

## PennyFlaskSpa/runserver.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # runserver.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import penny
10:
11: def main():
12:
13:     if len(sys.argv) != 2:
14:         print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
15:         sys.exit(1)
16:
17:     try:
18:         port = int(sys.argv[1])
19:     except Exception:
20:         print('Port must be an integer.', file=sys.stderr)
21:         sys.exit(1)
22:
23:     try:
24:         penny.app.run(host='0.0.0.0', port=port, debug=True)
25:     except Exception as ex:
26:         print(ex, file=sys.stderr)
27:         sys.exit(1)
28:
29: if __name__ == '__main__':
30:     main()

```

## PennyFlaskSpa/database.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # database.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sqlite3
9: import contextlib
10:
11: #-----
12:
13: _DATABASE_URL = 'file:penny.sqlite?mode=ro'
14:
15: #-----
16:
17: def get_books(author):
18:
19:     books = []
20:
21:     with sqlite3.connect(_DATABASE_URL, isolation_level=None,
22:                          uri=True) as connection:
23:
24:         with contextlib.closing(connection.cursor()) as cursor:
25:
26:             query_str = "SELECT isbn, author, title FROM books "
27:             query_str += "WHERE author LIKE ?"
28:             cursor.execute(query_str, [author+'%'])
29:
30:             table = cursor.fetchall()
31:             for row in table:
32:                 book = {'isbn': row[0], 'author': row[1],
33:                        'title': row[2]}
34:                 books.append(book)
35:
36:     return books
37:
38: #-----
39:
40: def _test():
41:
42:     books = get_books('ker')
43:     for book in books:
44:         print(book['isbn'])
45:         print(book['author'])
46:         print(book['title'])
47:         print()
48:
49: if __name__ == '__main__':
50:     _test()

```

## PennyFlaskSpa/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import json
9: import flask
10: import database
11:
12: #-----
13:
14: app = flask.Flask(__name__)
15:
16: #-----
17:
18: @app.route('/', methods=['GET'])
19: @app.route('/index', methods=['GET'])
20: def index():
21:
22:     return flask.send_file('index.html')
23:
24: #-----
25:
26: @app.route('/searchresults', methods=['GET'])
27: def search_results():
28:
29:     author = flask.request.args.get('author')
30:     if author is None:
31:         author = ''
32:     author = author.strip()
33:
34:     if author == '':
35:         books = []
36:     else:
37:         books = database.get_books(author) # Exception handling omitted
38:
39:     json_doc = json.dumps(books)
40:     response = flask.make_response(json_doc)
41:     response.headers['Content-Type'] = 'application/json'
42:     return response

```

## blank (Page 1 of 1)

1: This page is intentionally blank.

## PennyFlaskSpa/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <hr>
8:     Good <span id="ampmSpan"></span> and welcome to
9:     <strong>Penny.com</strong>
10:    <hr>
11:    <h1>Author Search</h1>
12:    Please enter an author name:
13:    <input type="text" id="authorInput" autoFocus>
14:    <hr>
15:    <div id="resultsDiv"></div>
16:    <hr>
17:    Date and time: <span id="datetimeSpan"></span><br>
18:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
19:    Bob Dondero</a>
20:    <hr>
21:    <script src=
22:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
23:    </script>
24:    <!-- <script src="/static/mustache.min.js"></script> -->
25:    <script>
26:      'use strict';
27:
28:      function getAmPm() {
29:        let dateTime = new Date();
30:        let hours = dateTime.getHours();
31:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
32:        let ampmSpan = document.getElementById('ampmSpan');
33:        ampmSpan.innerHTML = amPm;
34:      }
35:
36:      function getDateTime() {
37:        let dateTime = new Date();
38:        let datetimeSpan =
39:        document.getElementById('datetimeSpan');
40:        datetimeSpan.innerHTML = dateTime.toLocaleString();
41:      }
42:
43:      function convertToHtml(books) {
44:        let template = `
45:          {{#books}}
46:            {{isbn}}:
47:            <strong>{{author}}</strong>:
48:            {{title}}
49:            <br>
50:          {{/books}}
51:        `;
52:        let map = {books: books};
53:        let html = Mustache.render(template, map);
54:        return html;
55:      }
56:
57:      function handleResponse() {

