

## Project 1

Name: Ariane Correa  
Andrew ID: ajcorrea

### Task 1

Screenshots:

1. Main Page (Before we enter string value)

The screenshot shows a web browser window with the URL `localhost:8080/Project1Task1-1.0-SNAPSHOT/` in the address bar. The page title is "Compute Hash Value". There is a form with the following fields:

- Type String value:** A text input field containing the placeholder "Enter a string value".
- Hash Type:** Radio buttons for "MD5" (selected) and "SHA-256".
- Submit:** A blue button.

Below the form, there are three empty text input fields labeled "Input Value:", "Base64 Encoded Value:", and "Hexadecimal Encoded Value:".

## 2. Results for MD5:

(We enter Type String Value= Ariane, select Hash Type=MD5 and click Submit)

localhost:8080/Project1Task1-1.0-SNAPSHOT/compute-hashes?inputValue=ariane&hashType=MD5

### Compute Hash Value

Type String value:  
Enter a string value

Hash Type:  
 MD5  SHA-256

Submit

Input Value: ariane

Base64 Encoded Value: 2NewlEzyuIM2ya/khzKZOQ==

Hexadecimal Encoded Value: D8D7B0944CF2B88336C9AFE487329939

3. Results for SHA-256:

4. (We enter Type String Value= Ariane, select Hash Type=SHA-256 and click Submit)

localhost:8080/Project1Task1-1.0-SNAPSHOT/compute-hashes?inputValue=ariane&hashType=SHA-256

## Compute Hash Value

Type String value:  
Enter a string value

Hash Type:  
 MD5  SHA-256

Submit

Input Value: ariane

Base64 Encoded Value: MCYC/FrMtUPKsOL48+WLDXR/5HZ0aDGzKaoosXwxSLA=

Hexadecimal Encoded Value: 302602FC5ACCB543CAB0E2F8F3E58B0D747FE476746831B329AA28B17C3148B0

## Code Snippets:

Code from *ComputeHashes.java*: This `doGet` method in a servlet handles HTTP GET requests. It first retrieves the user's input, including the hash type and a string value, from the request parameters. Then, it calculates the hash value of the input string using the specified hash algorithm (e.g., MD5 or SHA-256) and encodes this hash value as both base64 and hexadecimal strings. These encoded values, along with the original input string, are set as attributes to be displayed in the JSP page. Finally, the method forwards the request to the "index.jsp" page for rendering, where the calculated hash values and original input are displayed to the user.

```
18 @Override
19 public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
20     response.setContentType("text/html");
21
22     // Get the hash type and input value from the request parameters
23     String hashType = request.getParameter("hashType");
24     String inputValue = request.getParameter("inputValue");
25
26
27     MessageDigest md = null;
28     try {
29         md = MessageDigest.getInstance(hashType);
30     } catch (NoSuchAlgorithmException e) {
31         throw new RuntimeException(e);
32     }
33
34     // Calculate the hash value
35     byte[] hashValue = md.digest(inputValue.getBytes());
36
37     // Encode the hash value as base64 and hexadecimal strings
38     String base64EncodedValue = DatatypeConverter.printBase64Binary(hashValue);
39     String hexEncodedValue = DatatypeConverter.printHexBinary(hashValue);
40
41     // Set attributes to return to the JSP
42     request.setAttribute("base64EncodedValue", base64EncodedValue);
43     request.setAttribute("hexEncodedValue", hexEncodedValue);
44     request.setAttribute("inputValue", inputValue);
45
46     // Forward the request to the JSP for rendering
47     request.getRequestDispatcher("index.jsp").forward(request, response);
48 }
49 }
```

## Project 1

Name: Ariane Correa  
Andrew ID: ajcorrea

### Task 2

Link to Main Page: <http://localhost:8080/Project1Task2-1.0-SNAPSHOT/>

Link to Result Page: <http://localhost:8080/Project1Task2-1.0-SNAPSHOT/getResults>

Screenshots: Desktop View

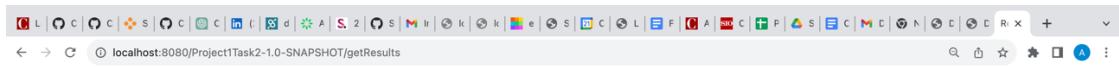
#### 1. Application with the Answer Options (Desktop View)

The screenshot shows a desktop browser window with the URL [localhost:8080/Project1Task2-1.0-SNAPSHOT/](http://localhost:8080/Project1Task2-1.0-SNAPSHOT/). The title bar reads "Distributed Systems Class Clicker". The main content area contains a question and four answer options:

Submit your answer to the current question:

A  
 B  
 C  
 D

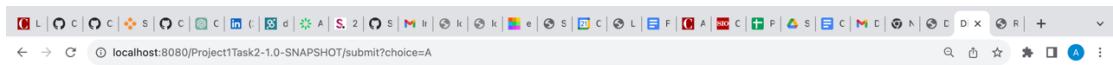
## 2. Application Result Page (Desktop View)



### Distributed Systems Class Clicker

There are currently no results

## 3. Application when we select a particular option (Here, we select A) and click Submit

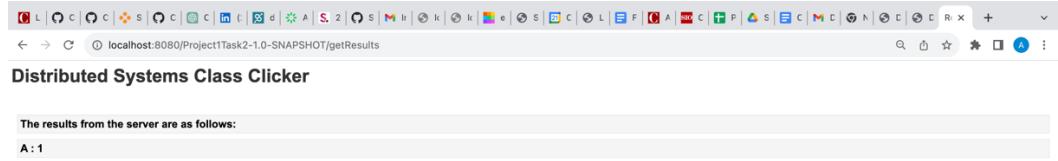


### Distributed Systems Class Clicker

Submit your answer to the current question:

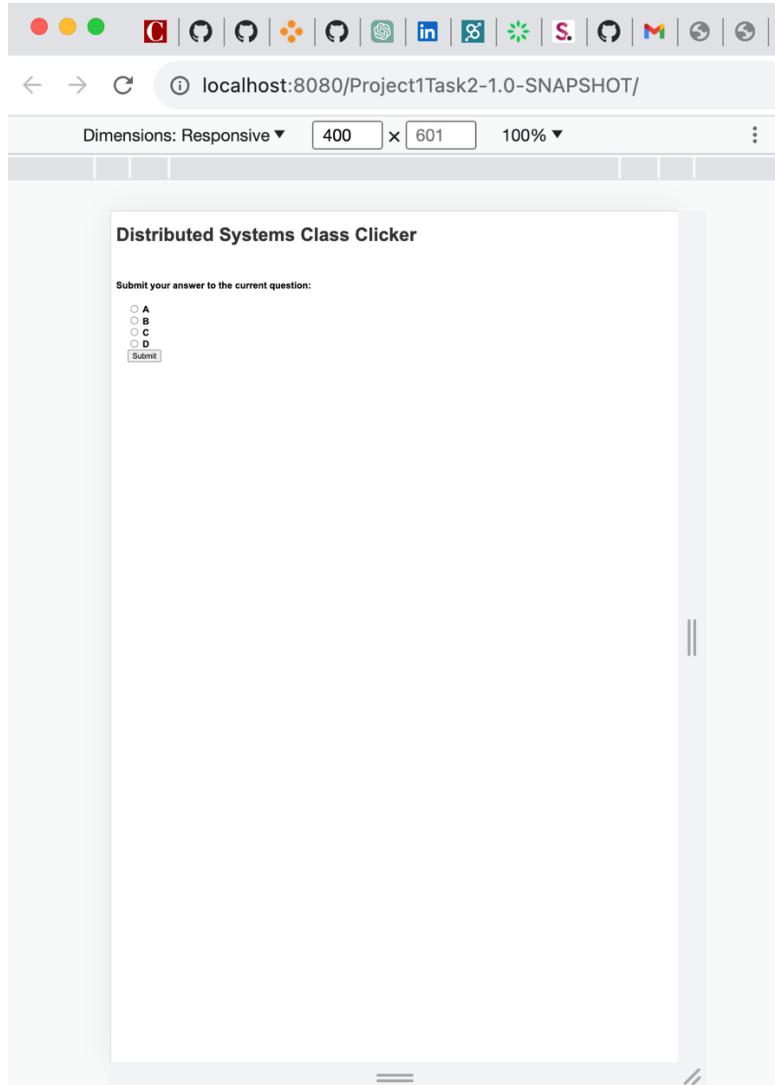
- A
- B
- C
- D

#### 4. Updated Application Result Page (Desktop View)

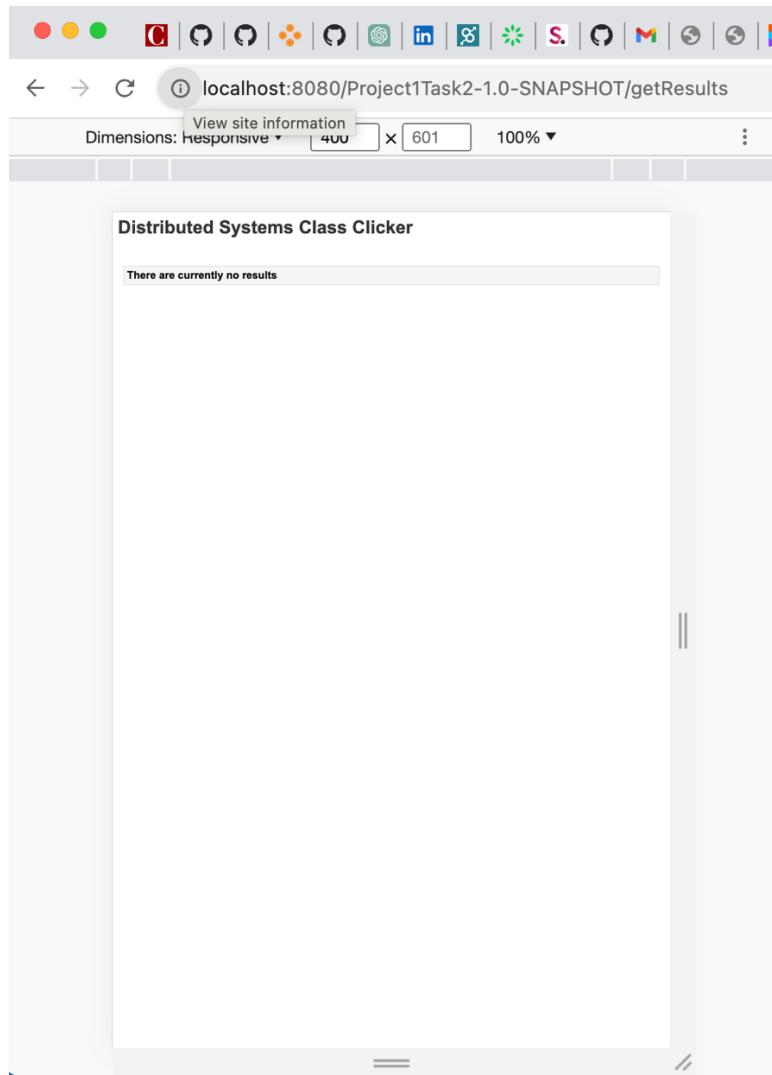


## Screenshots: Mobile View

### 1. Application with the Answer Options (Mobile View)



## 2. Application Result Page (Mobile View)

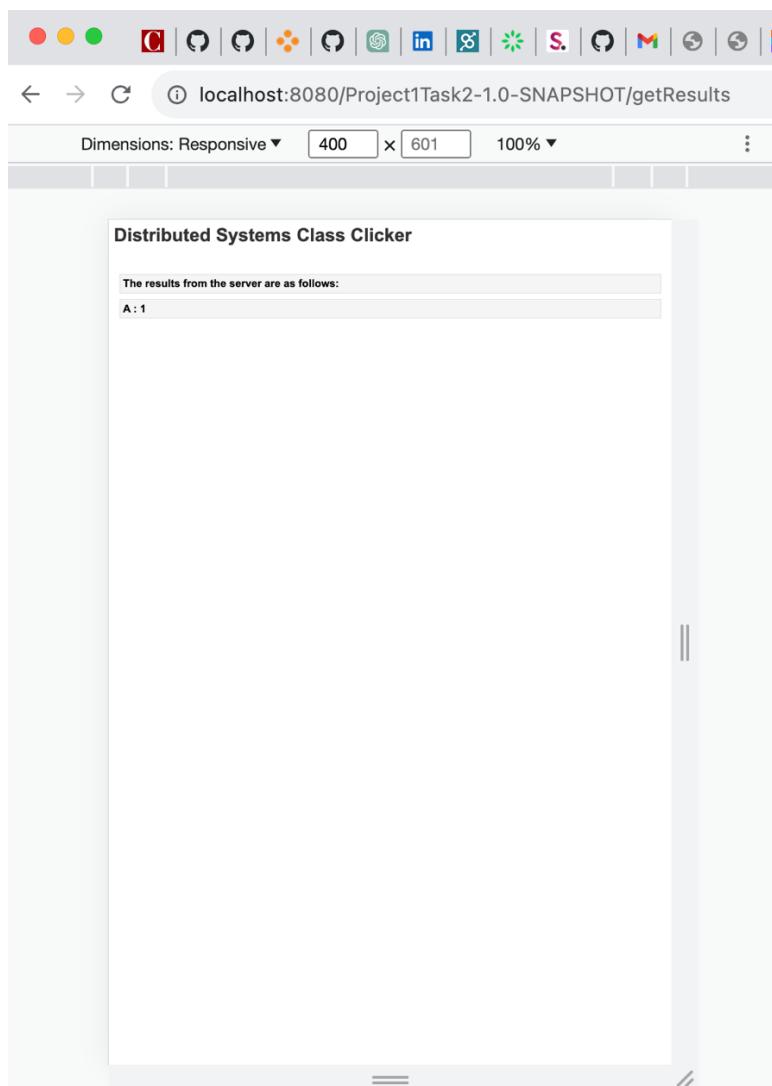


3. Application when we select a particular option (Here, we select A) and click Submit

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:8080/Project1Task2-1.0-SNAPSHOT/submit?choice=
- Toolbar:** Includes icons for back, forward, search, and other browser functions.
- Dimensions:** Responsive, 400 x 601, 100%.
- Content Area:**
  - Title:** Distributed Systems Class Clicker
  - Text:** Submit your answer to the current question:
  - Form:** Radio button group for choice A (selected), B, C, or D. The "Submit" button is visible below the radio buttons.

#### 4. Updated Application Result Page (Mobile View)



## Code Snippets:

Code from *ClickerScorer.java*: Here, we create and maintain a TreeMap that stores values entered by the User in ascending order.

```
 9  public class ClickerScorer {
10
11      // A TreeMap is used to maintain key values in ascending order
12      2 usages
13      private Map<String, Integer> score = new TreeMap<>();
14
15  >      2 usages
16  >      public Map<String, Integer> getScore() { return score; }
17
18  >      3 usages
19  >      public void setScore(Map<String, Integer> score) { this.score = score; }
20
21  }
22
23
24
```

Code from *ClickerMain.java*: Here, we make necessary initializations of the new object. The String outputText is initialized since we need to send it to the .jsp files for display on the application.

```
16  @WebServlet(name = "clicker-controls", urlPatterns = {"/submit", "/getResults"})
17  public class ClickerMain extends HttpServlet {
18
19      3 usages
20      private String outputText;
21      5 usages
22      private ClickerScorer clickerScorer = new ClickerScorer();
23
24  @Override
25      public void init() {
26          outputText = "";
27          clickerScorer.setScore(new HashMap<>());
28      }
29
30
```

Code from *ClickerMain.java*: The logic for producing the results.

