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(f'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(f'{sys.argv[0]}: Port must be an integer.',
21:             file=sys.stderr)
22:         sys.exit(1)
23:
24:     try:
25:         penny.app.run(host='0.0.0.0', port=port, debug=True)
26:     except Exception as ex:
27:         print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
28:         sys.exit(1)
29:
30: if __name__ == '__main__':
31:     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'
14:
15: #-----------------------------------------------------------------------
16:
17: def get_books(author):
18:
19:     books = []
20:
21:     with contextlib.closing(
22:         sqlite3.connect(_DATABASE_URL + '?mode=ro',
23:         isolation_level=None, uri=True)) as connection:
24:
25:         with contextlib.closing(connection.cursor()) as cursor:
26:
27:             query_str = '''
28:                 SELECT isbn, author, title FROM books
29:                 WHERE author LIKE ?
30:                 '''
31:             cursor.execute(query_str, [author+'%'])
32:
33:             table = cursor.fetchall()
34:             for row in table:
35:                 book = {'isbn': row[0], 'author': row[1],
36:                     'title': row[2]}
37:                 books.append(book)
38:
39:     return books
40:
41: #-----------------------------------------------------------------------
42:
43: def _test():
44:
45:     books = get_books('ker')
46:     for book in books:
47:         print(book['isbn'])
48:         print(book['author'])
49:         print(book['title'])
50:         print()
51:
52: if __name__ == '__main__':
53:     _test()
```

PennyFlaskSpa/penny.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import json
import flask
import database

#-----------------------------------------------------------------------

app = flask.Flask(__name__)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    return flask.send_file('index.html')

#-----------------------------------------------------------------------

@app.route('/searchresults', methods=['GET'])
def search_results():

    author = flask.request.args.get('author')
    if author is None:
        author = ''
    author = author.strip()

    if author == '':
        books = []
    else:
        books = database.get_books(author) # Exception handling omitted

    json_doc = json.dumps(books)
    response = flask.make_response(json_doc)
    response.headers['Content-Type'] = 'application/json'
    return response
```

blank (Page 1 of 1)

```
This page is intentionally blank.
```

PennyFlaskSpa/index.html (Page 1 of 2)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>

   <body>

      <hr>
      Good <span id="ampmSpan"></span> and welcome to
      <strong>Penny.com</strong>
      <hr>

      <h1>Author Search</h1>
      Please enter an author name:
      <input type="text" id="authorInput" autoFocus>
      <hr>
      <div id="resultsDiv"></div>

      <hr>
      Date and time: <span id="datetimeSpan"></span><br>
      Created by <a href="https://www.cs.princeton.edu/~rdondero">
      Bob Dondero</a>
      <hr>

      <script src=
         "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
      </script>

      <!-- <script src="/static/mustache.min.js"></script> -->

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = (hours < 12) ? 'morning' : 'afternoon';
            let ampmSpan = document.getElementById('ampmSpan');
            ampmSpan.innerHTML = amPm;
         }

         function getDateTime() {
            let dateTime = new Date();
            let datetimeSpan =
               document.getElementById('datetimeSpan');
            datetimeSpan.innerHTML = dateTime.toLocaleString();
         }

         function convertToHtml(books) {
            let template = `
               {{#books}}
                  {{isbn}}:
                  <strong>{{author}}</strong>:
                  {{title}}
                  <br>
               {{/books}}
            `;
            let map = {books: books};
            let html = Mustache.render(template, map);
            return html;
         }

         function handleResponse() {
```

PennyFlaskSpa/index.html (Page 2 of 2)

