FALL 2016

Web Programming Languages Project

Team tHE vOID sQUAD

Chintan Vachani

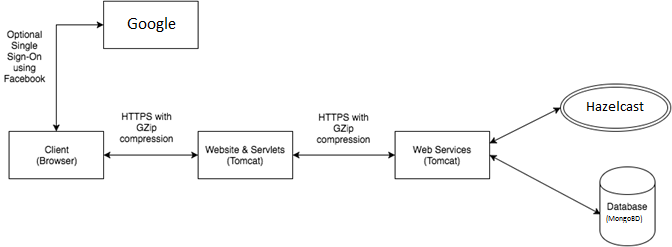
Gaurav Fogla

preethi

**INTRODUCTION:**

We have designed an online 3D print request system as a part of this project. This system enables the user to place orders for their 3D printable items online and helps the user to receive different bids/proposals for his/her requirement.

**ARCHITECTURE:**



* **Client:** The client makes a request from the browser to the website through HTTPS and the response is compressed using GZip, which is later uncompressed by the browser. The client can choose to create an account with our website or choose to sign in using Facebook and then use our website.
* **Website & Servlets:** The website and the servlets are hosted in a Tomcat Server running on port number 8080. The Tomcat server serves all the HTML, CSS and JS files needed on the front-end. It also receives the requests from the clients, validates the session information and then forwards the request to the Web Services. All the requests and responses are served through HTTPS and the responses are compressed using GZip.
* **Web Services:** The Web Services run on a different Tomcat instance and listen on port 9090. The Web Services access the Memcached pool of servers to check if the requested data is available. If there is a cache hit, then the data is served from the cache. Else, the Web Services contact the Database to fetch the requested data. The requests and responses are served through HTTPS and compressed using GZip. The Web Services serve only the requests from the Website’s Server running on the other Tomcat instance. All other requests are turned down by default.
* **Memcached:** The Memcached pool of servers are used to cache the data, so that it avoids the overhead of contacting the Database every time to fetch the data. They store the data in memory and are typically much faster than the Database. We have used a single Memcached server and is deployed on the same machine.
* **Database**: The Database is hosted on the Mongo DB server.

**TECHNOLOGIES:**

|  |  |
| --- | --- |
| Front-end:   * HTML * CSS * jQuery * JavaScript * Bootstrap | Back-end:   * Spring * MongoDB Database * Memcached * HTTPS * GZip Compression * Single Sign-On |

**FUNCTIONALITIES OF THE SYSTEM:**

* User Registration: User can register a new account using the login page and access his/her account. User has to enter his name, email-id and password to create an account.
* Existing user login and logout: Existing users can access their account via login page. They can access different pages and functionalities like bidding for posts, posting a request, accessing their user profile, view past orders and confirm their orders through the checkout option.
* User Login Information: User login information consisting of last date/time and the location of valid login.
* User profile information display and editing: User profile can be viewed and updated in Home page itself.
* Navigation Bar: Navigation bar consists of entries like Home page, My Posts, My Bids, My Orders, Shopping Cart and the logout button.
* Search Functionality: Search functionality can be accessed through the search option on the home page, where user can search either by Title or use any of the filters that are provided. The results are displayed in a sortable table.
* There are 4 filters provided from which the user can choose from. Multiple selections are also allowed.
* My Posts: We have created a page for posted requests where we are displaying all the posts that where posted by the user.
* My Bids: We have created a page for bids received where we are displaying all the bids that were made by the user.
* Shopping Cart: We have created a page for displaying all the accepted bids by the user. Here the user can add or remove the order, also can update the quantity of the accepted order. Finally it has a checkout button that sends emails to both the parties confirming the order.
* My Orders: We have created a page for the confirmed orders where we are displaying all the orders of the user.

**WEBSITE:**

The majority part of the website is made using the Bootstrap framework.

