|  |
| --- |
| University of Texas at Dallas |
| Reader’s Delight- An online bookstore |
| Project Report |
| Authors  Sharannya Ranjith [sxr154730]  Nipun Chawla [nxc141130]  Lahari Ganesha [lxg150730] |

|  |
| --- |
| A Web Application Project |

**Web Architecture**

Reader’s Delight is scalable, programming-language/platform independent application implemented out of loosly coupled modules that are self contained and implemented via RESTful methodologies.

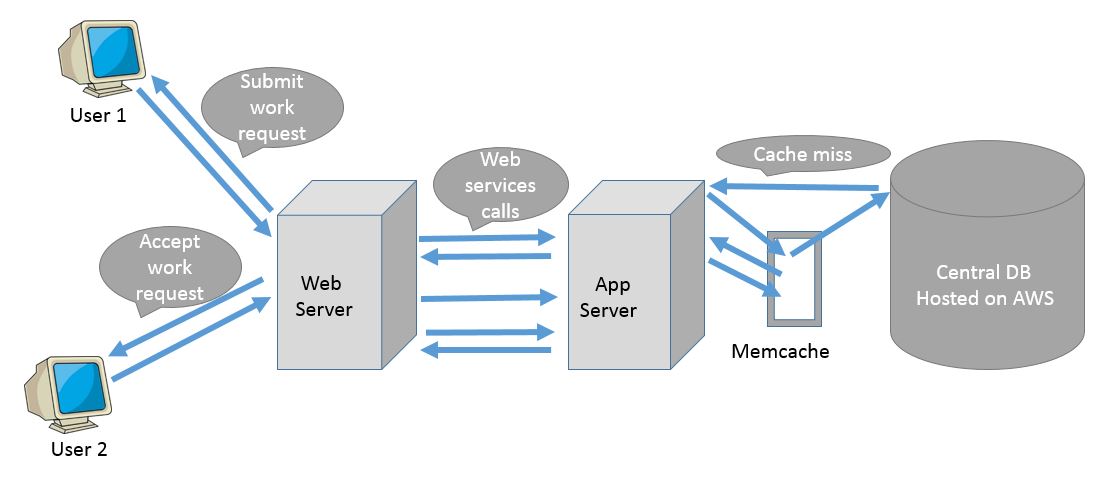


Fig. 1 Web Architecture

Here, the user visits Reader’s Delight and logs in. When the user logs in, his credentials are passed along with the request method and these credentials are verified by the web server. The web server looks for the user’s username and password in the cache. If this information is unavailable in the cache, this corresponds to a cache miss and the web server looks for the information in the database. When a user chooses to search for a particular book, the book information like title, author or genre is passed along with the request method, the web server looks for these books in the the database. The book information is then formulated to a response object and it is sent back to the user by the web server as a response method.

*Client Browser*

The browser we are considering here is Mozilla Firefox 38.0. This browser is being run on a 64 bit machine.

*Web Server*

We choose to use Apache Tomcat 8.0.

The web server hosts website and make calls to the Rest Service to obtain data.

•The interaction between the web server and Rest Service; and browser and web server is secured using https protocol.

•The web server is developed using JSP/Servlets and deployed on Tomcat Server.

•The interaction between webserver and Rest Service Server is secured using secret key.

*Rest Service*

•The Rest Service is the one which interacts with the database and provides data to the web server.

•The Rest Service first checks in the cache and then if there is cache miss interacts with database for getting the data.

•The Rest Service are developed using Jersey Framework and deployed on Tomcat server.

*Cache*

This stores the frequently used data and makes data available faster to the rest service preventing expensive database calls.

•Whenever there is cache miss, data will be populated from database.

•Whenever there is cache hit, data will be sent from cache.

*Database*

We have chosen MySQL Workbench 6.3

The database will be called in case of cache miss.

•Whenever there is cache miss, it will first save data in cache and then return data to the rest Service.

**Technologies Used**

*Front End:* HTML5, CSS and jQuery

Other Technologies considered: As HTML5 is most comfortable, other technologies need not be considered.

Reason:

The UI of the application was developed using HTML5, CSS and jQuery

A sample template was used to build the application.

*Database*: MySQL 6.3 Workbench

Other Options: Orcale, MongoDB

Reason:

•MySQL is relatively light-weight, can be extremely fast when applications leverage architecture.

•Lots of features stay free as the database servers grow such as replication and partitioning.

•MySQL excels when high speed reads can be used for web, gaming and small/medium data warehouses and OLTP systems.

