ww-standalone-js

Contains work toward a javascript app that inserts a webwork problem into an html page.

The idea is to allow more flexibility in inserting iframe/webwork sections into an html page and to provide some protection for passwords which will be kept in localStorage and not in the html file.

Modulo a few minor tweaks in presentation this technique works with any webwork site using webwork 2.13 or later in other words any site updated within the last two years.

"summative" embedded problems.

- The short description of this scenario is that it is an alternative front end to a standard WeBWorK course. The problems are interspersed within the instruction text (PreTeXt, Libretexts, HTML page) instead of being presented within a separate WeBWorK homework application linked to at the end of the chapter. The results of the students' work is recorded in the WeBWorK course and can be inspected by the instructor.
    - Students can either do problem sets from within the instruction text or they can login to the WeBWorK course directly and do the same homework problems there.
    - There is a login javascript applet on the instruction text page which allows the student to enter the courseName and their userID and their password for the WeBWorK course. Specifying the courseName (possibly also the site name) allows the same instruction text to be used for different classes. This data is stored in localStorage on the student's computer and used to authenticate all of the problems in the current text. The information in the localStorage expires after ???? .
    - The communication with the webwork course is through the standard WeBWorK course interface, e.g. urls such as https://siteName/webwork2/courseName/homeworkSetName/problemNumber? followed by key/value pairs. The problem contents is returned within an iframe embedded in the instructional text.

  ### requirements

- Requires a **webwork site** and a **course** on the webwork site
- Students are registered in the course by an instructor and are assigned homework just as in a direct access WeBWorK course. They are given a unique userID for the course and a password. (there no option to change the password from the instruction text but they can change it by logging into the webwork

course directly).

- The course must be set up on the webwork site with each homework set constructed as usual with all of the problems appearing in the text and each student uses the text must be enrolled in the course and assigned the problems appearing in the text.
    - While the problems are organized into homework sets on the server each problem can be placed anywhere within the text. The server homework organization is useful for viewing the grades.
- The homework is identified by **courseName/homeworkSetName/problemNumber** (just as in a direct access WeBWorK course)
- Students
- To embed one of these homework problems in a text such as PreTexT or Libretexts or even just an HTML page the following information must be specified
    - The **courseName** and the user's **userID** and password must have been entered and stored in localStorage.
    - The **site**, **courseName**, **homeworkSetName**, and **problemNumber** must be specified in the text calling the homework problem.
    - The problem seed is NOT needed and is provided by the course on the server when the student is assigned the problem.
- Standard options: choose templateName=simple,
- Advantages and risks.
    - The student's userID and password do not appear in the text anywhere, however they are stored in localStorage on the student's computer. While this entry can be erased using "reset" or automatically disappear after a short time -- it is not that secure, particularly if used on public machines. (Research needed is session_key being used most of the time?)
    - Since this is really a standard WeBWorK course the opportunities to "cheat" are exactly the same as using a WeBWorK course directly. Manipulating the arguments embedded in the text simply mean that the grades for the problem will not be recorded in the right place.

"formative" embedded problems

- This uses an alternative "webservice" interface which exposes the capabilities of a WeBWorK course. It can be implemented in any webwork course but it could also be implemented in a minimalist manner using software under development.

- This version handles a student's submission anonymously. It grades the students submissions and reports the results to the student but does not record the transaction against the student's identity.
- The communication url is of the form https://mySite/webwork2/html2xml? followed by key/value pairs. The problem is returned within an iframe embedded in the instructional text.

requirements

- Requires a webwork site in which the webservice has been enabled in `webwork.apache2.conf` (this is the default).
- Requires a course: (e.g. `daemon_course`) with a designated special user (e.g. userID is `daemon`) whose permission level is at least 2 (proctorTA).
    - For security it is best if there are no other 'students' in the course and no homework sets created or assigned in the course. This insures that if something goes wrong with the daemon_course due to the relatively open exposure to the internet other students will not be affected.
    - As implemented on current WeBWorK installations only users with permission level 2 or more have access to the webservice -- ordinary student and guest accounts cannot be reached by the webservice.
- The homework is identified by it's **sourceFilePath** to the file in the OpenProblemLibrary, e.g. `sourceFilePath: Library/Union/setSeriesTaylor/ur_sr_9_6.pg` and by it's **seed** `problemSeed: 1234567`
- There are other options that can be specified such as the template producing the output format. ("outputformat" : "sticky")
    - The standard usage exposes problems available in the OpenProblemLibrary installed at the WeBWorK site.
    - By adding extra links to the daemon_course templates directory one can also expose problems available in **Contrib**, in **CAPA**, or in local libraries stored at the WeBWorK site.
    - It is also possible to create specific homework set collections of problems within the daemon course and use **sourceFilePath** to point to those problems. This is not a common use case in general but might be useful when creating an on the fly HTML page exposing recently created PG problems.
- The **sourceFilePath** and **problemSeed** are exposed in the text (PreTeXt file, HTML source, etc.) calling the WeBWorK site. It does not require much knowledge of the technology to change these -- which means a student might actually be doing a different version of the problem (change the problemSeed) or even a different problem altogether (manipulate the sourceFilePath). Since the useage is of these types of problems is for practice this does not seem to be deal breaker, and in fact could be used to advantage,

allowing a student to choose a different problemSeed in order to practice more than one version of the problem.

- There is nothing special about the daemon_course other than that course is not used for other purposes. If you have a userID and password in any webwork course with permission greater than 2 (e.g. you are a TA or an instructor) you can use the method above to access the resources of that course, including the link to the OPL. Using default permissions students and guests cannot access a course via the webservice -- it would give them unrestricted views of homework sets and problems not assigned to them.
- Research needed: How often is password being used in this version is session_key used from cookie?

The webservice API: html2xml

- defined in `renderViaXMLRPC.pm`
- url access: `https://mySite/webwork2/html2xml?`
- userID (often daemon)
- courseID (often daemon_course)
- displayMode (MathJax or images)
- course_password (for userID in this course -- often 'daemon')
- answersSubmitted (0 or 1)
- problemSeed
- problemUUID (aka 'problemIdentifierPrefix'; 'problemUUID' is preferred)
- sourceFilePath
- outputformat