We start with a login page. After a user logs in, he has access to 5 different pages. These pages are linked on a navigation bar at the header. The pages are as follows:

* 1. **Home** 
     + This is the home page which has a navigation bar at the top with website’s name, 5 navigation links and a logout button. This navigation page remains same for all the following pages.
     + On the left, A section is implemented displaying the user information and two buttons for :
       1. Viewing the complete user profile.
       2. Editing the user profile.
     + In the middle section, we have a module that helps the user to request for an item or make a post. There are 4 categories from which the user has to select one from each.
     + On the right, a search field is implemented. It has 4 filters for the user to arrow down the search.
     + The center portion of the page displays all the posts that are made so far in a tabular format. The columns of the table are sortable by clicking the column headers.
  2. **My Posts.**
     + Page displays all the posts made by the user so far in a tabular format. The columns of the table are sortable by clicking the column headers.
     + The table has an additional column having a display bid button.
     + The button when clicked opens a modal displaying all the bids received for that particular post.
     + There is Add to Cart button with every bid, which adds that bid/order to the shopping cart of the user.
  3. **My Bids**
     + Page displays all the bids made by the user so far in a tabular format. The columns of the table are sortable by clicking the column headers.
     + There is an additional column having an info button. This button displays the information about that particular post in a popup modal.
  4. **Shopping Cart**
     + Page displays all the accepted bids by the user so far in a tabular format. The columns of the table are sortable by clicking the column headers.
     + Here the user can specify the quantity of the requested item and then confirm the order by pressing the checkout button.
  5. **My Orders**
     + This page keeps a record for all the confirmed orders i.e. those which are checked out by the user in a sortable tabular format.

**WEB SERVICES:**

The Web Services are hosted on the Tomcat instance running on port number 9090. All the Web Services are REST based.

* **Frameworks used:**

1. Spring boot
2. Springframework: To log the messages.
3. Spring ODM: ODM framework to handle the interaction between the Database and the Web Services.

* **Libraries used:**
  1. Springframework: To support GZip Compression.
  2. Memcached Java Client: To communicate with Memcached
* **List of Web Services:**
  1. UserController:
     + It takes the email ID and password and verifies against the details persisted in the database. If the credentials are valid, then it returns a userId, which is set as an attribute in the session.
     + It also supports Single Sign-On using Google. If the user is authenticated using Google, then a profile information is created for the first time, from details obtained from Google.
     + Takes the user details and creates a new user or updates the user profile details.

* 1. PostController:
     + Takes the data passed from the home page, which was forwarded by the Email Servlet and sends an email to the user.
     + Receives the details from the OrderService and sends an email to the user with the order information.
  2. OrderService:
     + Receives the order request from the Servlets and places the order. The order details are persisted in the database. After the order is successfully placed, it calls the EmailService to send an email to the customer’s registered email ID.
     + It also fetches the following details from the Database, if the requested data is not present in the cache. If it is present in the cache, then it gets the data from the cache.
       1. Recent orders
       2. Order details for a given orderId
       3. List of orders based on the constraint on order amount.
  3. ReviewsService:
     + Processes the add review request from the Review Servlet and stores it in the database.
     + It also fetches the reviews from the Database, if the data is not present in the cache, based on item ID or recent reviews.

**SINGLE SIGN-ON:**

The user is given the flexibility to use his/her Google account to use our service. A Google login button is added in the user login page. On clicking this button, a pop-up opens asking the user to login to Google. We request the user’s name and email address from Google during the login process. Once the user is successfully logged in, we obtain the name and email address of the user and map it the corresponding user in our system, if the user has already used our system. If no such mapping is found, then a new user profile is created in our Database with this mapping.

**PROBLEMS ENCOUNTERED:**

* We had designed the Database schema in such a way, that the identifiers are generated using Database Sequences to ensure consistency in identifier generation. We have used Hibernate ORM Framework in the Web Services to interact with the Database. Hibernate by default uses it’s default Sequence generator along with the value fetched from the Database to generate the identifier. Hibernate does this to avoid having to rely on the Database to generate sequence value every time. It occasionally queries the Database for the sequence value, caches it and uses its logic to generate new values later. This generation logic created inconsistencies and constraint violations with existing test data in the database even though the range of the Database sequence was defined to be far from the range of values used in the test data. To resolve this issue, we defined the allocation size attribute in the SequenceGenerator hibernate annotation, which forces Hibernate to rely completely on the database to generate sequence values.

**CONCLUSION:**

Through the implementation of this project, we have got a good understanding of the real world use of the web technologies and languages discussed in the class over the entire semester. Incorporating technologies like HTTPS, Memcached and Frameworks like Hibernate ORM helped us to realize the actual need and purpose of these in real world projects.