attribute information contains columns with information about the attributes' name, type and key information (if applicable). The `attribute` column simply contains the name of the attribute. The `type` column contains the type of value this attribute has. These types are the types found in the PostgreSQL system and are explained in the documentation for PostgreSQL (e.g Momjian 200X).

Finally, the `key information` column specifies if the attribute is a primary or foreign key or if it is not a key. If the attribute is not a key this column contains only a dash (–). If the attribute is a foreign key, information about which table it references is also provided.

#### 4.2.1 Table answers

The table `answers` stores the students' answers to the tests. Each row in the table contains the answer to a question (an item in QTI terminology). The answers are organized according to a XML format designed especially for the Didax system. This XML format is described in section 6.1. The answers in XML format is stored in the table as binary large objects.

Attribute	Type	Description	Key information
<code>answer_nbr</code>	integer	Index number for a student's answers to a test instance.	primary key
<code>instance_nbr</code>	integer	The <code>instance_nbr</code> of the test instance the answer belongs to.	foreign key, references table <code>test_instances</code> .
<code>item_answer_nbr</code>	integer	Order number of the answer in the test instance, i.e. the order number of the item this answer belongs to.	primary key
<code>examinee</code>	integer	The user number of the examinee (student) this answer belongs to.	foreign key, references table <code>users</code> .
<code>answer_oid</code>	oid	The oid-number of the binary large object holding this answer in XML format.	–
<code>test_date</code>	timestamp	The date and time the answer was submitted.	–
<code>graded</code>	boolean	A boolean value indicating whether this answer has been graded.	–

### 4.2.2 Table courses

The table `courses` contains information about the courses in the Didax system. Each row in the table contains information about one course.

Attribute	Type	Description	Key information
<code>course_nbr</code>	integer	A unique identifier for this course.	primary key
<code>course_name</code>	text	The name of the course	–
<code>department</code>	text	Department giving the course. For example “eng” for English Department or “spa” for Spanish Department. How the department names are represented is not specified by the Didax system.	–
<code>subject</code>	text	The subject of the course. For example “eng” for English. How the subject names are represented is not specified by the Didax system.	–
<code>level</code>	text	The level of the course. For example A (in Swedish level system). How the level names are represented is not specified by the Didax system.	–
<code>insert_date</code>	timestamp	The date and time this course information was inserted into the database.	–
<code>semester</code>	text	Semester the course is/was given. For example vt 01 (Swedish example) How the semester names are represented is not specified by the Didax system.	–

### 4.2.3 Table grades

The table `grades` contains the teachers’ grading to the students’ answers to the test instances. Each row in the table contains the grading to one question (item in QTI). The gradings, containing of comments and points, are organized according to a to a XML format designed specially for the Didax system. This XML format is described in section 6.2. The gradings in XML format is stored in the table as

binary large objects.

Attribute	Type	Description	Key information
answer_nbr	integer	The answer_nbr (in table answers) this grading corresponds to.	primary key, foreign key, references answers
answer_order_nbr	integer	Order number of the answer in the test instance that this grading belongs to, i.e. the order number of the item this grading belongs to	–
points	float	The total amount of points awarded for the answer this grading corresponds to.	–
comments_oid	oid	The oid-number of the binary large object holding this grading in XML format.	–
graded_by	integer	user_nbr of teacher who has graded.	primary key, foreign key, references users

#### 4.2.4 Table groups

Table groups contains information of what different group codes used in other tables, e.g. users and test\_instances, stand for.

Attribute	Type	Description	Key information
group_code	text	Group code	primary key
explanation	text	Description of group code	–
insert_date	timestamp	Date and time group code and explanation was inserted into table	–

#### 4.2.5 Table items

Table items contains items (questions). Each line in the table holds information about one item. The items are in the XMLformat specified by the Question & Test Interoperability Specification (QTI). Items in the QTI specification corresponds to question in a test. The table contains items in XML format and information which describes the item, i.e. meta-data. This information is extracted from the XML when the item is inserted into the database.

<b>Attribute</b>	<b>Type</b>	<b>Description</b>	<b>Key information</b>
item_nbr	integer	Unique identifier for this item	primary key
item_title	text	Title of item. Extracted from XML attribute <code>title</code> .	–
status	text	Status of the item. Can be normal, experimental or retired. Extracted from XML element <code>qmd_status</code>	–
subject	text	Subject of item. For example eng for English. How the subject names are represented is not specified by the Didax system. Extracted from XML element <code>qmd_topic</code> .	–
level	text	The level of the item. For example A (in Swedish level system). How the level names are represented is not specified by the Didax system. Extracted from the XML element <code>langstring</code> .	–
difficulty	smallint	The difficulty of the item. Can range from 0 (easiest) to 4 (hardest). Extracted from XML element <code>difficulty</code> .	–
q_type	text	The type of the item e.g. Grammar or Vocabulary. Extracted from XML element <code>langstring</code> .	–
q_form	text	The rendering type of the item e.g. Hotspot or Fib. Extracted from XML element <code>qmd_renderingtype</code> .	–
r_form	text	The response type of the item. Can be single, ordered or multiple. Extracted from XML element <code>qmd_responsetype</code> .	–
item_oid	oid	The oid of the binary large object containing the item in XML format.	–
course_nbr	integer	course_nbr of the course this item is intended for.	foreign key, references courses

Attribute	Type	Description	Key information
author	integer	user_nbr of user who inserted the item into the database.	foreign key, references users
i_mod	integer	Indicates which, if any, item this item is an modification of. If an item is modified both the original and the modified item is kept in the table i.e. the old item will not be deleted.	foreign key, references items
insert_date	timestamp	Date and time the item was inserted.	–
ident	text	The items ident string. Extracted from the XML attribute ident.	–
max_score	smallint	The item's maximum score. Extracted from XML element qmd_maximumscore.	–

#### 4.2.6 Table links

Table links contains information about the links the teachers can use as a part of the grading. The links are added during the grading through the teacher client described in section 5.2.3. The information about the link's name and location is structured in a XML format especially designed for Didax. This XML format is described in section 6.3. The table also has information about who is allowed to see the link, which teacher who inserted the link and to which test instance the link belongs.