```
            if (this.status !== 200) {
               alert('Error: Failed to fetch data from server');
               return;
            }
            let books = JSON.parse(this.response);
            let html = convertToHtml(books);
            let resultsDiv = document.getElementById('resultsDiv');
            resultsDiv.innerHTML = html;
         }

         function handleError() {
            alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let authorInput = document.getElementById('authorInput');
            let author = authorInput.value;
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request !== null)
               request.abort();
            request = new XMLHttpRequest();
            request.onload = handleResponse;
            request.onerror = handleError;
            request.open('GET', url);
            request.send();
         }

         let timer = null;

         function debouncedGetResults() {
            clearTimeout(timer);
            timer = window.setTimeout(getResults, 500);
         }

         function setupHeader() {
            getAmPm();
            let apInterval = window.setInterval(getAmPm, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(apInterval);})
         }

         function setupFooter() {
            getDateTime();
            let dtInterval = window.setInterval(getDateTime, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(dtInterval);})
         }

         function setup() {
            setupHeader();
            setupFooter();
            let authorInput = document.getElementById('authorInput');
            authorInput.addEventListener('input', debouncedGetResults);
         }

         document.addEventListener('DOMContentLoaded', setup);

      </script>
   </body>
</html>
```

PennySparkSpa/runserver (Page 1 of 1)

```
#!/usr/bin/env bash

#-----------------------------------------------------------------------
# runserver
# Author: Bob Dondero
#-----------------------------------------------------------------------

if [ $# -ne 1 ]
then
   echo "usage: runserver port"
   exit 1
fi

mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
   -Dexec.args="$@"
```

PennySparkSpa/runserver.bat (Page 1 of 1)

```
REM -------------------------------------------------------------------
REM runserver.bat
REM Author: Bob Dondero
REM -------------------------------------------------------------------

mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
   -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)

```
//-------------------------------------------------------------------------
// Database.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import org.sqlite.SQLiteConfig;
import java.util.ArrayList;
import java.util.Hashtable;
import java.io.File;
import java.net.URL;

public class Database
{
   private static final String DATABASE_FILE = "penny.sqlite";

   public static ArrayList<Hashtable> getBooks(String author)
      throws Exception
   {
      ClassLoader classLoader = Database.class.getClassLoader();
      URL resource = classLoader.getResource(DATABASE_FILE);
      if (resource == null)
         throw new Exception("Database connection failed");
      String fileName = resource.getFile();

      SQLiteConfig config = new SQLiteConfig();
      config.setReadOnly(true);
      Connection connection =
         DriverManager.getConnection("jdbc:sqlite:" + fileName,
            config.toProperties());

      try
      {
         ArrayList<Hashtable> result = new ArrayList<Hashtable>();

         PreparedStatement statement = connection.prepareStatement(
            "SELECT isbn, author, title FROM books "
            + "WHERE author LIKE ?");
         statement.setString(1, author + "%");
         ResultSet resultSet = statement.executeQuery();

         while (resultSet.next())
         {
            String isbn = resultSet.getString("isbn");
            String foundAuthor = resultSet.getString("author");
            String title = resultSet.getString("title");
            Hashtable hashtable = new Hashtable();
            hashtable.put("isbn", isbn);
            hashtable.put("author", foundAuthor);
            hashtable.put("title", title);
            result.add(hashtable);
         }
         return result;
      }
      finally
      {
         connection.close();
      }
   }
}
```

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

```
//-------------------------------------------------------------------------
// Penny.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import spark.Spark;
import spark.Request;
import spark.Response;
import java.util.ArrayList;
import java.util.Hashtable;
import com.google.gson.Gson;

public class Penny
{
   private static String searchResults(Request req, Response res)
   {
      String prevAuthor;
      ArrayList<Hashtable> books;

      String author = req.queryParams("author");
      if (author == null)
         author = "";
      author = author.trim();

      if (author.equals(""))
      {
         books = new ArrayList<Hashtable>();
      }
      else
      {
         try
         {
            books = Database.getBooks(author);
         }
         catch (Exception e)
         {
            // Database exception handling omitted.
            return null;
         }
      }

      res.type("application/json");
      return new Gson().toJson(books);
   }

   public static void main(String[] args)
   {
      if (args.length != 1)
      {
         System.err.println("Usage: java Penny port");
         System.exit(1);
      }

      Spark.port(Integer.parseInt(args[0]));
      Spark.staticFiles.location("/public");
      Spark.get("/",
         (req, res) -> {res.redirect("/index.html"); return null;}
      );
      Spark.get("/searchresults",
         (req, res) -> searchResults(req, res)
      );
   }
}
```

