System Design

Fundamentals

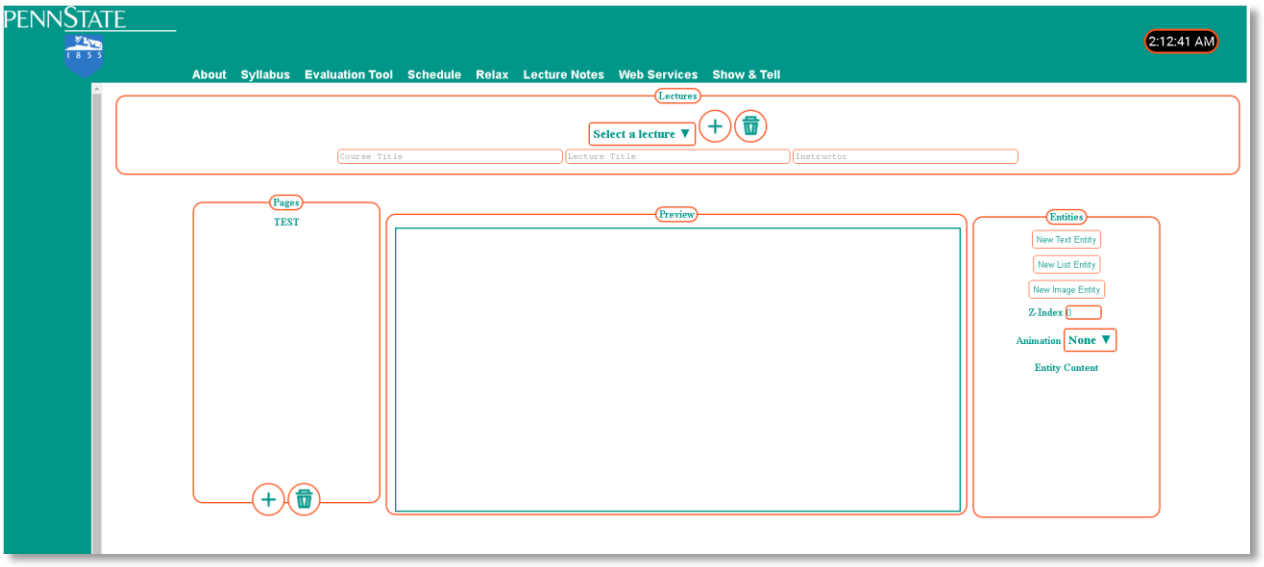
Our system follows closely to the representational state transfer (RESTful) architecture style. The system is split into three distinct modules – the Model, Controller, and View.

The Controller handles the connection between the client and the server, processing the requests and changes. The View includes the JSP generated HTML page that is sent back to the user for the user interface. The Model includes the Lecture, Page, and Entity Java objects.

In short - a lecture contains slides (pages), which contain moveable, editable entities.

Design

The slide editor appears when Show & Tell in the menu bar is clicked. The editor is split up into four different regions.



The upper region is the space to create a new lecture or pick and select old lectures to edit.

There is also a button to delete the selected lecture.

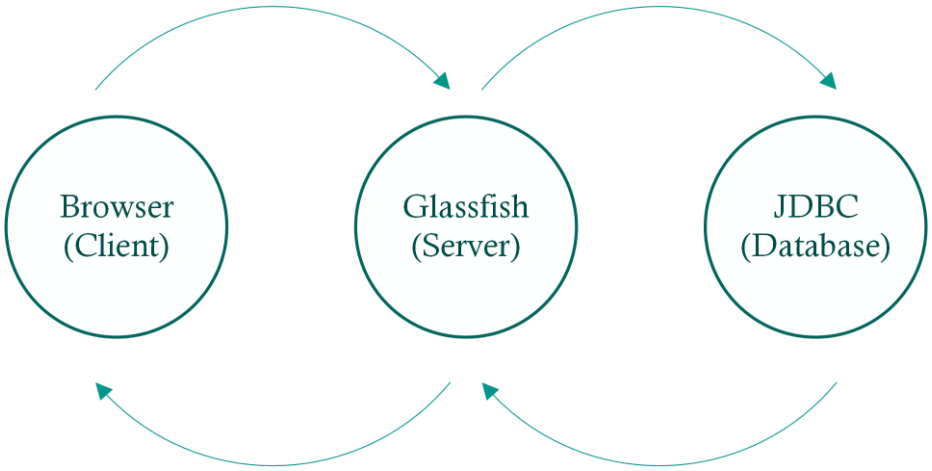
The leftmost region is the slide selector. It displays all of the slides that are in the selected lecture. A slide can be created, or deleted from the panel. When a slide is selected, it will display in the slide editor for viewing and editing.