Attribute	Type	Description	Key information
link_nbr	integer	Unique identifier for the link.	primary key
for_who	integer	user_nbr of the student this link is intended for. Has no effect if link_view is set to all.	foreign key, references users
inserted_by	integer	user_nbr of teacher who added the link.	foreign key, references users
link_oid	oid	number of binary large object storing the link information as XML.	–
insert_date	timestamp	date and time the link was inserted.	–

Attribute	Type	Description	Key information
instance_nbr	integer	instance_nbr of test instance this link belongs to.	foreign key, references test_instances
link_view		Indicates who is allowed to see the link. If set to <code>student</code> , only student specified by <code>for_who</code> can see the link. If set to <code>all</code> , all students taking the test instance can see the link.	–

#### 4.2.7 Table multimedia

Table multimedia stores all multimedia files used in the different tests, e.g. pictures, audiofiles and video sequences. The files are stored in the table as binary large objects. Information about what sort of file it is, e.g. video or audio, the name of the file, the file's extension, user\_nbr of the user who inserted the file and when it was inserted is also stored in the table. When a multimedia file is used by a test instance the idea is that it is to be put in a specific multimedia directory, from which it can be accessed when needed by the clients. Since the client responsible for creating tests has not yet been constructed, this has not yet been implemented.

Attribute	Type	Description	Key information
file_nbr	integer	An unique identifier for the multimedia file.	primary key
file_name	text	The file's name, including extension.	–
file_oid	oid	The oid number of the binary large object storing the file.	–
file_type	text	Type of file e.g. audio or image.	–
file_ext	text	The file's extension e.g. .jpg or .gif.	–
inserted_by	integer	user_nbr of user who inserted the file.	foreign key, references users
insert_date	timestamp	Date and time file was inserted into table.	–

#### 4.2.8 Table resprocessing

Table `resprocessing` stores the QTI-elements `resprocessing` and `itemfeedback`. These elements store information used by the Didax system when automatic grading is performed and are extracted from the items when they are inserted into the database. The `resprocessing` element contains information about points for different responses to a question and has pointers to appropriate feedback. The feedback is stored in the `itemfeedback` element. For more information about these QTI-elements, consult the IMS-QTI documentation available online at <http://www.imsproject.org/question/>. The `resprocessing` and `itemfeedback` elements are stored as binary large objects in the table. The table also holds information about which item the `resprocessing` and `itemfeedback` elements belong to, when they were first inserted into the table and when they were last updated. Each row in the table holds the `resprocessing` and `itemfeedback` information for one item (question).

Attribute	Type	Description	Key information
<code>item_nbr</code>	integer	<code>item_nbr</code> of item this <code>resprocessing</code> and <code>itemfeedback</code> belong to.	foreign key, references items
<code>resprocessing_oid</code>	oid	oid number of the binary large object holding this <code>resprocessing</code> element.	–
<code>itemfeedback_oid</code>	oid	oid number of the binary large object holding this <code>itemfeedback</code> element.	–
<code>insert_date</code>	timestamp	Date and time this information was first inserted into the table.	–
<code>last_updated</code>	timestamp	Date and time this information was last updated.	–

#### 4.2.9 Table test\_instances

Table `test_instances` stores information about the different test instances in the system, i.e. which test type the test instance is, which course it belongs to, between which times it is available, which teacher is responsible for the test instance, who it is intended for, when it was inserted into the table, which semester it belongs to and if it has been graded or not.

Attribute	Type	Description	Key information
instance_nbr	integer	Unique identifier for this test instance.	primary key
test_nbr	integer	test_nbr (from table tests) of the test this test instance is an instance of.	foreign key, references tests
course_nbr	integer	course_nbr (from table courses) of the course this test instance belongs to.	foreign key, references courses
start_time	timestamp	Date and time this test instance becomes available.	–
end_time	timestamp	Date and time this test instance becomes unavailable, i.e. answers to this test instance must be submitted before this date and time.	–
from_who	integer	user_nbr (from table users) of the teacher responsible for the test.	foreign key, references users
to_whom	text	Group codes of the groups this test instance is intended for. The group codes are separated by vertical bars ( ). Not used in this implementation of Didax.	–
insert_date	timestamp	date and time the information about this test instance was inserted into the table.	–
semester	text	Semester this test instance belongs to. For example vt01 (Swedish example). How the semester names are represented is not specified by the Didax system.	–
graded	boolean	Boolean value indicating whether this test instance has been graded.	–

#### 4.2.10 Table tests

Table tests contains information about the different tests, i.e. the different test types currently in the system. Each row in the table corresponds to one item in

a test (assessment in QTI-terminology). The value of the attribute `item_nbr` is the item's order number in the test. Information is also provided about who constructed the test and when it was constructed. The items are stored as binary large objects in the table and an attribute in the table also holds the `item_nbr` of the item, corresponding to the `item_nbr` in table `items`. The reason why the items are stored as binary large objects in the table and not only linked to is that if an item is removed from table `items` it must have no effect on the tests that contains the removed item.

