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

PennyServlets/runserver.bat (Page 1 of 1)

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

mvn clean
mvn compile
mvn exec:java -Dexec.mainClass="edu.princeton.penny.Penny" \
   -Dexec.args="%1"
```

PennyServlets/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
   http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>

   <groupId>edu.princeton.penny</groupId>
   <artifactId>Penny</artifactId>
   <version>1.0</version>
   <name>Penny</name>

   <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.source>21</maven.compiler.source>
      <maven.compiler.target>21</maven.compiler.target>
   </properties>

   <dependencies>
      <dependency>
         <groupId>org.xerial</groupId>
         <artifactId>sqlite-jdbc</artifactId>
         <version>3.34.0</version>
      </dependency>

      <dependency>
         <groupId>org.eclipse.jetty</groupId>
         <artifactId>jetty-server</artifactId>
         <version>9.4.18.v20190429</version>
      </dependency>

      <dependency>
         <groupId>org.eclipse.jetty</groupId>
         <artifactId>jetty-webapp</artifactId>
         <version>9.4.18.v20190429</version>
      </dependency>

      <dependency>
         <groupId>org.eclipse.jetty.websocket</groupId>
         <artifactId>websocket-server</artifactId>
         <version>9.4.18.v20190429</version>
      </dependency>

      <dependency>
         <groupId>org.eclipse.jetty.websocket</groupId>
         <artifactId>websocket-servlet</artifactId>
         <version>9.4.18.v20190429</version>
      </dependency>
      <dependency>
         <groupId>org.apache.commons</groupId>
         <artifactId>commons-text</artifactId>
         <version>1.9</version>
      </dependency>

   </dependencies>

</project>
```

blank (Page 1 of 1)

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

PennyServlets/src/main/java/edu/princeton/penny/Book.java (Page 1 of 1)

```
 1: //---------------------------------------------------------------------
 2: // Book.java
 3: // Author: Bob Dondero
 4: //---------------------------------------------------------------------
 5:
 6: package edu.princeton.penny;
 7:
 8: public class Book
 9: {
10:    private String isbn;
11:    private String author;
12:    private String title;
13:
14:    public Book(String isbn, String author, String title)
15:    {
16:       this.isbn = isbn;
17:       this.author = author;
18:       this.title = title;
19:    }
20:
21:    public String toString()
22:    {
23:       return isbn + ", " + author + ", " + title;
24:    }
25:
26:    public String getIsbn() { return isbn; }
27:
28:    public String getAuthor() { return author; }
29:
30:    public String getTitle() { return title; }
31:
32: }
```

PennyServlets/src/main/java/edu/princeton/penny/Database.java (Page 1 of

```
 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.io.File;
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<Book> 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<Book> result = new ArrayList<Book>();
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:             Book book = new Book(isbn, foundAuthor, title);
52:             result.add(book);
53:          }
54:
55:          return result;
56:       }
57:       finally
58:       {
59:          connection.close();
60:       }
61:    }
62: }
```

PennyServlets/src/main/java/edu/princeton/penny/Common.java (Page 1 of 1)

```
1: //-------------------------------------------------------------------------
2: // Common.java
3: // Author: Bob Dondero
4: //-------------------------------------------------------------------------
5:
6: package edu.princeton.penny;
7:
8: import java.util.Date;
9: import java.util.Calendar;
10: import java.text.DateFormat;
11:
12: public class Common
13: {
14:    private static String getAmPm()
15:    {
16:       if (Calendar.getInstance().get(Calendar.AM_PM) == Calendar.AM)
17:          return "morning";
18:       return "afternoon";
19:    }
20:
21:    private static String getCurrentTime()
22:    {
23:       DateFormat df = DateFormat.getDateInstance(DateFormat.FULL);
24:       return df.format(new Date());
25:    }
26:
27:    public static String getHeader()
28:    {
29:       String html = "";
30:       html += "<hr>";
31:       html += "Good " + getAmPm() + " and welcome to ";
32:       html += "<strong>Penny.com</strong>";
33:       html += "<hr>";
34:       return html;
35:    }
36:
37:    public static String getFooter()
38:    {
39:       String html = "";
40:       html += "<hr>";
41:       html += "Today is " + getCurrentTime() + ".<br>";
42:       html += "Created by ";
43:       html += "<a href=\"https://www.cs.princeton.edu/~rdondero\">";
44:       html += "Bob Dondero</a>";
45:       html += "<hr>";
46:       return html;
47:    }
48: }
```