The rightmost region is for entities. Entities can be created or deleted through this panel. When the entity is created, it is added to the middle region.

The middle region is the slide viewer and editor. Entities can be selected to be click and dragged to move around. This is where entities are edited and the slide is designed by the user.

Controller

The Controller handles the connections between the client, Glassfish webserver, and the SQL database. When a request is sent from the client to Glassfish, Glassfish uses the Controller to process the request to make changes to the database, and generate the response.



The possible post requests from the client to the server are:

getPages - Retrieve an array of slides (pages) for the selected lecture

getEntities - Retrieve an array of entities for the selected page

newLecture - Create a lecture object and insert into database

deleteLecture - Delete a lecture object from the database

updateLecture - Upload all of the information of the edited slide to the server

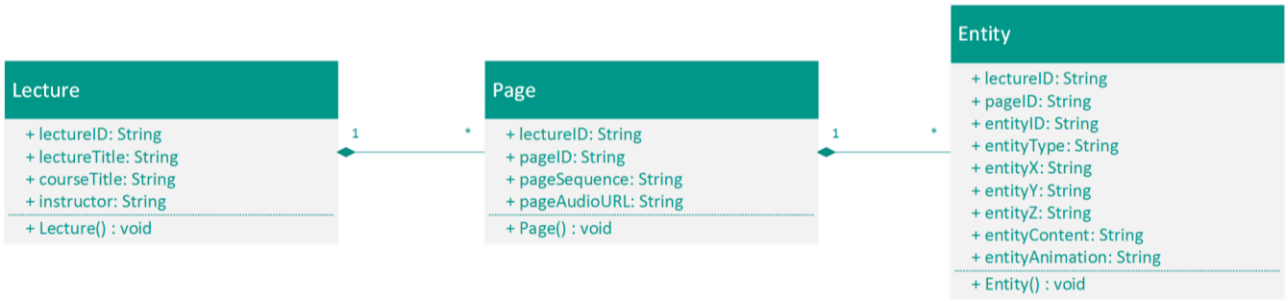
updateLecture is the core of the post requests as it processes all of the major changes to a slide. The information uploaded includes an array of the deleted entities, new entities, updated entities, deleted pages, new pages, updated pages, and the lecture. The server then handles the images, deletions, updates, and additions to the database.

The SQL database is implemented using Java Database Connectivity (JDBC). The database contains three tables – LECTURE, PAGE, and ENTITY. The tables are utilized in each post request which includes retrievals, insertions, and deletions.

Model

The Model module includes the three distinct objects – Lecture, Page, and Entity.

- A lecture is an object that holds the ID of the lecture, the title, course title, and instructor
- A page, or slide, contains the IDs of the parent lecture and page, (and sequence?)
- An entity is a moveable object within the page that contains the ID of the parent lecture and parent page, and information relating to the positioning and content of the entity