Attribute	Type	Description	Key information
<code>test_nbr</code>	integer	Identifier for the test.	primary key
<code>test_name</code>	text	Name of test.	–
<code>item_nbr</code>	integer	This item's order number in the test. Each row holds one item.	primary key
<code>item_oid</code>	oid	The oid number of the binary large object holding this item.	–
<code>author</code>	integer	<code>user_nbr</code> (from table <code>users</code> ) of the user who has constructed the test.	foreign key, references <code>users</code>
<code>insert_date</code>	timestamp	Date and time this test was constructed.	–
<code>items_ref</code>	integer	<code>item_nbr</code> (from table <code>items</code> ) of the item in this table row. Used for resprocessing.	foreign key, references <code>items</code>
<code>section_title</code>	text	Contains a section name if a section in the test starts with this item.	–

#### 4.2.11 Table `user_courses`

Table `user_courses` stores information about which different roles the users in the system have on different courses. Each row in the table holds information about one user's function on a course. A user may only have one function (role) per course. Each user may also be in one or more groups on the course. For example, to be allowed to grade a test instance the user has to be in the group `g` (as in graded).

Attribute	Type	Description	Key information
user_nbr	integer	The user_nbr (from table users) of the user.	primary key, foreign key, references users
role	text	The user's role on the course. Can be student or teacher.	–
course_nbr	integer	course_nbr (from table courses) of the course this user and role belongs. to	primary key, foreign key, references courses
insert_date	timestamp	Date and name the information was inserted into the table.	–
groups	text	Group codes of the groups the user belongs to on this course, if any. The group codes are separated by vertical bars ( ).	–

#### 4.2.12 Table users

Table users stores user information about the users in the system i.e. username, password, group info and real name. The group info allows users to be member of groups that can be used for access control. In the present implementatin of Didax the group info is not used. Each row in the table contains the user information of one user. The table always contains a special computer user used for automatic grading.

Attribute	Type	Description	Key information
user_nbr	integer	Unique identifier for the user.	primary key
user_name	text	The user's user name used when logging in.	–
password	text	Thes user's password used when loggin in.	–
groups	text	Group codes of the groups the user is in, if any. The group codes are separated by vertical bars ( ).	–
insert_date	timestamp	Date and time the information was inserted into the table.	–
name	text	The user's real name e.g. Thomas Magnum.	–

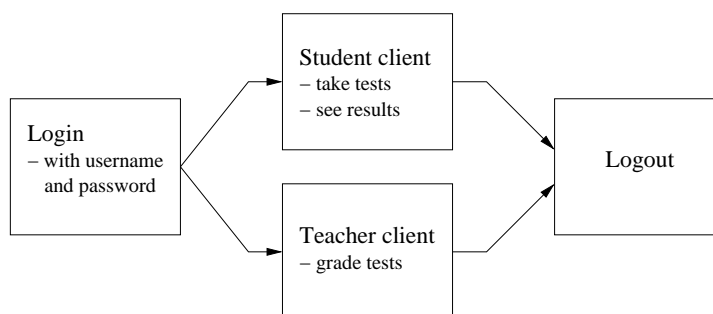


Figure 2: The main parts of the Didax system

## 5 Client-side design

The client side of the Didax system consists of two main clients: a student client and a teacher client. The student client allows the student to take tests and view his or her results and the teacher client allows the teachers to view the students' answers to the tests and grade them. Before the user can access the clients a login process must be undertaken. The login process simply asks for the user's username, password and role (teacher or student) and redirect the user to the appropriate client. After using the system, the user can logout. All communication between the users and the system is performed through a normal web browser. Below is a picture of the main steps when using the Didax system.

In order to prevent users from using the system in an uncontrolled way, for example by manually entering addresses in the web browser, the system uses the technology of sessions. An example of misusing the system can be to try to bypass the login process by entering the address of the client directly in the browser. Another example is to send the address of a test, for example by e-mail, to another person who types the address in his or her web browser and thereby gets access to the test. A session is tied to a specific login and checks are made in the system to make sure that the same session is used when accessing the different parts of the system.

### 5.1 Technical information

The users of the system interacts with the system using a web browser. The clients was developed using Netscape 4.7x for Linux, but should work well with both Netscape and Internet Explorer of versions 4.0 and up.

The clients were written using jdk java 1.2 and JSDK 2.0. Java servlets provides CGI-like functionality to the java language.

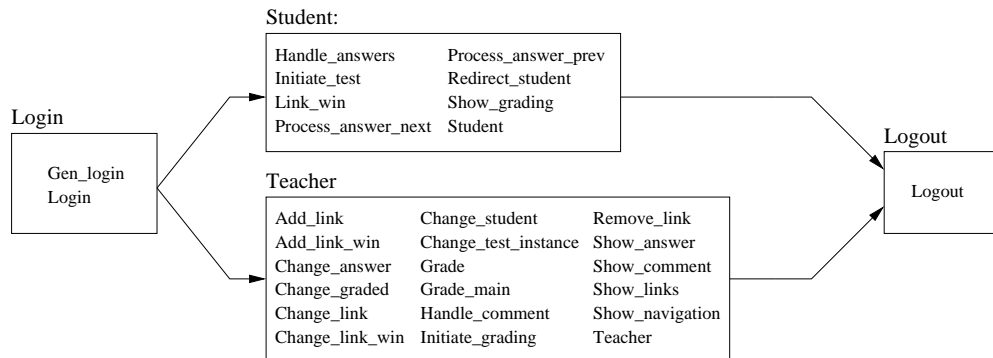


Figure 3: The servlets used by the different parts

The development of the system was made on a Red Hat 6.2 system running Apache web server 1.3.14.

The java servlets session capabilities was used for session management. JSDK 2.0 has good support for session handling, but later version of JSDK provide even more functionality. The reason JSDK 2.0 is used is that the version of Apache that was used does not support newer versions of JSDK.

## 5.2 Description of the client-side parts

This section describes the different parts of the client-side of the didax system. The client-side consists of, as explained above, four main parts: login process, student client, teacher client and logout process. Each of the parts uses a number of servlets for their functionality. The picture below shows which servlets are used by which part of the system.