- Handling the "/submit" Path: When the servlet receives a request with the "/submit" path, it processes the user's choice and updates or adds it to the maintained map. Additionally, it sends the "outputText" attribute with the User-selected answer back to the JSP file for display.
- Handling the "/getResults" Path: In case the servlet receives a request with the "/getResults" path, it prepares the map object containing the selected options for display. It then forwards the request to a new results JSP page, which

showcases the various selected options. Importantly, this step also clears the map to ensure it maintains new scores for subsequent requests.

```
27 @@@ public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {  
28     response.setContentType("text/html");  
29  
30     // If the path is "/submit"  
31     if (request.getServletPath().equals("/submit")) {  
32         String choice = request.getParameter("choice");  
33  
34         // Create an output message for the user  
35         outputText = "Your " + choice + " has been registered";  
36         Map<String, Integer> score = clickerScorer.getScore();  
37  
38         // Get the value of the "choice" parameter and update the map accordingly.  
39         // If the value is already contained in the map, update its count; otherwise, add a new entry.  
40         score.put(choice, score.containsKey(choice) ? score.get(choice) + 1 : 1);  
41  
42         clickerScorer.setScore(score);  
43         request.setAttribute("outputText", outputText);  
44         request.getRequestDispatcher("index.jsp").forward(request, response);  
45     }  
46  
47     // If the path is "/getResults"  
48     if (request.getServletPath().equals("/getResults")) {  
49  
50         //Set map attribute as object and retrieve in the servlet page  
51         request.setAttribute("choiceMap", clickerScorer.getScore());  
52         request.getRequestDispatcher("result.jsp").forward(request, response);  
53         clickerScorer.setScore(new HashMap<>());  
54     }  
55 }  
56 }
```

Code from *result.jsp*: Code that handles Display in the “/getResults” page. If the map is empty, the first message is displayed. Otherwise, the JSP page iterates through the map and displays all key-value pairs. Since the Map object used is a TreeMap, the values are automatically presented in ascending order.

```
24 </head>  
25 <body>  
26 <h1>Distributed Systems Class Clicker</h1>  
27 <%  
28     Map<String, Integer> choiceMap = (HashMap<String, Integer>) request.getAttribute("choiceMap");  
29     if (choiceMap.isEmpty()) {  
30         out.print("<br/>");  
31         out.print("<h3> There are currently no results </h3>");  
32     } else {  
33         out.print("<br/>");  
34         out.print("<h3> The results from the server are as follows: </h3>");  
35         for (Map.Entry<String, Integer> entry : choiceMap.entrySet()) {  
36             out.print("<h3>" + entry.getKey() + " : " + entry.getValue() + "</h3>");  
37         }  
38     }  
39 %>  
40 </body>  
41 </html>
```

## Project 1

Name: Ariane Correa

Andrew ID: ajcorrea

### Task 3

URLS Scraped:

website 1: for list of MLB team names:

<https://www.espn.com/mlb/teams>

website 2: for MLB team logos:

[https://www.sportslogos.net/teams/list\\_by\\_year/42023/2023\\_MLB\\_Logos/](https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/)

website 3: for mlb team home stadium info:

<https://geojango.com/pages/list-of-mlb-teams>

api 1: for MLB team information:

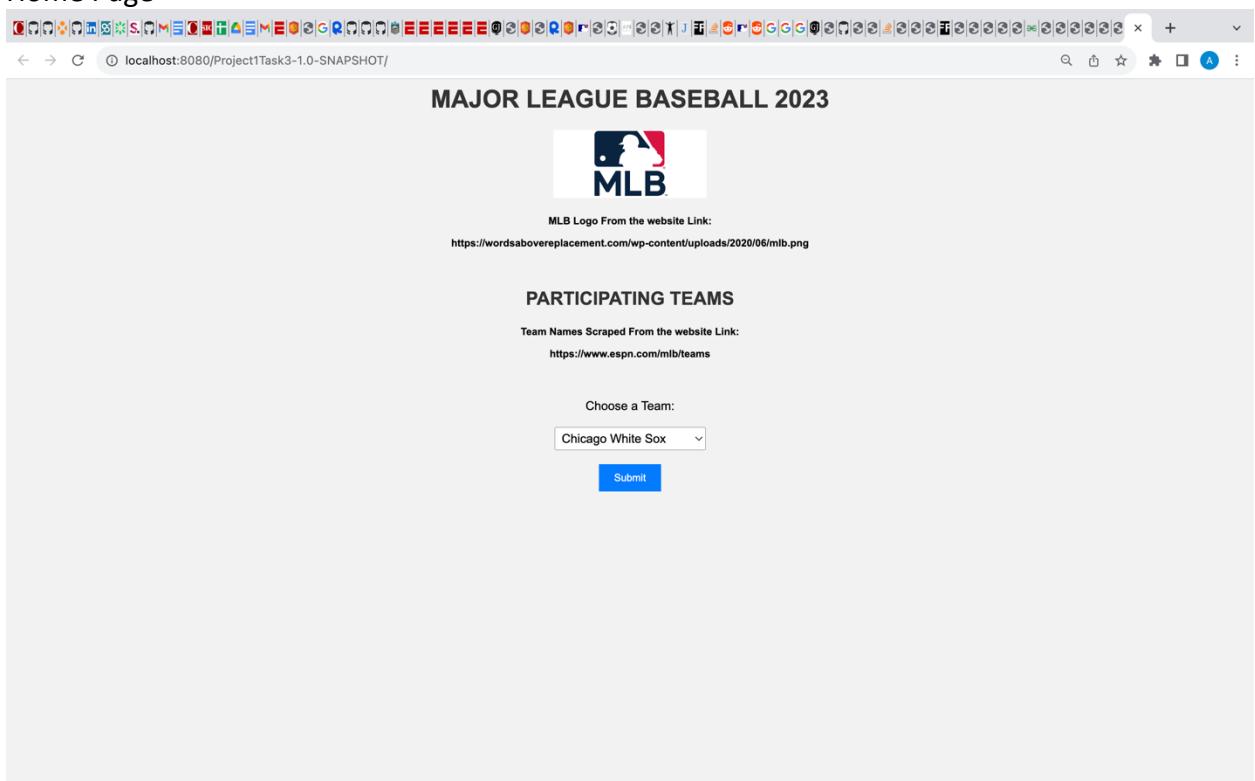
<https://statsapi.mlb.com/api/v1/teams>

api 2: for mlb team top hitter stats:

<https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting>

Screenshots:

1. Home Page



2. Home Page with Dropdown menu to select Team

The screenshot shows a web browser window with the URL [localhost:8080/Project1Task3-1.0-SNAPSHOT/](http://localhost:8080/Project1Task3-1.0-SNAPSHOT/). The page title is "MAJOR LEAGUE BASEBALL 2023". At the top, there is an "MLB" logo. Below it, a message says "MLB Logo From the website Link: <https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png>". A section titled "PARTICIPATING TEAMS" follows, with a note: "Team Names Scrapped From the website Link: <https://www.espn.com/mlb/teams>". Below this is a dropdown menu titled "Choose a Team:" containing a list of MLB teams. The "New York Yankees" option is highlighted with a blue background.

- ✓ Chicago White Sox
- Cleveland Guardians
- Detroit Tigers
- Kansas City Royals
- Minnesota Twins
- Baltimore Orioles
- Boston Red Sox
- New York Yankees**
- Tampa Bay Rays
- Toronto Blue Jays
- Houston Astros
- Los Angeles Angels
- Oakland Athletics
- Seattle Mariners
- Texas Rangers
- Chicago Cubs
- Cincinnati Reds

3. We Select Team Los Angeles Angels from the dropdown

The screenshot shows the same web browser window as the previous one, but now the "Los Angeles Angels" option in the dropdown menu is highlighted with a blue background, indicating it has been selected.

#### 4. We hit Submit

localhost:8080/Project1Task3-1.0-SNAPSHOT/index.jsp

MAJOR LEAGUE BASEBALL 2023

MLB Logo From the website Link:  
<https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png>

PARTICIPATING TEAMS

Team Names Scrapped From the website Link:  
<https://www.espn.com/mlb/teams>

Choose a Team:

Los Angeles Angels

Submit

#### 5. Results – Team Information

localhost:8080/Project1Task3-1.0-SNAPSHOT/fetch-team-details?team=Los%2520Angels%2520Angels

MAJOR LEAGUE BASEBALL 2023

MLB Logo From the website Link:  
<https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png>

**Los Angeles Angels**

Team Logo Scrapped From the website Link:  
[https://www.sportslogos.net/teams/list\\_by\\_year/42023/2023\\_MLB\\_Logos/](https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/)

**About the Team**

Team Information Scrapped From the API Link:  
<https://statsapi.mlb.com/api/v1/teams>

**HeadQuarters:** Anaheim  
**Season:** 2023  
**Abbreviation:** LAA  
**First Appeared In:** 1961

## 6. Results – Team Information (continued) with a back button to the Home Page

The screenshot shows a web browser window with the URL [localhost:8080/Project1Task3-1.0-SNAPSHOT/fetch-team-details?team=Los%2520Angels%2520Angels](http://localhost:8080/Project1Task3-1.0-SNAPSHOT/fetch-team-details?team=Los%2520Angels%2520Angels). The page title is "The Team's Home Stadium". Below it, a note says "Home Stadium Information Scrapped From the website Link: <https://gejango.com/pages/list-of-mlb-teams>". The stadium information is listed as "Angel Stadium | Los Angeles, California | 45,517 | 1966". A section titled "Featured Players" follows, with a note "Player Information Scrapped From the API Link: <https://statsapi.mlb.com/api/v1/statistics?stats=season&group=hitting>". A specific player profile for Shohei Ohtani is shown, listing various statistics like Rank, Games Played, Ground Outs, Air Outs, Runs, Doubles, Triples, Home Runs, and Strike Outs. At the bottom left is a blue "« Back" button.

## 7. We try for another team so as to verify results are specific and accurate

The screenshot shows a web browser window with the URL [localhost:8080/Project1Task3-1.0-SNAPSHOT/index.jsp](http://localhost:8080/Project1Task3-1.0-SNAPSHOT/index.jsp). The page title is "MAJOR LEAGUE BASEBALL 2023". A dropdown menu is open, showing a list of MLB teams. The menu includes a logo for "MI R" and a link to "https://wordsabovereads/2020/06/mlb.png". The list of teams is as follows:

- ✓ Chicago White Sox
- Cleveland Guardians
- Detroit Tigers
- Kansas City Royals
- Minnesota Twins
- Baltimore Orioles
- Boston Red Sox
- New York Yankees
- Tampa Bay Rays
- Toronto Blue Jays
- Houston Astros
- Los Angeles Angels
- Oakland Athletics
- Seattle Mariners
- Texas Rangers
- Chicago Cubs
- Cincinnati Reds
- Milwaukee Brewers
- Pittsburgh Pirates
- St. Louis Cardinals
- Atlanta Braves
- Miami Marlins
- New York Mets**
- Philadelphia Phillies
- Washington Nationals
- Arizona Diamondbacks
- Colorado Rockies
- Los Angeles Dodgers
- San Diego Padres
- San Francisco Giants

## 8. We Hit Submit

MAJOR LEAGUE BASEBALL 2023

MLB Logo From the website Link:  
<https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png>

PARTICIPATING TEAMS

Team Names Scrapped From the website Link:  
<https://www.espn.com/mlb/teams>

Choose a Team:

New York Mets

Submit

## 9. Results – Team Information

MAJOR LEAGUE BASEBALL 2023

MLB Logo From the website Link:  
<https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png>

New York Mets

Team Logo Scrapped From the website Link:  
[https://www.sportlogos.net/teams/list\\_by\\_year/42023/2023\\_MLB\\_Logos/](https://www.sportlogos.net/teams/list_by_year/42023/2023_MLB_Logos/)