```

## PennyFlaskSpa/index.html (Page 2 of 2)

```

58:        if (this.status !== 200) {
59:          alert('Error: Failed to fetch data from server');
60:          return;
61:        }
62:        let books = JSON.parse(this.response);
63:        let html = convertToHtml(books);
64:        let resultsDiv = document.getElementById('resultsDiv');
65:        resultsDiv.innerHTML = html;
66:      }
67:
68:      function handleError() {
69:        alert('Error: Failed to fetch data from server');
70:      }
71:
72:      let request = null;
73:
74:      function getResults() {
75:        let authorInput = document.getElementById('authorInput');
76:        let author = authorInput.value;
77:        let encodedAuthor = encodeURIComponent(author);
78:        let url = '/searchresults?author=' + encodedAuthor;
79:        if (request !== null)
80:          request.abort();
81:        request = new XMLHttpRequest();
82:        request.onload = handleResponse;
83:        request.onerror = handleError;
84:        request.open('GET', url);
85:        request.send();
86:      }
87:
88:      let timer = null;
89:
90:      function debouncedGetResults() {
91:        clearTimeout(timer);
92:        timer = window.setTimeout(getResults, 500);
93:      }
94:
95:      function setupHeader() {
96:        getAmPm();
97:        let apInterval = window.setInterval(getAmPm, 1000);
98:        window.addEventListener('beforeunload',
99:          function(e) {window.clearInterval(apInterval);})
100:      }
101:
102:      function setupFooter() {
103:        getDateTime();
104:        let dtInterval = window.setInterval(getDateTime, 1000);
105:        window.addEventListener('beforeunload',
106:          function(e) {window.clearInterval(dtInterval);})
107:      }
108:
109:      function setup() {
110:        setupHeader();
111:        setupFooter();
112:        let authorInput = document.getElementById('authorInput');
113:        authorInput.addEventListener('input', debouncedGetResults);
114:      }
115:
116:      document.addEventListener('DOMContentLoaded', setup);
117:
118:    </script>
119:  </body>
120: </html>

```

**PennySparkSpa/runserver (Page 1 of 1)**

```
1: #!/usr/bin/env bash
2:
3: #-----
4: # runserver
5: # Author: Bob Dondero
6: #-----
7:
8: if [ $# -ne 1 ]
9: then
10:     echo "usage: runserver port"
11:     exit 1
12: fi
13:
14: mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
15:     -Dexec.args="$@"
```

**PennySparkSpa/runserver.bat (Page 1 of 1)**

```
1: REM -----
2: REM runserver.bat
3: REM Author: Bob Dondero
4: REM -----
5:
6: mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
7:     -Dexec.args="%1"
```

**PennySparkSpa/pom.xml (Page 1 of 1)**

```

1: <?xml version="1.0" encoding="UTF-8"?>
2:
3: <project xmlns="http://maven.apache.org/POM/4.0.0"
4:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5:   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6:     http://maven.apache.org/xsd/maven-4.0.0.xsd">
7:
8:   <modelVersion>4.0.0</modelVersion>
9:
10:  <groupId>edu.princeton.penny</groupId>
11:  <artifactId>Penny</artifactId>
12:  <version>1.0</version>
13:  <name>Penny</name>
14:
15:  <properties>
16:    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17:    <maven.compiler.source>21</maven.compiler.source>
18:    <maven.compiler.target>21</maven.compiler.target>
19:  </properties>
20:
21:  <dependencies>
22:    <dependency>
23:      <groupId>org.xerial</groupId>
24:      <artifactId>sqlite-jdbc</artifactId>
25:      <version>3.34.0</version>
26:    </dependency>
27:    <dependency>
28:      <groupId>com.sparkjava</groupId>
29:      <artifactId>spark-core</artifactId>
30:      <version>2.9.3</version>
31:    </dependency>
32:    <dependency>
33:      <groupId>com.google.code.gson</groupId>
34:      <artifactId>gson</artifactId>
35:      <version>2.10.1</version>
36:    </dependency>
37:  </dependencies>
38: </project>

```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

PennySparkSpa/src/main/java/edu/princeton/penny/Database.java (Page 1 of 1) PennySparkSpa/src/main/java/edu/princeton/penny/Penny.java (Page 1 of 1)

```

1: //-----
2: // Database.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import java.sql.DriverManager;
9: import java.sql.Connection;
10: import java.sql.PreparedStatement;
11: import java.sql.ResultSet;
12: import org.sqlite.SQLiteConfig;
13: import java.util.ArrayList;
14: import java.util.Hashtable;
15: import java.io.File;
16: import java.net.URL;
17:
18: public class Database
19: {
20:     private static final String DATABASE_FILE = "penny.sqlite";
21:
22:     public static ArrayList<Hashtable> getBooks(String author)
23:     throws Exception
24:     {
25:         ClassLoader classLoader = Database.class.getClassLoader();
26:         URL resource = classLoader.getResource(DATABASE_FILE);
27:         if (resource == null)
28:             throw new Exception("Database connection failed");
29:         String fileName = resource.getFile();
30:
31:         SQLiteConfig config = new SQLiteConfig();
32:         config.setReadOnly(true);
33:         Connection connection =
34:             DriverManager.getConnection("jdbc:sqlite:" + fileName,
35:                 config.toProperties());
36:
37:         try
38:         {
39:             ArrayList<Hashtable> result = new ArrayList<Hashtable>();
40:
41:             PreparedStatement statement = connection.prepareStatement(
42:                 "SELECT isbn, author, title FROM books "
43:                 + "WHERE author LIKE ?");
44:             statement.setString(1, author + "%");
45:             ResultSet resultSet = statement.executeQuery();
46:
47:             while (resultSet.next())
48:             {
49:                 String isbn = resultSet.getString("isbn");
50:                 String foundAuthor = resultSet.getString("author");
51:                 String title = resultSet.getString("title");
52:                 Hashtable hashtable = new Hashtable();
53:                 hashtable.put("isbn", isbn);
54:                 hashtable.put("author", foundAuthor);
55:                 hashtable.put("title", title);
56:                 result.add(hashtable);
57:             }
58:             return result;
59:         }
60:         finally
61:         {
62:             connection.close();
63:         }
64:     }
65: }