### 5.2.1 Login process

The login process simply lets the user login using his or her username and password. The user can choose which function, teacher or student, he or she wants to login as. It is possible for a user to have the function of both student and teacher at the same time in the system, though not on the same course. The login process uses two servlets:

Servlet name	Description
Gen_login	Generates the HTML code to display the Didax login page
Login	Checks the user's username and password against the database and redirects to the appropriate client

### 5.2.2 Student client

The student client allows the student to take tests and to see the results. These two use-cases are described in section 2. The student client uses 8 servlets:

<b>Servlet name</b>	<b>Description</b>
Handle_answers	Processes the user's answers to a test, converts them to the XML format described in section 6.1 and inserts them into the database (table answers). Also performs automatic grading of multiple choice questions and fill-in-blank questions.
Initiate_test	Collects information needed to take the test the student has chosen to take e.g. collects the questions belonging to the test and other preparations. Redirects to servlet test.
Link_win	Window that shows the links belonging to the test.
Process_answer_next	Processes the answers the student has given in the HTML forms. Sees to that the same information is in the form if student returns to the question. Called when student presses button to see the next question.
Process_answer_prev	Processes the answers the student has given in the HTML forms. Sees to that the same information is in the form if student returns to the question. Called when student presses button to see the previous question.
Redirect_student	Redirects to students course page.
Show_grading	Shows the points and comments given by the teacher to the student's answer to a test.
Student	Generates the student's course page which has information about the courses and tests available to the student. From here the student can choose to take tests or see results of tests.

Figure 4 shows the main servlets' functions and how they interact.

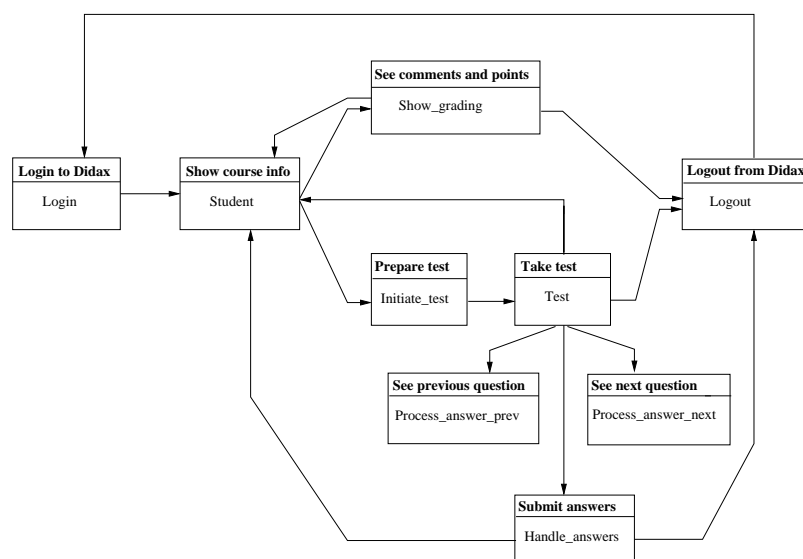


Figure 4: The functions of the main servlets in the student client and their interaction. The bold text at the top of the boxes describes functions in the system and the text inside the boxes are name of servlets.

### 5.2.3 Teacher client

The teacher client allows the teacher to grade the answers the students have given to the tests. This use case is described in section 2. The teacher client uses 18 servlets:

<b>Servlet name</b>	<b>Description</b>
Add_link	Create a link in the XML format described in section 6.3 and insert it into the database.
Add_link_win	Generates the HTML form used by the teacher to create a new link.
Change_answer	Changes which answer that is displayed to the teacher when grading a test.
Change_graded	Changes the status of a test instance, i.e. changes a test instance from having the status graded to status not graded and vice versa. This servlet is called from the teacher's course information page.
Change_link	Changes link information, e.g. the link's name, reference name and address.

<b>Servlet name</b>	<b>Description</b>
Change_link_win	Generates the HTML form used by the teacher to change link information.
Change_student	Changes which student whose answers is displayed to the teacher when grading a test.
Change_test_instance	Changes which test instance that is displayed to the teacher when grading a test.
Grade	Generates the three different frames on each vertical side of the grading screen, i.e. the navigation frame, the comment frame and the frame where the answer is displayed.
Grade_main	Generates the two different vertical frames displayed when grading a test. Each frame contains three horizontal frames generated by servlet Grade.
Handle_comment	Processes the comment given by the teacher to an answer, structures the information according to the XML format described in section 6.2 and inserts it into the database.
Initiate_grading	Collects information needed to grade a test, e.g. all the answers given to the test by the students.
Remove_link	Removes a link from the database
Show_answer	Displays the answer by a student to a question. Called from servlet Grade.
Show_comment	Displays the comment, if any, previously given to a student's answer to a question.
Show_links	Displays the links associated with this test instance.
Show_navigation	Displays the navigation used when grading a test. Allows the teacher to change answer (servlet Change_answer), test instance (servlet Change_test_instance) and student (servlet Change_student).
Teacher	Generated the teacher's course information page which has information about the courses and tests available to the teacher. From here the teacher can choose which test instance to grade.

The picture below shows the main servlets' functions and how they interact.