About the Team

Team Information Scrapped From the API Link:  
<https://statsapi.mlb.com/api/v1/teams>

HeadQuarters: Flushing  
Season: 2023  
Abbreviation: NYM  
First Appeared In: 1962

## 10. Results – Team Information (continued)

The screenshot shows a web browser window with the URL [localhost:8080/Project1Task3-1.0-SNAPSHOT/fetch-team-details?team=New%2520York%2520Mets](http://localhost:8080/Project1Task3-1.0-SNAPSHOT/fetch-team-details?team=New%2520York%2520Mets). The page displays team details for the New York Mets, including their abbreviation (NYM) and first appearance year (1962). The browser interface includes standard navigation buttons (back, forward, search, etc.) and a tab bar.

### The Team's Home Stadium

Home Stadium Information Scrapped From the website Link:  
<https://geojango.com/pages/list-of-mlb-teams>

Citi Field | Queens, New York | 41,922 | 2009

### Featured Players

Player Information Scrapped From the API Link:  
<https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting>

#### Player Name: Jeff McNeil

Rank: 44

Games Played: 154

Ground Outs: 159

Air Outs: 202

Runs: 75

Doubles: 25

Triples: 4

Home Runs: 10

Strike Outs: 65

## 11. Results – Team Information (continued) with Back button to Home Page

The screenshot shows a web browser window with the same URL as the previous screenshot. The page displays the same team details for the New York Mets. A blue rectangular button labeled "« Back" is visible at the bottom left of the page content area. The browser interface includes standard navigation buttons and a tab bar.

#### Player Name: Brandon Nimmo

Rank: 45

Games Played: 149

Ground Outs: 127

Air Outs: 160

Runs: 87

Doubles: 29

Triples: 6

Home Runs: 24

Strike Outs: 142

« Back

## Code Snippets:

1. This method scrapes team names from the URL: <https://www.espn.com/mlb/teams> and adds the values extracted to a text file called teams.txt which is used for the dropdown and also for further scraping purposes such as comparing other extracted values with team name values that we have to ensure we only gather data for MLB teams.

```
41     void ScrapeTeamName() throws IOException {
42         // Use Jsoup to fetch HTML data into a document
43         Document doc = Jsoup.connect(teamNameScrapperUrl).get();
44
45         // Select the HTML elements that contain team names
46         Elements elem = doc.select( cssQuery: ".TeamsWrapper");
47         Elements table = elem.select( query: ".ContentList__Item");
48
49         // Define the path for the output text file
50         String outputPath = "teams.txt";
51         Path path = Paths.get(outputPath);
52
53         try (FileWriter writer = new FileWriter(path.toFile())) {
54             // Loop through the HTML elements and extract team names
55             for (Element e : table) {
56                 Element h2 = e.select( cssQuery: "h2").first();
57                 if (h2 != null) {
58                     String teamName = h2.text();
59                     // Write the team name to the output text file
60                     writer.write( str: teamName + "\n");
61                 }
62             }
63             // Notify the user when team names have been successfully scraped and saved to teams.txt
64             System.out.println("Scraped data has been saved to teams.txt");
65         } catch (IOException e) {
66             // Handle any IOException that may occur during file writing
67             System.err.println("Error writing to the file: " + e.getMessage());
68         }
69     }
```

2. This method extracts team information such as location, season, abbreviation and first year of play from an API linked: <https://statsapi.mlb.com/api/v1/teams> and stores it in a list

```
137     public List<TeamInfo> extractTeamInfo(String jsonData, List<String> fileData) {  
138         List<TeamInfo> teamInfoList = new ArrayList<>();  
139  
140         // Create a JSON parser to parse the JSON data  
141         JsonParser jsonParser = new JsonParser();  
142         JsonObject rootObject = jsonParser.parse(jsonData).getAsJsonObject();  
143  
144         // Extract team data  
145         for (JsonElement teamElement : rootObject.get("teams").getAsJsonArray()) {  
146             TeamInfo teamInfo = new TeamInfo();  
147             JsonObject teamObject = teamElement.getAsJsonObject();  
148  
149             // Check if the team's name is present in the provided list  
150             if (fileData.contains(teamObject.get("name").getAsString())) {  
151                 // Extract and set team information  
152                 teamInfo.setName(teamObject.get("name").getAsString());  
153                 teamInfo.setLocationName(teamObject.get("locationName").getAsString());  
154                 teamInfo.setSeason(teamObject.get("season").getAsInt());  
155                 teamInfo.setAbbreviation(teamObject.get("abbreviation").getAsString());  
156                 teamInfo.setFirstYearOfPlay(teamObject.get("firstYearOfPlay").getAsString());  
157  
158                 // Add the TeamInfo object to the list  
159                 teamInfoList.add(teamInfo);  
160  
161             }  
162             // Print the extracted TeamInfo object  
163             System.out.println(teamInfo);  
164         }  
165     }
```