PennyServlets/src/main/java/edu/princeton/penny/Util.java (Page 1 of 1)

```
1: //-------------------------------------------------------------------------
2: // Util.java
3: // Author: Bob Dondero
4: //-------------------------------------------------------------------------
5:
6: package edu.princeton.penny;
7:
8: import javax.servlet.http.HttpServletRequest;
9: import javax.servlet.http.Cookie;
10:
11: public class Util
12: {
13:    public static String getCookie(HttpServletRequest request,
14:       String soughtName)
15:    {
16:       Cookie[] cookies = request.getCookies();
17:       if ((cookies == null) || (cookies.length == 0))
18:          return null;
19:       for (Cookie cookie : cookies)
20:          if (cookie.getName().equals(soughtName))
21:             return cookie.getValue();
22:       return null;
23:    }
24: }
```

PennyServlets/src/main/java/edu/princeton/penny/IndexServlet.java (Page 1 of 1)

```
1: //-------------------------------------------------------------------------
2: // IndexServlet.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 javax.servlet.ServletException;
11: import javax.servlet.http.HttpServlet;
12: import javax.servlet.http.HttpServletRequest;
13: import javax.servlet.http.HttpServletResponse;
14:
15: public class IndexServlet extends HttpServlet
16: {
17:    protected void doGet(HttpServletRequest request,
18:       HttpServletResponse response)
19:       throws ServletException, IOException
20:    {
21:       response.setContentType("text/html; charset=utf-8");
22:       PrintWriter pw = response.getWriter();
23:
24:       pw.println("<!DOCTYPE html>");
25:       pw.println("<html>");
26:       pw.println("<head>");
27:       pw.println("<title>Penny.com</title>");
28:       pw.println("</head>");
29:       pw.println("<body>");
30:       pw.println(Common.getHeader());
31:       pw.println("<br>");
32:       pw.println(
33:          "Click here to <a href=\"/searchform\">begin</a>.<br>");
34:       pw.println("<br>");
35:       pw.println(Common.getFooter());
36:       pw.println("</body>");
37:       pw.println("</html>");
38:    }
39: }
```

PennyServlets/src/main/java/edu/princeton/penny/SearchFormServlet.java (

```
1: //-------------------------------------------------------------------------
2: // SearchFormServlet.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.net.URLDecoder;
11: import javax.servlet.ServletException;
12: import javax.servlet.http.HttpServlet;
13: import javax.servlet.http.HttpServletRequest;
14: import javax.servlet.http.HttpServletResponse;
15: import javax.servlet.http.Cookie;
16: import org.apache.commons.text.StringEscapeUtils;
17:
18: public class SearchFormServlet extends HttpServlet
19: {
20:    protected void doGet(HttpServletRequest request,
21:       HttpServletResponse response)
22:       throws ServletException, IOException
23:    {
24:       String prevAuthor = Util.getCookie(request, "prevAuthor");
25:       if (prevAuthor == null)
26:          prevAuthor = "(none)";
27:       prevAuthor = URLDecoder.decode(prevAuthor);
28:
29:       response.setContentType("text/html; charset=utf-8");
30:       PrintWriter pw = response.getWriter();
31:       pw.println("<!DOCTYPE html>");
32:       pw.println("<html>");
33:       pw.println("<head>");
34:       pw.println("<title>Penny.com</title>");
35:       pw.println("</head>");
36:       pw.println("<body>");
37:       pw.println(Common.getHeader());
38:       pw.println("<h1>Author Search</h1>");
39:       pw.println("<form action=\"/searchresults\" method=\"get\">");
40:       pw.println("Please enter an author name:");
41:       pw.println("<input type=\"text\" name=\"author\" autofocus>");
42:       pw.println("<input type=\"submit\" value=\"Go\">");
43:       pw.println("</form>");
44:       pw.println("<br>");
45:       pw.println("<br>");
46:       pw.println("<strong>Previous author search:</strong> ");
47:       pw.println(StringEscapeUtils.escapeHtml4(prevAuthor));
48:       pw.println("<br>");
49:       pw.println("<br>");
50:       pw.println(Common.getFooter());
51:       pw.println("</body>");
52:       pw.println("</html>");
53:    }
54: }
```

```
 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.net.URLEncoder;
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 javax.servlet.http.Cookie;
17: import org.apache.commons.text.StringEscapeUtils;
18:
19: public class SearchResultsServlet extends HttpServlet
20: {
21:    protected void doGet(HttpServletRequest request,
22:       HttpServletResponse response)
23:       throws ServletException, IOException
24:    {
25:       String prevAuthor;
26:       ArrayList<Book> books;
27:
28:       String author = request.getParameter("author");
29:       if (author == null)
30:          author = "";
31:       author = author.trim();
32:
33:       if (author.equals(""))
34:       {
35:          prevAuthor = "(None)";
36:          books = new ArrayList<Book>();
37:       }
38:       else
39:       {
40:          prevAuthor = author;
41:          try
42:          {
43:             books = Database.getBooks(author);
44:          }
45:          catch (Exception e)
46:          {
47:             response.setContentType("text/html");
48:             response.getWriter().println(e.toString());
49:             return;
50:          }
51:       }
52:
53:       String safeAuthor = URLEncoder.encode(prevAuthor, "UTF-8");
54:       response.addCookie(new Cookie("prevAuthor", safeAuthor));
55:
56:       response.setContentType("text/html; charset=utf-8");
57:       PrintWriter pw = response.getWriter();
58:
59:       pw.println("<!DOCTYPE html>");
60:       pw.println("<html>");
61:       pw.println("<head>");
62:       pw.println("<title>Penny.com</title>");
63:       pw.println("</head>");
64:       pw.println("<body>");
65:       pw.println(Common.getHeader());
```



```
66:       pw.println("<h1>Author Search Results</h1>");
67:       pw.println("<h2>Books by ");
68:       pw.println(StringEscapeUtils.escapeHtml4(prevAuthor));
69:       pw.println(":</h2>");
70:
71:       if (books.size() == 0)
72:          pw.println("(None)<br>");
73:       else
74:          for (Book book : books)
75:             pw.println(
76:                StringEscapeUtils.escapeHtml4(book.getIsbn())
77:                + ": "
78:                + "<strong>"
79:                + StringEscapeUtils.escapeHtml4(book.getAuthor())
80:                + "</strong>: "
81:                + StringEscapeUtils.escapeHtml4(book.getTitle())
82:                + "<br>");
83:
84:       pw.println("<br>");
85:       pw.println("<br>");
86:       pw.println("Click here to do another ");
87:       pw.println("<a href=\"/searchform\">author search</a>.");
88:       pw.println("<br>");
89:       pw.println("<br>");
90:       pw.println("<br>");
91:       pw.println("<br>");
92:       pw.println(Common.getFooter());
93:       pw.println("</body>");
94:       pw.println("</html>");
95:    }
96: }
```