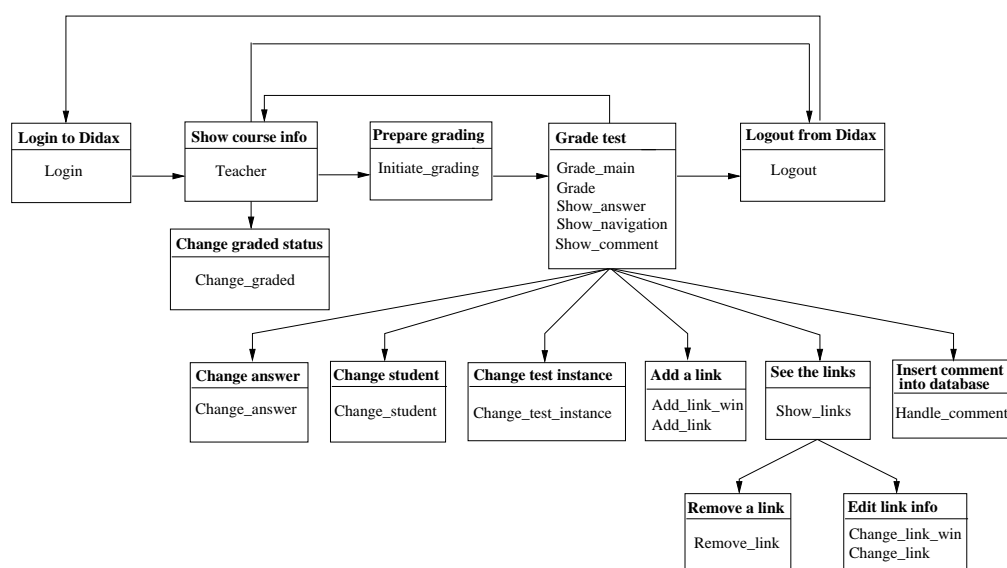


Figure 5: The functions of the servlets in the teacher client and their interaction. The bold text at the top of the boxes describes functions in the system and the text inside the boxes are name of servlets.

### 5.3 Logout process

The logout process simply allows the user to logout from the system. When a user logs out the session associated with the login is terminated. The logout process uses one servlet named Logout.

### 5.4 Other classes used by the servlets

There are also a number of classes used by the different servlets to perform specific tasks:

Name	Description
DatabaseC	Contains methods to interact with the database via JDBC. This class is used by all servlets that communicates with the database.
File_utilities	Provides various methods for handling files, e.g. converting files to strings or string arrays
ImageServlet	Used by tests that shows images. Fetches an image from the file system and displays sends it to the web browser

## 6 The Didax-specific XML formats

The Didax system uses a number of specially designed XML formats. These formats describe how answers to tests, comments and points to answers and links are to be structured. The next sections describes these XML formats and provides DTDs for them. In each section there is a table over the elements and attributes specified by the XML format. These tables has three columns: `name` – the name of the element/attribute, `type` – element or attribute, and `description` – a description of the element/attribute.

### 6.1 `didax_answer`

The XML structure `didax_answer` describes how the students' answers to tests are to be structured.

Name	Type	Description
<code>didax_answer</code>	element	The root element.
<code>item_answer</code>	element	The element containing the student's answer to an item.
<code>i_ref</code>	attribute	The ident string of the item this answer belongs to.
<code>choice_response</code>	element	Answer to a multiple choice question.
<code>fib_string_response</code>	element	Answer to a fill in blank question.
<code>fib_text_response</code>	element	Answer to an essay type question.
<code>name</code>	element	Name of radio button, checkbox, textfield or textarea used to enter answer.
<code>value</code>	element	Value of radio button or checkbox used to enter answer.
<code>string</code>	element	Value of textfield used to enter answer.
<code>text</code>	element	Value of text area used to enter answer.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE didax_answer SYSTEM "didax_answer.dtd">
<!DOCTYPE didax_answer [
  <!-- root element -->
    <!ELEMENT didax_answer (item_answer)>
  <!-- answer to an item -->
    <!ELEMENT item_answer
      ((choice_response|
        fib_string_response|
        fib_text_response)+)>
      <!ATTLIST item_answer i_ref CDATA #REQUIRED>
  <!-- response with radio button or checkbox -->
    <!ELEMENT choice_response (name,value)>
  <!-- response with textfield -->
    <!ELEMENT fib_string_response (name,string)>
  <!-- response with text area -->
    <!ELEMENT fib_text_response (name,text)>
  <!-- name of radio button, checkbox, textfield or text area -->
    <!ELEMENT name (#PCDATA)>
  <!-- value of radio button or checkbox -->
    <!ELEMENT value (#PCDATA)>
  <!-- value of textfield -->
    <!ELEMENT string (#PCDATA)>
  <!-- value of text area -->
    <!ELEMENT text (#PCDATA)>
]>
```

## 6.2 didax\_grade

The XML structure `didax_grade` describes how the teachers' comments and points are structured:

Name	Type	Description
<code>didax_grade</code>	element	The root element
<code>item_grade</code>	element	Element containing grading to an item.
<code>sub_item_grade</code>	element	Element containing grading of subitem (part of question).
<code>name</code>	attribute	The name of the subitem. Extracted from the HTML form element names.
<code>value</code>	attribute	The value of the subitem. Extracted from the HTML form element values.
<code>correct</code>	element	Indicates whether an answer is correct or not. Can have values <code>yes</code> , <code>no</code> or <code>n/a</code> (for essay type questions).
<code>points</code>	element	Points awarded for the answer to this subitem.
<code>comment</code>	element	Comment to the answer of this subitem.
<code>item_points</code>	element	Points awarded for the item, i.e. or all subitems together.
<code>item_comment</code>	element	General comment to the item, i.e. all subitems.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE didax_grade SYSTEM "didax_grade.dtd">
<!DOCTYPE didax_grade [
  <!-- root element -->
    <!ELEMENT didax_grade (item_grade)>
  <!-- element containing grading to item -->
    <!ELEMENT item_grade (sub_item_grade+,item_points,item_comment)>
  <!-- element for grading of subitem -->
    <!ELEMENT sub_item_grade (correct,points,comment)>
  <!-- name and value if subitem
    <!ATTLIST sub_item_grade name CDATA #REQUIRED>
    <!ATTLIST sub_item_grade value CDATA #REQUIRED>
  <!-- points and comment for item -->
    <!ELEMENT item_points (#PCDATA)>
    <!ELEMENT item_comment (#PCDATA)>
  <!-- whether answer to subitem is correct, -->
  <!-- values 'yes', 'no' or 'n/a'
    <!ELEMENT correct (#PCDATA)>
  <!-- points for the answer to the subitem -->
    <!ELEMENT points (#PCDATA)>
  <!-- comment for the answer to the subitem -->
    <!ELEMENT comment (#PCDATA)>
]>
```

### 6.3 `didax_link`

The XML structure `didax_link` describes how information about the links used by the teachers and displayed to the students are structured.