```

```

1: //-----
2: // Penny.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import spark.Spark;
9: import spark.Request;
10: import spark.Response;
11: import java.util.ArrayList;
12: import java.util.Hashtable;
13: import com.google.gson.Gson;
14:
15: public class Penny
16: {
17:     private static String searchResults(Request req, Response res)
18:     {
19:         String prevAuthor;
20:         ArrayList<Hashtable> books;
21:
22:         String author = req.queryParams("author");
23:         if (author == null)
24:             author = "";
25:         author = author.trim();
26:
27:         if (author.equals(""))
28:         {
29:             books = new ArrayList<Hashtable>();
30:         }
31:         else
32:         {
33:             try
34:             {
35:                 books = Database.getBooks(author);
36:             }
37:             catch (Exception e)
38:             {
39:                 // Database exception handling omitted.
40:                 return null;
41:             }
42:         }
43:
44:         res.type("application/json");
45:         return new Gson().toJson(books);
46:     }
47:
48:     public static void main(String[] args)
49:     {
50:         if (args.length != 1)
51:         {
52:             System.err.println("Usage: java Penny port");
53:             System.exit(1);
54:         }
55:
56:         Spark.port(Integer.parseInt(args[0]));
57:         Spark.staticFiles.location("/public");
58:         Spark.get("/",
59:             (req, res) -> {res.redirect("/index.html"); return null;}
60:         );
61:         Spark.get("/searchresults",
62:             (req, res) -> searchResults(req, res)
63:         );
64:     }
65: }

```

## PennyExpressSpa/runserver.js (Page 1 of 1)

```

1: //-----
2: // runserver.js
3: // Author: Bob Dondero
4: //-----
5:
6: const penny = require('./penny.js');
7:
8: function main() {
9:
10:   if (process.argv.length !== 3) {
11:     process.stderr.write('Usage: node penny.js port\n');
12:     process.exit(1);
13:   }
14:
15:   let port = process.argv[2];
16:
17:   penny.app.listen(port,
18:     () => {
19:       process.stdout.write('Listening at port ' + port + '\n');
20:     }
21:   );
22: }
23:
24: if (require.main === module)
25:   main();

```

## PennyExpressSpa/package.json (Page 1 of 1)

```

1: {
2:   "dependencies": {
3:     "express": "^4.19.2",
4:     "sqlite3": "^5.1.7"
5:   }
6: }

```

## PennyExpressSpa/database.js (Page 1 of 1)

```

1: //-----
2: // database.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const sqlite3 = require('sqlite3').verbose();
9:
10: //-----
11:
12: function getBooks(author, callback) {
13:   let db = new sqlite3.Database('penny.sqlite', sqlite3.OPEN_READONLY);
14:   let stmt =
15:     'SELECT isbn, author, title FROM books WHERE author LIKE ?';
16:   db.all(stmt, [author + '%'],
17:     (err, data) => {db.close(); callback(err, data);}
18:   );
19: }
20:
21: //-----
22:
23: // For testing:
24:
25: function writeRows(err, rows) {
26:   console.log('in writeRows');
27:   if (err)
28:     console.log(err.message);
29:   else
30:     console.log(rows);
31: }
32:
33: function main() {
34:   try {
35:     getBooks('Kernighan', writeRows);
36:     getBooks('Sedgewick', writeRows);
37:     getBooks('Dondero', writeRows);
38:   }
39:   catch (err) {
40:     console.log(err.message);
41:   };
42: }
43:
44: //-----
45:
46: if (require.main === module)
47:   main();
48:
49: //-----
50:
51: module.exports = { getBooks };