PennyServlets/src/main/java/edu/princeton/penny/Penny.java (Page 1 of 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:          SearchFormServlet.class, "/searchform");
31:       handler.addServletWithMapping(
32:          SearchResultsServlet.class, "/searchresults");
33:
34:       try
35:       {
36:          server.start();
37:          server.join();
38:       }
39:       catch (Exception e) {System.err.println(e); }
40:    }
41: }
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennySpring/runserver (Page 1 of 1)

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

PennySpring/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 spring-boot:run
```

PennySpring/pom.xml (Page 1 of 2)

```
<?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>3.3.1</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>edu.princeton</groupId>
   <artifactId>demo</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>demo</name>
   <description>Penny</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-mustache</artifactId>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-web</artifactId>
      </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>
      <dependency>
         <groupId>org.apache.commons</groupId>
         <artifactId>commons-text</artifactId>
         <version>1.9</version>
      </dependency>
   </dependencies>
   <build>
      <plugins>
         <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
         </plugin>
      </plugins>
```

PennySpring/pom.xml (Page 2 of 2)

```
   </build>
</project>
```

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

```
1: spring.application.name=penny
2: server.port=55555
3: spring.mustache.prefix=classpath:/templates/
4: spring.mustache.suffix=.html
```

blank (Page 1 of 1)

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

PennySpring/src/main/resources/templates/header.html (Page 1 of 1)

```
1: <hr>
2: Good {{ampm}} and welcome to <strong>Penny.com</strong>
3: <hr>
```

PennySpring/src/main/resources/templates/footer.html (Page 1 of 1)

```
1: <hr>
2: Today is {{currentTime}}.<br>
3: Created by <a href="https://www.cs.princeton.edu/~rdondero">
4: Bob Dondero</a>
5: <hr>
```

PennySpring/src/main/resources/templates/index.html (Page 1 of 1)

```
 1: <!DOCTYPE html>
 2:  <html>
 3:    <head>
 4:       <title>Penny.com</title>
 5:       </head>
 6:    <body>
 7:       {{>header}}
 8:       <br>
 9:       Click here to <a href="/searchform">begin</a>.<br>
10:       <br>
11:       {{>footer}}
12:    </body>
13:  </html>
```

PennySpring/src/main/resources/templates/searchform.html (Page 1 of 1)