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:
47: app.get('/', index);
48: app.get('/index', index);
49: app.get('/searchresults', searchResults);
50:
51: module.exports = { app };
```

PennyServletsSpa/runserver (Page 1 of 1)

```
#!/usr/bin/env bash

#-----------------------------------------------------------------------
# runserver
# Author: Bob Dondero
#-----------------------------------------------------------------------

if [ $# -ne 1 ]
then
   echo "usage: runserver port"
   exit 1
fi

mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
   -Dexec.args="$@"
```

PennyServletsSpa/runserver.bat (Page 1 of 1)

```
REM -------------------------------------------------------------------
REM runserver.bat
REM Author: Bob Dondero
REM -------------------------------------------------------------------

mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
   -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 1)

```
//-------------------------------------------------------------------------
// Database.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import org.sqlite.SQLiteConfig;
import java.util.ArrayList;
import java.util.Hashtable;
import java.net.URL;

public class Database
{
   private static final String DATABASE_FILE = "penny.sqlite";

   public static ArrayList<Hashtable> getBooks(String author)
      throws Exception
   {
      ClassLoader classLoader = Database.class.getClassLoader();
      URL resource = classLoader.getResource(DATABASE_FILE);
      if (resource == null)
         throw new Exception("Database connection failed");
      String fileName = resource.getFile();

      SQLiteConfig config = new SQLiteConfig();
      config.setReadOnly(true);
      Connection connection =
         DriverManager.getConnection("jdbc:sqlite:" + fileName,
            config.toProperties());

      try
      {
         ArrayList<Hashtable> result = new ArrayList<Hashtable>();

         PreparedStatement statement = connection.prepareStatement(
            "SELECT isbn, author, title FROM books "
            + "WHERE author LIKE ?");
         statement.setString(1, author + "%");
         ResultSet resultSet = statement.executeQuery();

         while (resultSet.next())
         {
            String isbn = resultSet.getString("isbn");
            String foundAuthor = resultSet.getString("author");
            String title = resultSet.getString("title");
            Hashtable hashtable = new Hashtable();
            hashtable.put("isbn", isbn);
            hashtable.put("author", foundAuthor);
            hashtable.put("title", title);
            result.add(hashtable);
         }

         return result;
      }
      finally
      {
         connection.close();
      }
   }
}
```

PennyServletsSpa/src/main/java/edu/princeton/penny/IndexServlet.ja

```
//-------------------------------------------------------------------------
// IndexServlet.java
// Author: Bob Dondero
//-------------------------------------------------------------------------

package edu.princeton.penny;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;
import java.net.URL;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.PrintWriter;

public class IndexServlet extends HttpServlet
{
   protected void doGet(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException
   {

      //RequestDispatcher requestDispatcher =
      //   request.getRequestDispatcher("/index.html");
      //requestDispatcher.forward(request, response);

      response.setContentType("text/html");
      ClassLoader classLoader = IndexServlet.class.getClassLoader();
      URL resource = classLoader.getResource("index.html");
      String fileName = resource.getFile();
      BufferedReader reader = new BufferedReader(
         new FileReader(fileName));
      PrintWriter writer = response.getWriter();
      String line;
      while ((line = reader.readLine()) != null)
         writer.println(line);
   }
}
```

PennyServletsSpa/src/main/java/edu/princeton/penny/SearchResultsServlet.java (Page 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: }
```

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

```
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)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>

   <body>

      <hr>
      Good <span id="ampmSpan"></span> and welcome to
      <strong>Penny.com</strong>
      <hr>