```

## PennyExpressSpa/penny.js (Page 1 of 1)

```

1: //-----
2: // penny.js
3: // Author: Bob Dondero
4: //-----
5:
6: // Omit handling of server-side errors.
7:
8: const express = require('express');
9: const database = require('./database');
10:
11: //-----
12:
13: function index(req, res) {
14:   process.stdout.write(req.originalUrl + '\n');
15:   res.sendFile('index.html', {root: __dirname},
16:     (err) => { res.end(); if (err) throw(err); }
17:   );
18: }
19:
20: //-----
21:
22: function searchResults(req, res) {
23:   function handleBooks(err, books) { // Error handling omitted
24:     res.setHeader('Content-Type', 'application/json');
25:     res.end(JSON.stringify(books));
26:   }
27:
28:   process.stdout.write(req.originalUrl + '\n');
29:
30:   let author = req.query.author;
31:   if (! author)
32:     author = '';
33:   author = author.trim();
34:
35:   if (author === '') {
36:     res.setHeader('Content-Type', 'application/json');
37:     res.end(JSON.stringify([]));
38:   }
39:   else
40:     database.getBooks(author, handleBooks);
41: }
42:
43: //-----
44:
45: let app = express();
46: app.set('view engine', 'html');
47: app.set('views', '.');
48:
49: app.get('/', index);
50: app.get('/index', index);
51: app.get('/searchresults', searchResults);
52:
53: module.exports = { app };

```

**PennyServletsSpa/runserver (Page 1 of 1)**

```
1: #!/usr/bin/env bash
2:
3: #-----
4: # runserver
5: # Author: Bob Dondero
6: #-----
7:
8: if [ $# -ne 1 ]
9: then
10:     echo "usage: runserver port"
11:     exit 1
12: fi
13:
14: mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
15:     -Dexec.args="$@"
```

**PennyServletsSpa/runserver.bat (Page 1 of 1)**

```
1: REM -----
2: REM runserver.bat
3: REM Author: Bob Dondero
4: REM -----
5:
6: mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
7:     -Dexec.args="%1"
```

**PennyServletsSpa/pom.xml (Page 1 of 1)**

```

1: <?xml version="1.0" encoding="UTF-8"?>
2:
3: <project xmlns="http://maven.apache.org/POM/4.0.0"
4:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5:   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
6:     http://maven.apache.org/xsd/maven-4.0.0.xsd">
7:
8:   <modelVersion>4.0.0</modelVersion>
9:
10:  <groupId>edu.princeton.penny</groupId>
11:  <artifactId>PennyServletsSpa</artifactId>
12:  <version>1.0</version>
13:  <name>PennyServletsSpa</name>
14:
15:  <properties>
16:    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17:    <maven.compiler.source>21</maven.compiler.source>
18:    <maven.compiler.target>21</maven.compiler.target>
19:  </properties>
20:
21:  <dependencies>
22:    <dependency>
23:      <groupId>org.eclipse.jetty</groupId>
24:      <artifactId>jetty-webapp</artifactId>
25:      <version>9.4.18.v20190429</version>
26:    </dependency>
27:    <dependency>
28:      <groupId>org.xerial</groupId>
29:      <artifactId>sqlite-jdbc</artifactId>
30:      <version>3.34.0</version>
31:    </dependency>
32:    <dependency>
33:      <groupId>com.google.code.gson</groupId>
34:      <artifactId>gson</artifactId>
35:      <version>2.11.0</version>
36:    </dependency>
37:  </dependencies>
38:
39: </project>

```

**blank (Page 1 of 1)**

1: This page is intentionally blank.

PennyServletsSpa/src/main/java/edu/princeton/penny/Database.java (Page 1 of 2) PennyServletsSpa/src/main/java/edu/princeton/penny/IndexServlet.java (Page 2 of 2)

```

1: //-----
2: // Database.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import java.sql.DriverManager;
9: import java.sql.Connection;
10: import java.sql.PreparedStatement;
11: import java.sql.ResultSet;
12: import org.sqlite.SQLiteConfig;
13: import java.util.ArrayList;
14: import java.util.Hashtable;
15: import java.net.URL;
16:
17: public class Database
18: {
19:     private static final String DATABASE_FILE = "penny.sqlite";
20:
21:     public static ArrayList<Hashtable> getBooks(String author)
22:         throws Exception
23:     {
24:         ClassLoader classLoader = Database.class.getClassLoader();
25:         URL resource = classLoader.getResource(DATABASE_FILE);
26:         if (resource == null)
27:             throw new Exception("Database connection failed");
28:         String fileName = resource.getFile();
29:
30:         SQLiteConfig config = new SQLiteConfig();
31:         config.setReadOnly(true);
32:         Connection connection =
33:             DriverManager.getConnection("jdbc:sqlite:" + fileName,
34:                 config.toProperties());
35:
36:         try
37:         {
38:             ArrayList<Hashtable> result = new ArrayList<Hashtable>();
39:
40:             PreparedStatement statement = connection.prepareStatement(
41:                 "SELECT isbn, author, title FROM books "
42:                 + "WHERE author LIKE ?");
43:             statement.setString(1, author + "%");
44:             ResultSet resultSet = statement.executeQuery();
45:
46:             while (resultSet.next())
47:             {
48:                 String isbn = resultSet.getString("isbn");
49:                 String foundAuthor = resultSet.getString("author");
50:                 String title = resultSet.getString("title");
51:                 Hashtable hashtable = new Hashtable();
52:                 hashtable.put("isbn", isbn);
53:                 hashtable.put("author", foundAuthor);
54:                 hashtable.put("title", title);
55:                 result.add(hashtable);
56:             }
57:
58:             return result;
59:         }
60:         finally
61:         {
62:             connection.close();
63:         }
64:     }
65: }