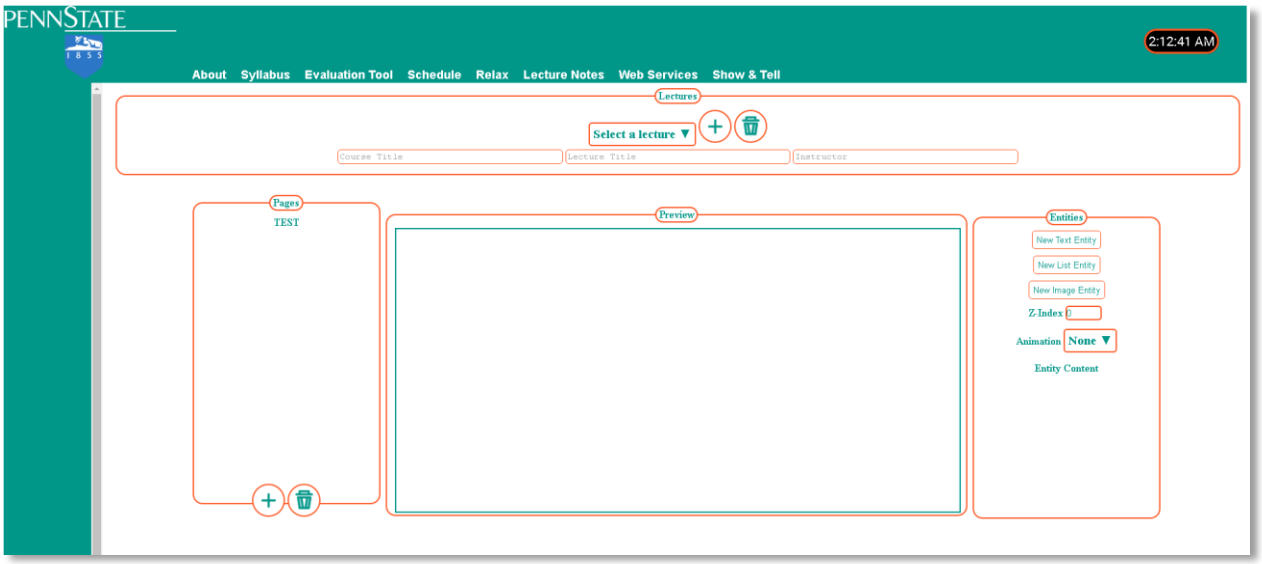
The Controller implements the Model to store the objects and information into the SQL database.

View

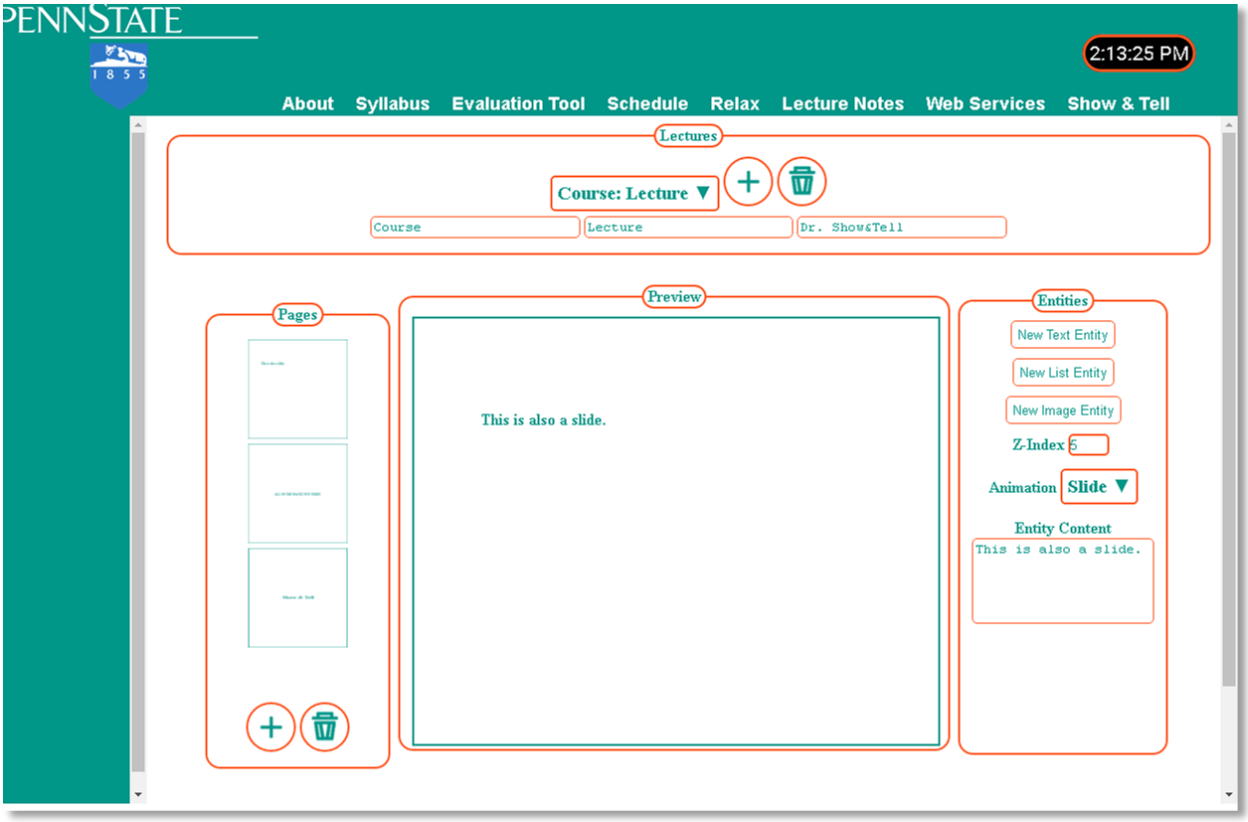
The View is the user interface and how the user edits the slides. It is an HTML file produced through JSP upon each post request. When the request has been processed or handled, the response is generated by the Controller on Glassfish and sent back to the client. It is thoroughly explained in System Execution.