*Web Server*: Tomcat Apache 8.0

Other Options: Apache Http Server.

Tomcat was preferred because of the following reasons:

•Provides the Java Servlet and JSP support for dynamically served pages

•Works as a light-weight testing server

•Can be run in different modes to promote better performance

*Rest Service Framework*: JAX-RS Jersey Rest framework.

Other Options: Spring MVC

JAX-RS was considered because

•JAX-RS has the advantage of creating APIs that are simpler to create and digest messages for in different browsers and mobile devices.

•The Rest Services are deployed on Apache Tomcat server.

•JAX-RS targets the development of web services (as opposed to HTML web applications) while Spring MVC has its roots in web application development.

*Cache*: MemCache Distributed Cache.

Other Options: Redis.

•Memcached is a good option for implementing a distributed caching mechanism.

•It stores data in the format of key value pair. Currently, only single node is implemented.

*Development Environment*: JDK 1.8(JSP/Servlet Framework)

Other Options: Javascript, Python

•Java is high performance compiled languages, with great support from a well known vendors, and entire ecosystem of companies that sell everything - from IDEs to libraries to automation tools.

*Compression*: GZip compression

**Functionalities Implemented**

1. *New User registration*:

A user, who is visiting Reader’s Delight for the first time, can sign up using his email and by creating his password. This allows a user to store his information like card and the books which he might want to buy later. This saves the user from searching for a particular book every time he wants to buy it. Also, this saves the user from entering his card details every time he wishes to buy a book.

2. *Existing User Login and Logout*:

A returning user can login using his registered email address and password. Now, a user will be able to view the books that he added to his cart and proceed with the payment or he can choose to search for a few more books and add them to his cart or just logout from the application. When a user is logged in, he has access to all his information like card details and he can choose to edit them anytime.

3. *User Login Information*

i. Last date/time of valid Login:

For security purposes, after a user logs in, he is displayed with the time he logged in previously. This allows a user to keep track of his online activity. If an unauthorized user has logged in, the user can get to know about this unauthorized login as the last login time would be different from his actual last login time.

This is done by storing the login time information of a user to the database and every time a user logs in, before updating the login time, the previous login time is fetched and displayed.

ii. Number of failed login:

When a user enters wrong password, it leads to a failed login. The number of such failed logins is counted. This helps in prohibiting a user from logging in even if he enters correct credentials after a specified number of logins, because there is a high probability that the user has used brute force method to arrive at the correct credentials.

4. *User Information Profile display and editing*:

As described in point 1, a logged in user has access to his profile information and card details. At any point of time he can choose to update these details. This lets the user to keep track of the books he added to his cart and saves the user from entering his card details every time he makes a purchase.

5. *Image slider*:

If the user is interested in a particular genre of books, he can choose to search for the books based on genre. Alternatively, he can choose to just click on the image slider on the image describing the genre of interest. This takes him to the books belonging to the genre of his interest.

6. *Navigation bar*:

The user has variety of options to do on Reader’s delight. He can just view *home* or go for user *login*, or *search* for a few books without logging in or choose to login as an *admin* or visit the *contact us* page to contact Reader’s delight administrators. Instead of searching for these functionalities, there is a navigation bar using which the user can go to his page of interest.

7. *Search functionality:*

The user might be interested in a particular book or a book written by a particular author or books belonging to a particular genre. Instead of going through all the listed books and then selecting the book of interest the user can go ahead and use the search functionality implemented in the application. Using this functionality, the user can search for books based on title of the book or author of the book or genre of interest.

Also, to make it easy to look up for a book in the search result, the books in the search result are sorted in alphabetic order based on title.

8. *Shopping cart and order purchase submission*:

The user might want to buy more than one book for a giver order. In that case, instead of buying each book individually the user can add all the books of interest to a cart and pay once. The user can also choose to add the book to the cart now and choose to pay later. Once the user pays for the entire order the user gets a confirmation about his payment and purchase submission.

i. Ability to add books

The user can add books to the cart. After he adds a book to the cart he can either go ahead and checkout or go back to search to add few more books to his cart.

ii. Ability to remove books from his cart

If a user adds a book to his cart by mistake or feels that the decision of buying the particular book is not correct, he can always go ahead and choose to buy the books of his choice in his cart.

iii. Ability to update book counts

If a user has entered wrong value for quantity for a particular book, or feels that he needs less/more number of copies than what is entered, he can always go ahead and edit the count value for the particular book. And he’ll be charged for the total number books after his latest modification.

iv. Confirmation email upon successful order submission

After a user checks out, an email is sent to the user containing confirmation message about his orders and order total. This enables a user to keep track of what he bought and he will also have a permanent copy of his orders.