      <h1>Author Search</h1>
      Please enter an author name:
      <input type="text" id="authorInput" autoFocus>
      <hr>
      <div id="resultsDiv"></div>

      <hr>
      Date and time: <span id="datetimeSpan"></span><br>
      Created by <a href="https://www.cs.princeton.edu/~rdondero">
      Bob Dondero</a>
      <hr>

      <script src=
         "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
      </script>

      <!-- <script src="/static/mustache.min.js"></script> -->

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = (hours < 12) ? 'morning' : 'afternoon';
            let ampmSpan = document.getElementById('ampmSpan');
            ampmSpan.innerHTML = amPm;
         }

         function getDateTime() {
            let dateTime = new Date();
            let datetimeSpan =
               document.getElementById('datetimeSpan');
            datetimeSpan.innerHTML = dateTime.toLocaleString();
         }

         function convertToHtml(books) {
            let template = `
               {{#books}}
                  {{isbn}}:
                  <strong>{{author}}</strong>:
                  {{title}}
                  <br>
               {{/books}}
            `;
            let map = {books: books};
            let html = Mustache.render(template, map);
            return html;
         }

         function handleResponse() {
```

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

```
            if (this.status !== 200) {
               alert('Error: Failed to fetch data from server');
               return;
            }
            let books = JSON.parse(this.response);
            let html = convertToHtml(books);
            let resultsDiv = document.getElementById('resultsDiv');
            resultsDiv.innerHTML = html;
         }

         function handleError() {
            alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let authorInput = document.getElementById('authorInput');
            let author = authorInput.value;
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request !== null)
               request.abort();
            request = new XMLHttpRequest();
            request.onload = handleResponse;
            request.onerror = handleError;
            request.open('GET', url);
            request.send();
         }

         let timer = null;

         function debouncedGetResults() {
            clearTimeout(timer);
            timer = window.setTimeout(getResults, 500);
         }

         function setupHeader() {
            getAmPm();
            let apInterval = window.setInterval(getAmPm, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(apInterval);})
         }

         function setupFooter() {
            getDateTime();
            let dtInterval = window.setInterval(getDateTime, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(dtInterval);})
         }

         function setup() {
            setupHeader();
            setupFooter();
            let authorInput = document.getElementById('authorInput');
            authorInput.addEventListener('input', debouncedGetResults);
         }

         document.addEventListener('DOMContentLoaded', setup);

      </script>
   </body>
</html>
```

PennySpringSpa/runserver (Page 1 of 1)

```
#!/usr/bin/env bash

#------------------------------------------------------------------------
# runserver
# Author: Bob Dondero
#------------------------------------------------------------------------

# To change the server port, edit the application.properties file.

if [ $# -ne 0 ]
then
   echo "usage: runserver"
   exit 1
fi

./mvnw spring-boot:run
```

PennySpringSpa/runserver.bat (Page 1 of 1)

```
REM ------------------------------------------------------------------------
REM runserver.bat
REM Author: Bob Dondero
REM ------------------------------------------------------------------------

REM To change the server port, edit the application.properties file.

mvnw.cmd spring-boot:run
```

PennySpringSpa/pom.xml (Page 1 of 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
   https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>3.4.4</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>edu.princeton</groupId>
   <artifactId>penny</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>war</packaging>
   <name>penny</name>
   <description>PennySpring Application</description>
   <url/>
   <licenses>
      <license/>
   </licenses>
   <developers>
      <developer/>
   </developers>
   <scm>
      <connection/>
      <developerConnection/>
      <tag/>
      <url/>
   </scm>
   <properties>
      <java.version>21</java.version>
   </properties>
   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-web</artifactId>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-tomcat</artifactId>
         <scope>provided</scope>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-test</artifactId>
         <scope>test</scope>
      </dependency>
      <dependency>
         <groupId>org.xerial</groupId>
         <artifactId>sqlite-jdbc</artifactId>
         <version>3.34.0</version>
      </dependency>
   </dependencies>
   <build>
      <plugins>
         <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
         </plugin>
      </plugins>
   </build>
</project>
```