System Execution

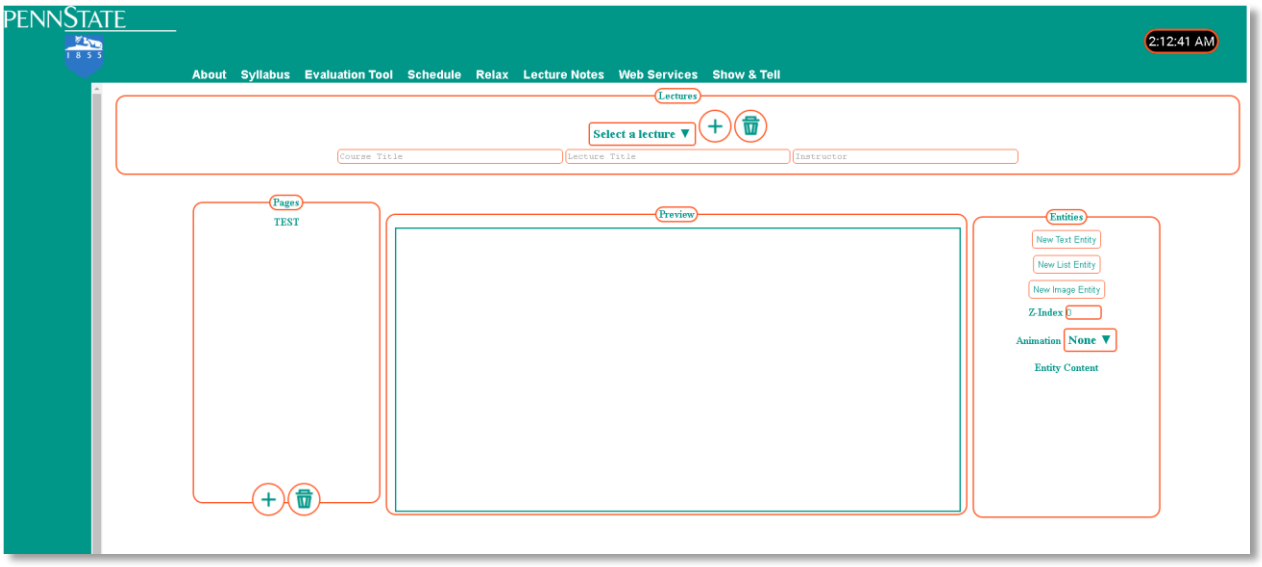
The slide editor is the first to appear when Show & Tell is selected. A user begins by adding a lecture. To do so, the three text fields Course Title, Lecture Title, and Instructor must be filled out.