```

```

1: //-----
2: // IndexServlet.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import java.io.IOException;
9: import javax.servlet.ServletException;
10: import javax.servlet.http.HttpServlet;
11: import javax.servlet.http.HttpServletRequest;
12: import javax.servlet.http.HttpServletResponse;
13: import javax.servlet.RequestDispatcher;
14: import java.net.URL;
15: import java.io.BufferedReader;
16: import java.io.FileReader;
17: import java.io.PrintWriter;
18:
19: public class IndexServlet extends HttpServlet
20: {
21:     protected void doGet(HttpServletRequest request,
22:         HttpServletResponse response)
23:         throws ServletException, IOException
24:     {
25:
26:         //RequestDispatcher requestDispatcher =
27:         //    request.getRequestDispatcher("/index.html");
28:         //requestDispatcher.forward(request, response);
29:
30:         response.setContentType("text/html");
31:         ClassLoader classLoader = IndexServlet.class.getClassLoader();
32:         URL resource = classLoader.getResource("index.html");
33:         String fileName = resource.getFile();
34:         BufferedReader reader = new BufferedReader(
35:             new FileReader(fileName));
36:         PrintWriter writer = response.getWriter();
37:         String line;
38:         while ((line = reader.readLine()) != null)
39:             writer.println(line);
40:     }
41: }

```

PennyServletsSpa/src/main/java/edu/princeton/penny/SearchResultsServlet.java (Page 1 of 1) PennyServletsSpa/src/main/java/edu/princeton/penny/Penny.java (Page 1 of 1)

```

1: //-----
2: // SearchResultsServlet.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import java.io.PrintWriter;
9: import java.io.IOException;
10: import java.util.ArrayList;
11: import java.util.Hashtable;
12: import javax.servlet.ServletException;
13: import javax.servlet.http.HttpServlet;
14: import javax.servlet.http.HttpServletRequest;
15: import javax.servlet.http.HttpServletResponse;
16: import com.google.gson.Gson;
17:
18: public class SearchResultsServlet extends HttpServlet
19: {
20:     protected void doGet(HttpServletRequest request,
21:         HttpServletResponse response)
22:         throws ServletException, IOException
23:     {
24:         ArrayList<Hashtable> books;
25:
26:         String author = request.getParameter("author");
27:         if (author == null)
28:             author = "";
29:         author = author.trim();
30:
31:         if (author.equals(""))
32:             books = new ArrayList<Hashtable>();
33:         else
34:             try
35:             {
36:                 books = Database.getBooks(author);
37:             }
38:             catch (Exception e)
39:             {
40:                 // Database exception handling omitted.
41:                 return;
42:             }
43:
44:         String jsonStr = new Gson().toJson(books);
45:         PrintWriter out = response.getWriter();
46:         response.setContentType("application/json");
47:         response.setCharacterEncoding("UTF-8");
48:         out.print(jsonStr);
49:         out.flush();
50:     }
51: }
```

```

1: //-----
2: // Penny.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import org.eclipse.jetty.server.Server;
9: import org.eclipse.jetty.servlet.ServletHandler;
10:
11: public class Penny
12: {
13:     public static void main( String[] args )
14:     {
15:         if (args.length != 1)
16:         {
17:             System.err.println("Usage: java Penny port");
18:             System.exit(1);
19:         }
20:
21:         Server server = new Server(Integer.parseInt(args[0]));
22:         ServletHandler handler = new ServletHandler();
23:         server.setHandler(handler);
24:
25:         handler.addServletWithMapping(
26:             IndexServlet.class, "/");
27:         handler.addServletWithMapping(
28:             IndexServlet.class, "/index");
29:         handler.addServletWithMapping(
30:             SearchResultsServlet.class, "/searchresults");
31:
32:         try
33:         {
34:             server.start();
35:             server.join();
36:         }
37:         catch (Exception e) {System.err.println(e); }
38:     }
39: }
```