Name	Type	Description
<code>didax_link</code>	element	The root element.
<code>link</code>	element	Element holding link info.
<code>ref</code>	element	Reference name for the link.
<code>name</code>	element	The name of the link.
<code>address</code>	element	The address of the link.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE didax_link SYSTEM "didax_link.dtd">
<!DOCTYPE didax_link [
  <!-- root element -->
    <!ELEMENT didax_link (link)>
  <!-- link info -->
    <!ELEMENT link (ref,name,address)>
  <!-- reference name for link -->
    <!ELEMENT ref (#PCDATA)>
  <!-- name of link -->
    <!ELEMENT name (#PCDATA)>
  <!-- address of link -->
    <!ELEMENT address (#PCDATA)>
]>
```

## References

Momjian, Bruce 2001. *PostgreSQL Reference Manual*.

# **Didax – a system for online testing: technical documentation for the current implementation of teacher client 2**

Sanja Babić

Department of Linguistics, Uppsala University  
and  
Uppsala Learning Lab, Uppsala University

## 1 Introduction

The teacher client 2 is supposed to allow teachers to create test questions, combine them into tests as well as to publish tests for students. All its functions have not yet been implemented.

This document is the technical documentation for the current implementation of the teacher client 2.

### *1.1 Structure of this document*

Section 2 describes the use cases in the teacher client 2, section 3 describes the design of the teacher client 2, section 4 specifies some suggestions for the client development in the future, section 5 specifies paths to system files.

## 2 Use cases

The current implementation of the teacher client 2 defines four different use cases:

- The teacher publishes a test instance . Consists of the following steps:
  1. The teacher launches the web browser.
  2. The teacher logs into the Didax system through the Login page
  3. The teacher's course information page is shown in the browser.
  4. The teacher chooses to publish a test instance by clicking the appropriate link that leads the teacher to the main menu for test creation.
  5. The main menu for test creation is shown.
  6. The teacher chooses the course that he/she wants to publish a test instance for by clicking the appropriate link
  7. All tests for the chosen course are shown.
  8. The teacher can see each test separately by clicking the appropriate question number.
  9. By clicking the appropriate link the teacher chooses the test he/she wants to publish an instance of.

10. The teacher specifies the period of time the test instance is to be available for students.
  11. The teacher submits the information needed to publish the test instance.
  12. The test instance is published automatically.
  13. The teacher logs out from Didax.
- The teacher creates a test. Consists of the following:
    1. The teacher launches the web browser.
    2. The teacher logs into the Didax system via the Login page
    3. The teacher's course information page is shown in the browser.
    4. The teacher chooses to create a test by clicking the appropriate link that leads the teacher to the main menu for test creation.
    5. The main menu for test creation is shown.
    6. The teacher chooses the course that he/she wants to create a test for by clicking the appropriate link.
    7. The form used by the teacher to create a test is shown. The teacher can choose questions after difficulty (from 0 to 4) and/or type (i.e. grammar or vocabulary).
    8. The questions specified by the teacher in the previous form are shown.
    9. The teacher can see each item separately by clicking the appropriate question number.
    10. The teacher chooses questions that the test is to be consisted of by clicking the appropriate question link.
    11. The questions specified by the teacher in the previous form are shown. These are the questions that the test is to be consisted of.
    12. The teacher specifies where each part of the test starts and gives a name to the parts as well as the whole test. (This is obligatory because of the XML document).
    13. The teacher submits information needed for the test to be created.
    14. The test is created automatically in the database. (OBS this part has not been finished yet. See section 4.)
    15. The teacher logs out from Didax.
  - The teacher creates a *fill-in-blank* question. Consists of the following:
    1. The teacher launches the web browser.
    2. The teacher logs into the Didax system through the Login page
    3. The teacher's course information page is shown in the browser.
    4. The teacher chooses to create a question by clicking the appropriate link that leads the teacher to the main menu for test creation.
    5. The main menu for test creation is shown.
    6. The teacher chooses the course that he/she wants to create a question for by clicking the appropriate link.
    7. The form used by the teacher to create either a *fill-in-blank question* or a *choice question* is shown.
    8. The teacher fills in the form and chooses to create a fill-in question.
    9. The form used by the teacher to create a fill-in-blank question is shown.
    10. The teacher fills in the form by writing the main question and specifying the number of subquestions.

11. The form with the number of subquestions specified is generated automatically.
  12. The teacher writes subquestions.
  13. By clicking the appropriate link the teacher chooses to see the whole question.
  14. The new question is shown.
  15. The teacher gives the answer by filling in the form.
  16. The teacher submits the question as well as the answer.
  17. The teacher logs out from Didax.
- The teacher creates a *choice* question. Consists of the following:
    1. The teacher launches the web browser.
    2. The teacher logs into the Didax system through the Login page
    3. The teacher's course information page is shown in the browser.