3. This method extracts team stadium information from <https://geojango.com/pages/list-of-mlb-teams>

```
177     public Map<String, String> extractStadiumInfo() throws IOException {
178         Map<String, String> combinedMap = new HashMap<>();
179         // Jsoup to fetch HTML data into a document
180         Document doc = null;
181         try {
182             doc = Jsoup.connect(stadiumUrl).get();
183         } catch (IOException e) {
184             e.printStackTrace();
185         }
186         if (doc == null) {
187             System.out.println("Failed to fetch the HTML document.");
188             return Collections.emptyMap();
189         }
190         Element table = doc.select(cssQuery: "table").first();
191         if (table == null) {
192             System.out.println("Table element not found in the HTML document.");
193             return Collections.emptyMap();
194         }
195         Elements rows = table.select(cssQuery: "tr");
196         // Create a new map to combine the values
197         Map<String, String> stadiumMap = new HashMap<>();
198
199         for (int i = 1; i < rows.size(); i++) {
200             Element row = rows.get(i);
201             Elements columns = row.select(cssQuery: "td");
202
203             // Extract data from the columns
204             String teamName = columns.get(0).text();
205             String value1 = columns.get(1).text();
206             String value2 = columns.get(2).text();
207             String value3 = columns.get(3).text();
208             String value4 = columns.get(4).text();
209
210             // Combine the values into a single string and put it into the combined map
211             String combinedValue = value1 + " | " + value2 + " | " + value3 + " | " + value4;
212             stadiumMap.put(teamName, combinedValue);
213         }
214         return stadiumMap;
215     }
216
217     1 usage
218     public String fetchStadiumInfo(String teamName) throws IOException {
219
220         Map<String, String> stadiumInfo = extractStadiumInfo();
221         String stadiumData = null;
222
223         for (Map.Entry<String, String> entry : stadiumInfo.entrySet()){
224             if(entry.getKey().equals(teamName)){
225                 System.out.println(entry.getKey() + "~~~" + entry.getValue());
226                 stadiumData = entry.getValue();
227                 break;
228             }
229         }
230         return stadiumData;
231     }
```

4. This method extracts individual team logo URLs from the website:

[https://www.sportslogos.net/teams/list\\_by\\_year/42023/2023\\_MLB\\_Logos/](https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/)

and stores it in a Map. We retrieve these URLs to display logos on our web page.

```
241     public Map<String, String> fetchLogoMap() throws IOException {
242         // Jsoup to fetch HTML data into a document
243         Document doc = Jsoup.connect(logoUrl).get();
244
245         // Select the HTML element containing the logos
246         Element logoWall = doc.select(cssQuery: "ul[class=logoWall]").first();
247
248         if (logoWall == null) {
249             System.out.println("Table element not found in the HTML document.");
250             return Collections.emptyMap();
251         }
252
253         // Create a map to store team name as the key and img src as the value
254         Map<String, String> teamLogoMap = new HashMap<>();
255
256         // Select all <a> elements within the logoWall
257         Elements anchorElements = logoWall.select(cssQuery: "a");
258
259         for (Element anchorElement : anchorElements) {
260             // Extract team name and img src
261             String teamName = anchorElement.text();
262             String imgSrc = anchorElement.select(cssQuery: "img").attr(attributeKey: "src");
263
264             // Put the data into the map
265             teamLogoMap.put(teamName, imgSrc);
266         }
267
268         return teamLogoMap;
269     }
```

5. This method extracts stats for Featured Players within the respective teams from the API linked: <https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting>

```
278     public List<TeamPlayerStats> extractTeamPlayerStats(String jsonData) {  
279         List<TeamPlayerStats> teamPlayerStatsList = new ArrayList<>();  
280         JsonParser jsonParser = new JsonParser();  
281         JSONArray jsonArray = jsonParser.parse(jsonData).getAsJsonObject() JsonObject  
282             .get("stats").getAsJsonArray().get(0) JsonElement  
283             .getAsJsonObject().get("splits").getAsJsonArray();  
284         for (JsonElement jsonElement : jsonArray) {  
285             JsonObject jsonObject = jsonElement.getAsJsonObject();  
286             TeamPlayerStats teamPlayerStats = new TeamPlayerStats();  
287  
288             // Extract rank  
289             teamPlayerStats.setRank(!jsonObject.get("rank").isJsonNull() ? jsonObject.get("rank").getAsInt() : 0);  
290  
291             // Extract team[name]  
292             JsonObject teamObject = jsonObject.getAsJsonObject(memberName: "team");  
293             teamPlayerStats.setTeamName(teamObject.get("name").getAsString());  
294  
295             // Extract player[fullName]  
296             JsonObject playerObject = jsonObject.getAsJsonObject(memberName: "player");  
297             teamPlayerStats.setPlayerName(playerObject.get("fullName").getAsString());  
298  
299             // Extract player statistics  
300             JsonObject statObject = jsonObject.getAsJsonObject(memberName: "stat");  
301             teamPlayerStats.setGamesPlayed(statObject.get("gamesPlayed").getAsInt());  
302             teamPlayerStats.setGroundOuts(statObject.get("groundOuts").getAsInt());  
303             teamPlayerStats.setAirOuts(statObject.get("airOuts").getAsInt());  
304             teamPlayerStats.setRuns(statObject.get("runs").getAsInt());  
305             teamPlayerStats.setDoubles(statObject.get("doubles").getAsInt());  
306             teamPlayerStats.setTriples(statObject.get("triples").getAsInt());  
307             teamPlayerStats.setHomeRuns(statObject.get("homeRuns").getAsInt());  
308             teamPlayerStats.setStrikeOuts(statObject.get("strikeOuts").getAsInt());  
309             teamPlayerStats.setAverage(statObject.get("avg").getAsString());  
310  
311             teamPlayerStatsList.add(teamPlayerStats);  
312         }  
313     }  
314 }
```