PennySpringSpa/src/main/resources/application.properties (Page 1 o

```
spring.application.name=penny
server.port=5000
```

PennySpringSpa/src/main/java/edu/princeton/penny/Database.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.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: }
```

PennySpringSpa/src/main/java/edu/princeton/penny/PennyController.j

```
1: package edu.princeton.penny;
2:
3: import java.util.ArrayList;
4: import java.util.Hashtable;
5:
6: import jakarta.servlet.http.HttpServletResponse;
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:       HttpServletResponse 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: }
```

This page is intentionally blank. (Page 1 of 1)

```
1: This page is intentionally blank.
```

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

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>

   <body>

      <hr>
      Good <span id="ampmSpan"></span> and welcome to
      <strong>Penny.com</strong>
      <hr>

      <h1>Author Search</h1>
      Please enter an author name:
      <input type="text" id="authorInput" autoFocus>
      <hr>
      <div id="resultsDiv"></div>

      <hr>
      Date and time: <span id="datetimeSpan"></span><br>
      Created by <a href="https://www.cs.princeton.edu/~rdondero">
      Bob Dondero</a>
      <hr>

      <script src=
         "https://cdn.jsdelivr.net/npm/mustache@4.2.0/mustache.min.js">
      </script>

      <!-- <script src="/static/mustache.min.js"></script> -->

      <script>

         'use strict';

         function getAmPm() {
            let dateTime = new Date();
            let hours = dateTime.getHours();
            let amPm = (hours < 12) ? 'morning' : 'afternoon';
            let ampmSpan = document.getElementById('ampmSpan');
            ampmSpan.innerHTML = amPm;
         }

         function getDateTime() {
            let dateTime = new Date();
            let datetimeSpan =
               document.getElementById('datetimeSpan');
            datetimeSpan.innerHTML = dateTime.toLocaleString();
         }

         function convertToHtml(books) {
            let template = `
               {{#books}}
                  {{isbn}}:
                  <strong>{{author}}</strong>:
                  {{title}}
                  <br>
               {{/books}}
            `;
            let map = {books: books};
            let html = Mustache.render(template, map);
            return html;
         }

         function handleResponse() {
```

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

```
            if (this.status !== 200) {
               alert('Error: Failed to fetch data from server');
               return;
            }
            let books = JSON.parse(this.response);
            let html = convertToHtml(books);
            let resultsDiv = document.getElementById('resultsDiv');
            resultsDiv.innerHTML = html;
         }

         function handleError() {
            alert('Error: Failed to fetch data from server');
         }

         let request = null;

         function getResults() {
            let authorInput = document.getElementById('authorInput');
            let author = authorInput.value;
            let encodedAuthor = encodeURIComponent(author);
            let url = '/searchresults?author=' + encodedAuthor;
            if (request !== null)
               request.abort();
            request = new XMLHttpRequest();
            request.onload = handleResponse;
            request.onerror = handleError;
            request.open('GET', url);
            request.send();
         }

         let timer = null;

         function debouncedGetResults() {
            clearTimeout(timer);
            timer = window.setTimeout(getResults, 500);
         }

         function setupHeader() {
            getAmPm();
            let apInterval = window.setInterval(getAmPm, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(apInterval);})
         }

         function setupFooter() {
            getDateTime();
            let dtInterval = window.setInterval(getDateTime, 1000);
            window.addEventListener('beforeunload',
               function(e) {window.clearInterval(dtInterval);})
         }

         function setup() {
            setupHeader();
            setupFooter();
            let authorInput = document.getElementById('authorInput');
            authorInput.addEventListener('input', debouncedGetResults);
         }

         document.addEventListener('DOMContentLoaded', setup);

      </script>
   </body>
</html>
```