## PennyServletsSpa/src/main/resources/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <hr>
8:     Good <span id="ampmSpan"></span> and welcome to
9:     <strong>Penny.com</strong>
10:    <hr>
11:
12:    <h1>Author Search</h1>
13:    Please enter an author name:
14:    <input type="text" id="authorInput" autoFocus>
15:    <hr>
16:    <div id="resultsDiv"></div>
17:
18:    <hr>
19:    Date and time: <span id="datetimeSpan"></span><br>
20:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
21:    Bob Dondero</a>
22:    <hr>
23:
24:    <script src=
25:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
26:    </script>
27:
28:    <!-- <script src="/static/mustache.min.js"></script> -->
29:
30:    <script>
31:      'use strict';
32:
33:      function getAmPm() {
34:        let dateTime = new Date();
35:        let hours = dateTime.getHours();
36:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
37:        let ampMspan = document.getElementById('ampmSpan');
38:        ampMspan.innerHTML = amPm;
39:      }
40:
41:      function getDateTime() {
42:        let dateTime = new Date();
43:        let datetimeSpan =
44:        document.getElementById('datetimeSpan');
45:        datetimeSpan.innerHTML = dateTime.toLocaleString();
46:      }
47:
48:      function convertToHtml(books) {
49:        let template = `
50:          {{#books}}
51:            {{isbn}}:
52:            <strong>{{author}}</strong>:
53:            {{title}}
54:            <br>
55:          {{/books}}
56:        `;
57:        let map = {books: books};
58:        let html = Mustache.render(template, map);
59:        return html;
60:      }
61:
62:      function handleResponse() {

```

## PennyServletsSpa/src/main/resources/index.html (Page 2 of 2)

```

63:        if (this.status !== 200) {
64:          alert('Error: Failed to fetch data from server');
65:          return;
66:        }
67:        let books = JSON.parse(this.response);
68:        let html = convertToHtml(books);
69:        let resultsDiv = document.getElementById('resultsDiv');
70:        resultsDiv.innerHTML = html;
71:      }
72:
73:      function handleError() {
74:        alert('Error: Failed to fetch data from server');
75:      }
76:
77:      let request = null;
78:
79:      function getResults() {
80:        let authorInput = document.getElementById('authorInput');
81:        let author = authorInput.value;
82:        let encodedAuthor = encodeURIComponent(author);
83:        let url = '/searchresults?author=' + encodedAuthor;
84:        if (request !== null)
85:          request.abort();
86:        request = new XMLHttpRequest();
87:        request.onload = handleResponse;
88:        request.onerror = handleError;
89:        request.open('GET', url);
90:        request.send();
91:      }
92:
93:      let timer = null;
94:
95:      function debouncedGetResults() {
96:        clearTimeout(timer);
97:        timer = window.setTimeout(getResults, 500);
98:      }
99:
100:      function setupHeader() {
101:        getAmPm();
102:        let apInterval = window.setInterval(getAmPm, 1000);
103:        window.addEventListener('beforeunload',
104:          function(e) {window.clearInterval(apInterval);})
105:      }
106:
107:      function setupFooter() {
108:        getDateTime();
109:        let dtInterval = window.setInterval(getDateTime, 1000);
110:        window.addEventListener('beforeunload',
111:          function(e) {window.clearInterval(dtInterval);})
112:      }
113:
114:      function setup() {
115:        setupHeader();
116:        setupFooter();
117:        let authorInput = document.getElementById('authorInput');
118:        authorInput.addEventListener('input', debouncedGetResults);
119:      }
120:
121:      document.addEventListener('DOMContentLoaded', setup);
122:
123:    </script>
124:  </body>
125: </html>

```

**PennySpringSpa/runserver (Page 1 of 1)**

```
1: #!/usr/bin/env bash
2:
3: #-----
4: # runserver
5: # Author: Bob Dondero
6: #-----
7:
8: # To change the server port, edit the application.properties file.
9:
10: if [ $# -ne 0 ]
11: then
12:     echo "usage: runserver"
13:     exit 1
14: fi
15:
16: ./mvnw spring-boot:run
```

**PennySpringSpa/runserver.bat (Page 1 of 1)**

```
1: REM -----
2: REM runserver.bat
3: REM Author: Bob Dondero
4: REM -----
5:
6: REM To change the server port, edit the application.properties file.
7:
8: mvnw.cmd spring-boot:run
```

## PennySpringSpa/pom.xml (Page 1 of 1)