## 6. Home Page jsp Output

```
JSP teamDetails.jsp      ⚡ TeamController.java      ⚡ TeamModel.java      ⚡ TeamInfo.java      JSP index.jsp ×  
51   <h1>MAJOR LEAGUE BASEBALL 2023</h1>  
52  
53  
54     
55   <h5>MLB Logo From the website Link:<br><p>https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png</p><br><br>  
56  
57   <h2>PARTICIPATING TEAMS</h2>  
58  
59   <h5>Team Names Scraped From the website Link:<br><p>https://www.espn.com/mlb/teams</p><br></h5>  
60  
61   <form method="get" action="fetch-team-details" enctype="text/plain">  
62     <label for="team">Choose a Team:</label><br><br>  
63     <select name="team" id="team">  
64  
65     <%  
66  
67       TeamModel teamDetailsModel = new TeamModel();  
68       List<String> fileDataList = teamDetailsModel.readTxtFile("teams.txt");  
69  
70       for(int i =0; i< fileDataList.size() ; i++){  
71         String team = fileDataList.get(i);  
72         String teamNoSpace = team.replace(" ", "%20");  
73       %>  
74       <option value=<%=teamNoSpace%>><%=team%></option>  
75       <%  
76       }  
77       %>  
78     </select>  
79     <br><br><input type="submit" value="Submit"/>  
80   </form>  
81   <br/>  
82   </body>  
83   </html>  
84
```

## 7. Result Page jsp Output

```
37 <body>
38 <div class="header-container">
39     <h1>MAJOR LEAGUE BASEBALL 2023</h1>
40     
41     <h5>MLB Logo From the website Link:<br><p>https://wordsaboverplacement.com/wp-content/uploads/2020/06/mlb.png</p><br></h5>
42
43 </div>
44
45 <h1>${teamName}</h1>
46
47 <h2><br><img src =${teamLogoUrl} width="200" height="125"/><br></h2>
48 <h5>Team Logo Scrapped From the website Link:<br><p>https://www.sportslogos.net/teams/list_by_year/42023/2023_MLB_Logos/</p><br></h5>
49
50
51 <h2>About the Team</h2>
52 <h5>Team Information Scrapped From the API Link:<br><p>https://statsapi.mlb.com/api/v1/teams</p><br></h5>
53
54 <h3>Headquarters: ${locationName}</h3>
55 <h3>Season: ${season}</h3>
56 <h3>Abbreviation: ${abbreviation}</h3>
57 <h3>First Appeared In: ${firstYearOfPlay}</h3>
58
59 <br><br>
60
61 <h2>The Team's Home Stadium</h2>
62 <h5>Home Stadium Information Scrapped From the website Link:<br><p>https://geojango.com/pages/list-of-mlb-teams</p><br></h5>
63
64 <h3>${arenaName}</h3>
65
66 <br><br>
67
```

html > body > h3

## 8. Result Page Output jsp Continued

```
66 <br><br>
67
68 <h2>Featured Players</h2>
69 <h5>Player Information Scraped From the API Link:<br><p>https://statsapi.mlb.com/api/v1/stats?stats=season&group=hitting</p><br></h5>
70
71 <c:forEach items="${playerDetails}" var="player">
72
73     <h2>Player Name: <c:out value="${player.playerName}" /></h2>
74     <h3>Rank: <c:out value="${player.rank}" /></h3>
75     <h3>Games Played: <c:out value="${player.gamesPlayed}" /></h3>
76     <h3>Ground Outs: <c:out value="${player.groundOuts}" /></h3>
77     <h3>Air Outs: <c:out value="${player.airOuts}" /></h3>
78     <h3>Runs: <c:out value="${player.runs}" /></h3>
79     <h3>Doubles: <c:out value="${player.doubles}" /></h3>
80     <h3>Triples: <c:out value="${player.triples}" /></h3>
81     <h3>Home Runs: <c:out value="${player.homeRuns}" /></h3>
82     <h3>Strike Outs: <c:out value="${player.strikeOuts}" /></h3>
83     <br><br>
84 </c:forEach>
85
86
87 <br><br>
88 <div class="button-container">
89     <a href="index.jsp" class="button">&laquo; Back</a>
90 </div>
91
92 </body>
93 </html>
94
```

## 9. This code invokes the Servlet Method

```
61 @Override
62     public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
63
64         request.setAttribute("teamNameScrapperUrl", teamNameScrapperUrl);
65         // select path for file which will store the scraped mlb team names
66
67         // Fetch JSON data from the API
68         response.setContentType("text/html");
69         String teamName = request.getParameter("team").replace(target: "%20", replacement: " ");
70
71
72         List<TeamPlayerStats> playerDetails = teamDetailsModel.fetchPlayersFromStadium(teamName);
73         System.out.println(playerDetails.size());
74         TeamInfo teamInfo = teamDetailsModel.fetchTeamInfo(teamName);
75
76         String stadiumInfo = teamDetailsModel.fetchStadiumInfo(teamName);
77         System.out.println("gwerthjkhgfd" + stadiumInfo);
78
79         request.setAttribute("teamName", teamName);
80         request.setAttribute("teamLogoUrl", teamDetailsView.getTeamLogoMap().get(teamName));
81         request.setAttribute("playerDetails", playerDetails);
82
83
84         request.setAttribute("locationName", teamInfo.getLocationName());
85         request.setAttribute("season", teamInfo.getSeason());
86         request.setAttribute("abbreviation", teamInfo.getAbbreviation());
87         request.setAttribute("firstYearOfPlay", teamInfo.getFirstYearOfPlay());
88
89         request.setAttribute("arenaName", stadiumInfo);
90
91
92         // Transfer control over the the correct "view"
93         RequestDispatcher view = request.getRequestDispatcher("teamDetails.jsp");
94         view.forward(request, response);
95
96     }
```