    4. The teacher chooses to create a question by clicking the appropriate link that leads the teacher to the main menu for test creation.
    5. The main menu for test creation is shown.
    6. The teacher chooses the course that he/she wants to create a question for by clicking the appropriate link.
    7. The form used by the teacher to create either a *fill-in-blank question* or a *choice question* is shown.
    8. The teacher fills in the form and chooses to create a choice question.
    9. The form used by the teacher to create a choice question is shown.
    10. The teacher fills in the form by writing the main question and specifying the number of subquestions and subquestions' options.
    11. The form with the number of subquestions specified, as well as their options, is generated automatically.
    12. The teacher writes subquestions and their options.
    13. By clicking the appropriate link the teacher chooses to see the whole question.
    14. The new question is shown.
    15. The teacher gives the answer by clicking the right option in the form.
    16. The teacher submits the question as well as the answer. (OBS not finished yet, see section 4).
    17. The teacher logs out from Didax

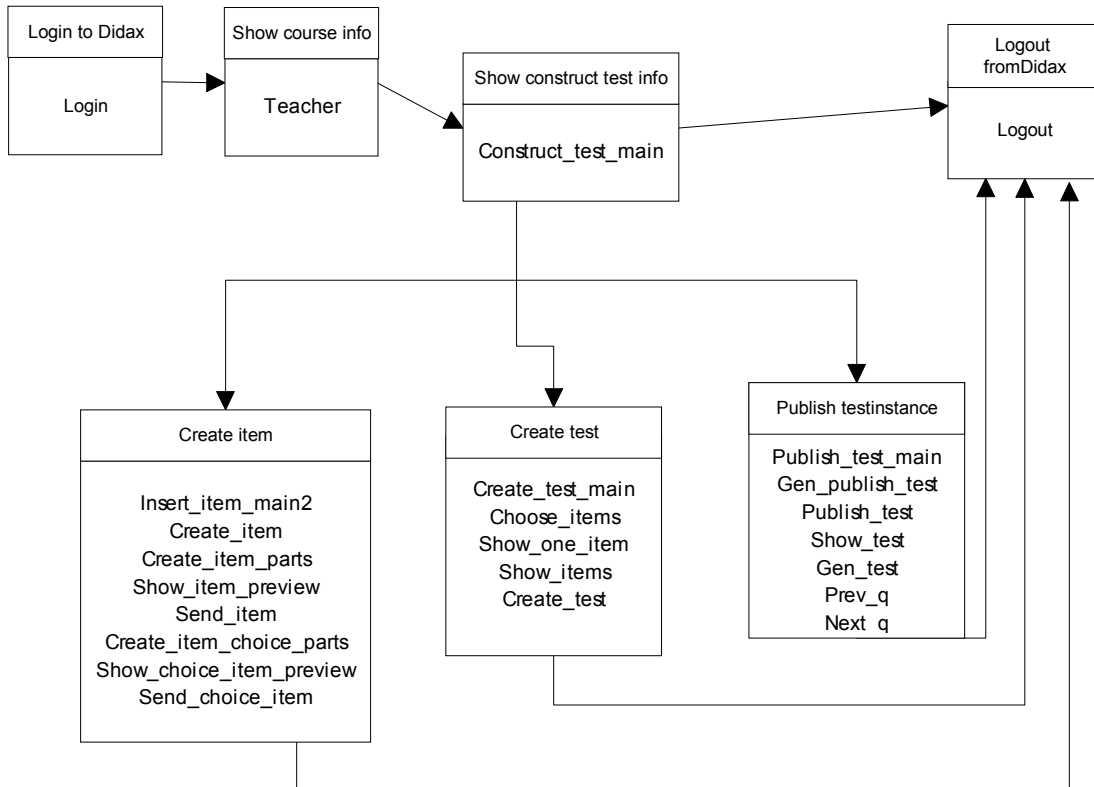
### 3 Design of the teacher client 2

The current version of the teacher client 2 consists of 21 servlets. It allows the teacher to create a test questions, combine them into a test and publish a test for students. These use cases are described in section 2.

<b>Servlet name</b>	<b>Description</b>
Choose_items	Displays the items specified in servlet Create_test_main. It also allows the teacher to see each item (servlet Show_one_item). From here the teacher can choose items that a test is to be consisted of.
Construct_test_main	Generates the teacher's course information page which has information about the courses and from here the teacher can choose to enter an item to the test item bank, to construct a test or to publish a testinstance.
Create_item	Generates the HTML form used by the teacher to create either a <i>fill- in-blank item</i> or a <i>choice item</i> .
Create_item_choice_parts	Generate the HTML form used by the teacher to create a <i>choice item</i> .
Create_item_parts	Generates the HTML form used by the teacher to create a <i>fill –in-blank item</i> .
Create_test	Creates a test and puts it in the test table in the database. (NB: not yet finished, see section 4 for necessary changes)
Create_test_main	Generates the HTML form used by the teacher to create a test. The teacher can choose questions after difficulty (from 0 to 4) and/or type (i.e. grammar or vocabulary).
Gen_publish_test	Generates the HTML form used by the teacher to publish a test instance to be available to students for a given period of time.
Gen_test	Collects information needed to display the test the teacher has chosen to see. Redirects to servlet Show_test.
Insert_item_main	Generates the HTML form used by the teacher to create a new item. It is possible to create two types of items: <i>fill- in-blank</i> and <i>choice</i> items (just now).
Next_q	Displays the next question. Called when the teacher presses button to see the next question.

<b>Servlet name</b>	<b>Description</b>
Prev_q	Displays the previous question. Called when the teacher presses button to see the previous question.
Publish_test	Creates a test instance and puts it in the test_instance table in the database. (OBS not yet finished, see section 4 for necessary changes)
Publish_test_main	Generates the teacher's test information page that has information about all tests for the course chosen. From here the teacher can choose which test to publish.
Send_item	Processes <i>the fill- in-blank item</i> (as well as the correct answers) created by the teacher, structures the information in the XML format and inserts it into the database (items table and resprocessing table).
Send_item_choice	Processes <i>the choice item</i> (as well as the correct answers) created by the teacher, structures the information in the XML format and inserts it into the database (items table and resprocessing table). (OBS not finished yet, see section 4.)
Show_items	Displays the items specified by servlet Choose_items. These are items that the test is to be consisted of. This servlet allows the teacher to specify where each part of the test starts and to give a name to the parts as well as the whole test.
Show_item_choice_preview	Displays <i>the choice item</i> that is to be entered into the test item bank. The teacher must also click the correct answers to the item here.
Show_item_preview	Displays <i>the fill- in-blank item</i> that is to be entered into the test item bank. The teacher must also write the correct answers to the item here-
Show_one_item	Display the item chosen by the teacher.
Show_test	Display the test chosen by the teacher.

The picture below shows the main servlets' functions and how they interact.



*Figure 1: The functions of the main servlets in the teacher client 2 and their interaction. The bold text at the top of the boxes describes functions in the system and the text inside the boxes are the names of servlets.*

## 4 Suggestions for continued development of the system

Here are some suggestions for the future development of the system:

- Teacher client 2

1. servlet `Create_test` → this servlet creates a test and puts all information in the table `test` in the database currently. It retrieves all appropriate questions from the database that are to be combined into a test.  
to do → all questions are retrieved as XML strings; write to a file and make an XML document that is to be the whole test file; this part should not be a problem;
2. servlet `Publish_test` → this servlet publishes a test instance and puts all information in the table `test_instance` in the database currently.  
to do → a) retrieve the test file and convert XML to HTML ; this task of conversion is currently done manually in the terminal:

```
java -Dcom.jclark.xml.sax.parser=com.jclark.xml.sax.Driver
com.jclark.xml.sax.Driver source_testfilename.xml asi_student.xml
```

with this command you are to generate multiple HTML pages ( test questions); it is not possible to call this command from a servlet; this multiple conversion has to be done in some other way; find out how!!☺ maybe you already know it! ☹

b) generate a `toc_file` automatically in a servlet; this is currently also done manually:

```
java -Dcom.jclark.xml.sax.parser=com.jclark.xml.sax.Driver
com.jclark.xml.sax.Driver source_testfilename.xml toc_student.xml
```

the same problem as above;

c) make a directory that has the same name as the test instance has in the table `test_instances` in the database (`instance_nbr`) automatically in a servlet; my suggestion: write a script that does this and then it is just to call the script from the servlet; it works fine;

d) put all those HTML pages (questions) and a `toc_file` in the directory described above;

3. servlet `Send_item_choice` → I had no enough time to finish this servlet because I could not access my servlets via web ( problems with a server again ☺);  
to do → it should not be a problem to finish it; the part that is not finished is marked like this ??????; it's just a parameter to get;
4. servlet `Gen_publish_test`  
to do → error handle if wrong date/time format

5. all servlets  
to do → check error handle in all servlets; I have tried not to forget to do this when developing servlets but I am not sure that I have done it in all of them;

6. develop a fill-in-blank question version2;  
In version 1 the fill-in-blank gap is at the end of the subquestion:  
Ex.  
Translate the word in the brackets into Swedish:  
My (mother) is cooking in the kitchen. \_\_\_\_\_

In version 2 the fill-in-blank gap is in the middle of the subquestion:  
Ex.  
Write the verbs in the brackets in the Simple Past Tense:  
He (does) his homework. He \_\_\_\_\_ his homework.  
to do → it should not be a problem to develop a fill-in-blank question version2; it's almost the same as the version1 with some tiny modifications;

- Student client

1. servlet Handle\_answers  
to do → when the student clicks to submit answers it often happens that it takes a lot of time for a system to process all the answers; the student is not aware of this; and what the student does very often is to click a few times on the button in order to submit the answers; it causes the system error because on each mouse click a transaction is stopped; to improve this function and prevent these problems you should implement a warning function - that the system is working and submitting the answers pops up when the student clicks the submit button; best done on the client-side, using JavaScript; no need for server to handle this.

- Administrator client

Develop an administrator client where teachers can register themselves and their courses as well as their students attending those courses.

## 5 System files

web address: stp.ling.uu.se/~sanja

log in as:

a teacher — username: lisa  
password: sumo  
a student — username: mats (or hanna, or jonna)  
password: sport

all servlets (.class files) --> /local/didax/servlets/\*.class

all java files --> /local/didax/src/\*.java

all tests --> /local/didax/servlets/tests

other files → /home/guest/sanja/public\_html/idex.html

/home/guest/sanja/public\_html/didax.js

/home/guest/sanja/public\_html/\*.gif

/local/didax/xml/didax\_dtds/\*.dtd

/local/didax/xml/didax\_xsl/asi\_student.xsl

/local/didax/xml/didax\_xsl/toc\_student.xsl

/local/didax/xml/qti\_dtd/IMS\_QTIv1p01.dtd

/local/didax/xml/test2/

dbname: swell



## Reports from Uppsala Learning Lab

---

- 1.2002 *A stitch in time: Enhancing university language education with web-based diagnostic testing*  
Lars Borin • Karine Åkerman Sarkisian • Camilla Bengtsson
- 2.2002 *DRHum in History — a status report*  
Esbjörn Larsson • György Nováky • John Rogers
- 3.2002 *Digital learning portfolios: inventory and proposal for Swedish teacher education*  
Jan Sjunnesson
- 4.2002 *Didax – a system for online testing: technical documentation*  
Sanja Babić • Camilla Bengtsson • Mattias Lingdell
-