```

1: <?xml version="1.0" encoding="UTF-8"?>
2: <project xmlns="http://maven.apache.org/POM/4.0.0"
3:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4:   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5:     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6:   <modelVersion>4.0.0</modelVersion>
7:   <parent>
8:     <groupId>org.springframework.boot</groupId>
9:     <artifactId>spring-boot-starter-parent</artifactId>
10:    <version>3.4.4</version>
11:    <relativePath/> <!-- lookup parent from repository -->
12:  </parent>
13:  <groupId>edu.princeton</groupId>
14:  <artifactId>penny</artifactId>
15:  <version>0.0.1-SNAPSHOT</version>
16:  <packaging>war</packaging>
17:  <name>penny</name>
18:  <description>PennySpring Application</description>
19:  <url/>
20:  <licenses>
21:    <license/>
22:  </licenses>
23:  <developers>
24:    <developer/>
25:  </developers>
26:  <scm>
27:    <connection/>
28:    <developerConnection/>
29:    <tag/>
30:    <url/>
31:  </scm>
32:  <properties>
33:    <java.version>21</java.version>
34:  </properties>
35:  <dependencies>
36:    <dependency>
37:      <groupId>org.springframework.boot</groupId>
38:      <artifactId>spring-boot-starter-web</artifactId>
39:    </dependency>
40:    <dependency>
41:      <groupId>org.springframework.boot</groupId>
42:      <artifactId>spring-boot-starter-tomcat</artifactId>
43:      <scope>provided</scope>
44:    </dependency>
45:    <dependency>
46:      <groupId>org.springframework.boot</groupId>
47:      <artifactId>spring-boot-starter-test</artifactId>
48:      <scope>test</scope>
49:    </dependency>
50:    <dependency>
51:      <groupId>org.xerial</groupId>
52:      <artifactId>sqlite-jdbc</artifactId>
53:      <version>3.34.0</version>
54:    </dependency>
55:  </dependencies>
56:  <build>
57:    <plugins>
58:      <plugin>
59:        <groupId>org.springframework.boot</groupId>
60:        <artifactId>spring-boot-maven-plugin</artifactId>
61:      </plugin>
62:    </plugins>
63:  </build>
64: </project>

```

## PennySpringSpa/src/main/resources/application.properties (Page 1 of 1)

```

1: spring.application.name=penny
2: server.port=5000

```

PennySpringSpa/src/main/java/edu/princeton/penny/Database.java (Page 1 of PennySpringSpa/src/main/java/edu/princeton/penny/PennyController.java (P

```

1: //-----
2: // Database.java
3: // Author: Bob Dondero
4: //-----
5:
6: package edu.princeton.penny;
7:
8: import java.sql.DriverManager;
9: import java.sql.Connection;
10: import java.sql.PreparedStatement;
11: import java.sql.ResultSet;
12: import org.sqlite.SQLiteConfig;
13: import java.util.ArrayList;
14: import java.util.Hashtable;
15: import java.net.URL;
16:
17: public class Database
18: {
19:     private static final String DATABASE_FILE = "penny.sqlite";
20:
21:     public static ArrayList<Hashtable> getBooks(String author)
22:     throws Exception
23:     {
24:         ClassLoader classLoader = Database.class.getClassLoader();
25:         URL resource = classLoader.getResource(DATABASE_FILE);
26:         if (resource == null)
27:             throw new Exception("Database connection failed");
28:         String fileName = resource.getFile();
29:
30:         SQLiteConfig config = new SQLiteConfig();
31:         config.setReadOnly(true);
32:         Connection connection =
33:             DriverManager.getConnection("jdbc:sqlite:" + fileName,
34:                 config.toProperties());
35:
36:         try
37:         {
38:             ArrayList<Hashtable> result = new ArrayList<Hashtable>();
39:
40:             PreparedStatement statement = connection.prepareStatement(
41:                 "SELECT isbn, author, title FROM books "
42:                 + "WHERE author LIKE ?");
43:             statement.setString(1, author + "%");
44:             ResultSet resultSet = statement.executeQuery();
45:
46:             while (resultSet.next())
47:             {
48:                 String isbn = resultSet.getString("isbn");
49:                 String foundAuthor = resultSet.getString("author");
50:                 String title = resultSet.getString("title");
51:                 Hashtable hashtable = new Hashtable();
52:                 hashtable.put("isbn", isbn);
53:                 hashtable.put("author", foundAuthor);
54:                 hashtable.put("title", title);
55:                 result.add(hashtable);
56:             }
57:
58:             return result;
59:         }
60:         finally
61:         {
62:             connection.close();
63:         }
64:     }
65: }

```