Then… (Other demonstration pictures)



Explanations blahblah



Lessons Learned

Nick:

I have learned a lot about different subjects during the course of this class and final project. One example would be using GitHub which will be very beneficial to know for future team projects. While we had problems with it and almost lost work because of it, it is still very useful to have a way to see the changes that each teammate has made and are working on, and to have the ability to revert any unwanted change. I have actually never worked on a team programming project before and was interesting to learn how to do so and work as a group. Splitting up the tasks between the three of us was beneficial as we were each able to focus and understand certain aspects of the editor, and a lot more can get done earlier, faster, and easier. It also gives us the ability to learn from each other when help was needed.

While I have done HTML and CSS in the past, I have learned a lot more of it which will be useful in the future. I have previously used MySQL in the past for hosting Minecraft servers which used Java to interface with SQL, so it was useful to touch on SQL and learn the basics of using SQL databases in NetBeans.

JavaScript and JavaBeans as a whole were new concept. I have only used PHP in past web projects and JavaScript appears to be very useful for client side processing. I do intend on using JavaScript it in the future. For the server side processing, JavaBeans is something I do plan on using in the future as well, since it is very convenient to be able to use Java with web design since I am much better with Java than most other programming languages.

Caleb:

I learned quite a bit in this course considering I barely understood how the web functioned before this semester. The Client-Server paradigm of coding is different from what I'm used to, and this course provided a great opportunity to practice tackling a task from multiple perspectives.

I also learned a lot more about collaboration tools, particularly Git and GitHub. While I had used these tools previously, their importance was stressed quite a bit more this time around due to our team's tougher schedules.

While I had plenty of knowledge of databases prior to this class, I had little knowledge of how they were typically implemented or interfaced with common applications/websites. Getting the opportunity to utilize the knowledge I had with databases and how to query/organize information in them was good practice. It also turned out to be beneficial, as I seemed to have the most knowledge of databases in my group.

A big thing that I had never fully understood before this class was XML. I had never fully understood the purpose of it or what precisely it accomplished, but now I see its purpose and realize that the idea is extremely useful.

The biggest takeaway from this class was about the teamwork aspect of it. SWENG 411 was my first experience with large group projects, and, while it was good experience, it gave me a lot of misconceptions about team projects. For starters, our schedules this semester didn't sync up nearly as well as my previous group, and it lead to a lot of difficulties we simply didn't experience last semester. I also realized the importance of vocalizing ideas and ensuring everyone is on the same page. This course is very difficult to complete without a solid group, and the only way to keep a group working together well is to ensure everyone understands everybody's thought processes.

Matt:

The first major lesson I learned that will stay with me in this class was JavaScript. Although I already knew a little about how to use JavaScript from past projects, my past understanding was fairly limited and was mostly just due to the familiarity of the common syntactal structure. In the case of the project for which I used JavaScript, what I did was merely use it as a method of arbitrarily invoking predefined Java methods at runtime by binding them to the interpreter – in other words I used it like a shell script to perform a batch of commands. My experience with JavaScript in this class taught me the nuances, special

functionality, and the use, function, and interaction of closures and scope as opposed to a more rigid definition of objects and scopes. Initially I thought JavaScript was messy and tried to be too flexible, but as I used it more and became more familiar I've learned that the parts I thought to be unnecessary were actually routinely helpful and that keeping the code neat isn't much more difficult than most languages. Additionally after watching some group's presentations and seeing how they stuck to more of an object oriented scheme despite the fact that there are no classes in JavaScript made me realize that it still has potential for object oriented programming (in terms of code organization and separation of functional concepts – you still can't really take advantage of it as object oriented since there are no classes or inheritance or types).

The other big lesson I will take away from this course is the importance of making use of templates and pre-existing features instead of doing a custom implementation of everything. In contrast to working with languages like Java and C# that have built in support to make things like GUIs without relying on external libraries, making a website or even just a single good looking page using only HTML, JavaScript, and custom CSS takes a lot of work and effort with a lot of potential for things to go wrong.