9*. Page listing all the previous orders*:

The user can keep track of all the purchased items by clicking on orders. It lists all the books that the user ordered and the amount he paid for it. This also enables the user to compare between the price he paid and the current price of the book.

10. *Ability to add and view review to purchased books:*

The user can add or view a book that he bought based on the standard of the book and service of Reader’s Delight.

11*. Error page if any unavailable page is accessed*:

If a user has accessibility to any page that is unavailable, a custom page with 404 error is displayed. This conveys the user that the requested page does not exist.

12. *Contact Us page*:

This page gives the details of the administrator’s contacts. If the user has any issue regarding purchase or unavailability of any book, the user can contact here.

13. *Submission of request from Contact US page:*

If a user submits a request from the contact us page, the user receives a mail saying that his request has been successfully submitted to Reader’s delight.

**Extra Functionalities Implemented**

1. *Admin Login:*

An Administrator can login and he has authorization to add a user, delete a user and view the orders of all users.

2. *Captcha Validation*:

To distinguish between user and computer captcha validation is also implemented.

3. *Update user profile picture*:

The user can upload his profile picture and update it anytime.

4. *Forgot Password*:

If a user forgets his password, he can click on forgot password link and a mail is sent to the user with the new password.

**Web Services**

To implement Reader’s delight, we have implemented two servers; one web server processes the requests requiring business logic and other non-database-requirement requests. The other server takes care of all the requests requiring database access. Both the servers we are using here are Tomcat Apache 8.0.

The following web services have been implemented:

1. *Register*:

The user is prompted to fill out a form to register onto the application. The user details such as name, address, phone number is asked for.

1. *Login*:

The user enters the user name and password. This information is authenticated with the database and if a matching record is found, the user’s first name is fetched.

1. *Number of failed log ins*:

The table InsertUserInfo stores the number of failed logins for a user and this is displayed and incremented every time a user fails to login with correct password.

1. *Last date/time of valid login*:

Whenever a user logs in, the date and time is recorded in the lastlogin field of InsertUserInfo table. Once a user login, this field is fetched and displayed and then it is updated to the current time.

1. *Admin Login*:

Administrator can login and he will have authorization to modify InsertUserInfo table. He can add a user, delete a user and view the orders of all users.

1. *Fetch User Profile and Store back the updated version*:

Whenever a user wants to edit his profile, the user profile information is fetched from the database and the user is allowed to edit the information and the information is stored back to the database.

1. *Search functionality*:

Here the user’s keywords are matched with the respective column (title or/and author or/and genre) in the database and all the matching tuples are fetched. Using this functionality, the user can search for books based on title of the book or author of the book or genre of interest. Also, to make it easy to look up for a book in the search result, the books in the search result are sorted in alphabetic order based on title.

1. *Shopping cart*:

Once a user adds a particular item to his cart, this item is added to the cart table in the database for that particular user id. This saves the cart permanently in the database. This allows the user to buy more than one book for a giver order. In that case, instead of buying each book individually the user can add all the books of interest to a cart and pay once. The user can also choose to add the book to the cart now and choose to pay later. Once the user pays for the entire order the user gets a confirmation about his payment and purchase submission.

1. *Page listing all the previous orders*:

Once the user checks out, the “bought” field in the cart table is made “yes”. This tells that the user has bought the item and the user can keep track of all the purchased items by clicking on orders. It lists all the books that the user ordered and the amount he paid for it (fetches all the tuples in cart for that particular user id where bought=”yes”).

1. *Ability to add and view review to purchased books*:

The user can review a book that he bought based on the standard of the book and service of Reader’s Delight. The user can also view the reviews provided by other users. This can help the user in using the book and other directions.

**Problems Faced**

1. *Send Email*:

The email server configuration was not working correctly. The solution was to configure Gmail smtp server for sending email. Also we had to disable the security feature in our Gmail account which allowed us to send a mail from an external application.

2. *Servlet Redirection*:

The redirection from servlet to JSP was giving null pointer exception because of data in JSP being null. The solution was to set beans correctly in request object.

3. *Add to Cart*:

We had form with two submits one for confirming the order and other for deleting an entry in the cart, so submit was not working as expected. The solution was to use attribute onclick to call form.action method.