```

1: package edu.princeton.penny;
2:
3: import java.util.ArrayList;
4: import java.util.Hashtable;
5:
6: import jakarta.servlet.http.HttpServletRequestResponse;
7:
8: import org.springframework.web.bind.annotation.GetMapping;
9: import org.springframework.web.bind.annotation.RestController;
10: import org.springframework.web.bind.annotation.RequestParam;
11:
12: @RestController
13: public class PennyController {
14:
15:     @GetMapping("/searchresults")
16:     public ArrayList<Hashtable> searchResults(
17:         @RequestParam(defaultValue="") String author,
18:         HttpServletRequestResponse res)
19:     {
20:         ArrayList<Hashtable> books;
21:
22:         author = author.trim();
23:         if (author.equals(""))
24:             books = new ArrayList<Hashtable>();
25:         else
26:             try
27:             {
28:                 books = Database.getBooks(author);
29:             }
30:             catch (Exception e)
31:             {
32:                 // Database exception handling omitted.
33:                 return null;
34:             }
35:
36:         return books;
37:     }
38: }

```

PennySpringSpa/src/main/java/edu/princeton/penny/PennyApplication.java (Page 1 of 1)

```
1: package edu.princeton.penny;
2:
3: import org.springframework.boot.SpringApplication;
4: import org.springframework.boot.autoconfigure.SpringBootApplication;
5:
6: @SpringBootApplication
7: public class PennyApplication {
8:
9:     public static void main(String[] args)
10:    {
11:        SpringApplication.run(PennyApplication.class, args);
12:    }
13: }
```

1: This page is intentionally blank.

## PennySpringSpa/src/main/resources/static/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <hr>
8:     Good <span id="ampmSpan"></span> and welcome to
9:     <strong>Penny.com</strong>
10:    <hr>
11:
12:    <h1>Author Search</h1>
13:    Please enter an author name:
14:    <input type="text" id="authorInput" autoFocus>
15:    <hr>
16:    <div id="resultsDiv"></div>
17:
18:    <hr>
19:    Date and time: <span id="datetimeSpan"></span><br>
20:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
21:    Bob Dondero</a>
22:    <hr>
23:
24:    <script src=
25:    "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
26:    </script>
27:
28:    <!-- <script src="/static/mustache.min.js"></script> -->
29:
30:    <script>
31:      'use strict';
32:
33:      function getAmPm() {
34:        let dateTime = new Date();
35:        let hours = dateTime.getHours();
36:        let amPm = (hours < 12) ? 'morning' : 'afternoon';
37:        let ampMspan = document.getElementById('ampmSpan');
38:        ampMspan.innerHTML = amPm;
39:      }
40:
41:      function getDateTime() {
42:        let dateTime = new Date();
43:        let datetimeSpan =
44:        document.getElementById('datetimeSpan');
45:        datetimeSpan.innerHTML = dateTime.toLocaleString();
46:      }
47:
48:      function convertToHtml(books) {
49:        let template = `
50:          {{#books}}
51:            {{isbn}}:
52:            <strong>{{author}}</strong>:
53:            {{title}}
54:            <br>
55:          {{/books}}
56:        `;
57:        let map = {books: books};
58:        let html = Mustache.render(template, map);
59:        return html;
60:      }
61:
62:      function handleResponse() {

```

## PennySpringSpa/src/main/resources/static/index.html (Page 2 of 2)

```

63:        if (this.status !== 200) {
64:          alert('Error: Failed to fetch data from server');
65:          return;
66:        }
67:        let books = JSON.parse(this.response);
68:        let html = convertToHtml(books);
69:        let resultsDiv = document.getElementById('resultsDiv');
70:        resultsDiv.innerHTML = html;
71:      }
72:
73:      function handleError() {
74:        alert('Error: Failed to fetch data from server');
75:      }
76:
77:      let request = null;
78:
79:      function getResults() {
80:        let authorInput = document.getElementById('authorInput');
81:        let author = authorInput.value;
82:        let encodedAuthor = encodeURIComponent(author);
83:        let url = '/searchresults?author=' + encodedAuthor;
84:        if (request !== null)
85:          request.abort();
86:        request = new XMLHttpRequest();
87:        request.onload = handleResponse;
88:        request.onerror = handleError;
89:        request.open('GET', url);
90:        request.send();
91:      }
92:
93:      let timer = null;
94:
95:      function debouncedGetResults() {
96:        clearTimeout(timer);
97:        timer = window.setTimeout(getResults, 500);
98:      }
99:
100:      function setupHeader() {
101:        getAmPm();
102:        let apInterval = window.setInterval(getAmPm, 1000);
103:        window.addEventListener('beforeunload',
104:          function(e) {window.clearInterval(apInterval);})
105:      }
106:
107:      function setupFooter() {
108:        getDateTime();
109:        let dtInterval = window.setInterval(getDateTime, 1000);
110:        window.addEventListener('beforeunload',
111:          function(e) {window.clearInterval(dtInterval);})
112:      }
113:
114:      function setup() {
115:        setupHeader();
116:        setupFooter();
117:        let authorInput = document.getElementById('authorInput');
118:        authorInput.addEventListener('input', debouncedGetResults);
119:      }
120:
121:      document.addEventListener('DOMContentLoaded', setup);
122:
123:    </script>
124:  </body>
125: </html>

```