```
 1: <!DOCTYPE html>
 2: <html>
 3:    <head>
 4:       <title>Penny.com</title>
 5:    </head>
 6:    <body>
 7:       {{>header}}
 8:       <h1>Author Search</h1>
 9:       <form action="/searchresults" method="get">
10:          Please enter an author name:
11:          <input type="text" name="author" autofocus>
12:          <input type="submit" value="Go">
13:       </form>
14:       <br>
15:       <br>
16:       <strong>Previous author search: {{ author }}</strong>
17:       <br>
18:       <br>
19:       {{>footer}}
20:    </body>
21: </html>
```

PennySpring/src/main/resources/templates/searchresults.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      {{>header}}
      <h1>Author Search Results</h1>
      <h2>Books by {{author}}:</h2>
      {{#books}}
         {{isbn}}: <strong>{{author}}</strong>: {{title}}<br>
      {{/books}}
      {{^books}}
         (None)
      {{/books}}
      <br>
      <br>
      Click here to do another
      <a href="/searchform">author search</a>.
      <br>
      <br>
      <br>
      <br>
       {{>footer}}
   </body>
</html>
```

PennySpring/src/main/java/edu/princeton/penny/PennyApplication.java (Pag

```
package edu.princeton.penny;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PennyApplication {

   public static void main(String[] args)
   {
      SpringApplication.run(PennyApplication.class, args);
   }
}
```

PennySpring/src/main/java/edu/princeton/penny/PennyController.java (Page 1 of 2)

```
 1: package edu.princeton.penny;
 2:
 3: import java.util.ArrayList;
 4: import java.util.Date;
 5: import java.util.Calendar;
 6: import java.util.Map;
 7: import java.text.DateFormat;
 8: import java.net.URLEncoder;
 9: import java.net.URLDecoder;
10: import java.io.UnsupportedEncodingException;
11:
12: import jakarta.servlet.http.Cookie;
13: import jakarta.servlet.http.HttpServletResponse;
14:
15: import org.apache.commons.text.StringEscapeUtils;
16:
17: import org.springframework.web.bind.annotation.GetMapping;
18: import org.springframework.web.bind.annotation.RestController;
19: import org.springframework.web.bind.annotation.CookieValue;
20: import org.springframework.web.bind.annotation.RequestParam;
21: import org.springframework.web.servlet.ModelAndView;
22:
23: //-------------------------------------------------------------------------
24:
25: @RestController
26: public class PennyController {
27:
28:    private static String getAmPm()
29:    {
30:       if (Calendar.getInstance().get(Calendar.AM_PM) == Calendar.AM)
31:          return "morning";
32:       return "afternoon";
33:    }
34:
35:    private static String getCurrentTime()
36:    {
37:       DateFormat df = DateFormat.getDateInstance(DateFormat.FULL);
38:       return df.format(new Date());
39:    }
40:
41:    @GetMapping("/")
42:    public ModelAndView index(Map<String, Object> model)
43:    {
44:       model.put("ampm", getAmPm());
45:       model.put("currentTime", getCurrentTime());
46:       return new ModelAndView("index", model);
47:    }
48:
49:    @GetMapping("/searchform")
50:    public ModelAndView searchForm(Map<String, Object> model,
51:       @CookieValue(value="prevAuthor", defaultValue="(None)")
52:       String prevAuthor)
53:    {
54:       prevAuthor = URLDecoder.decode(prevAuthor);
55:
56:       model.put("ampm", getAmPm());
57:       model.put("currentTime", getCurrentTime());
58:       model.put("author", prevAuthor);
59:       return new ModelAndView("searchform", model);
60:    }
61:
62:    @GetMapping("/searchresults")
63:    public ModelAndView searchResults(Map<String, Object> model,
64:       @RequestParam(defaultValue="") String author,
65:       HttpServletResponse res)
```

PennySpring/src/main/java/edu/princeton/penny/PennyController.java (Page

```
66:       throws UnsupportedEncodingException
67:    {
68:       String prevAuthor;
69:       ArrayList<Book> books = new ArrayList<Book>();
70:
71:       author = author.trim();
72:
73:       if (author.equals(""))
74:       {
75:          prevAuthor = "(None)";
76:       }
77:       else
78:       {
79:          prevAuthor = author;
80:          try
81:          {
82:             books = Database.getBooks(author);
83:          }
84:          catch (Exception e)
85:          {
86:             // Ignore database errors.
87:          }
88:       }
89:
90:       String safeAuthor = URLEncoder.encode(prevAuthor, "UTF-8");
91:       Cookie cookie = new Cookie("prevAuthor", safeAuthor);
92:       res.addCookie(cookie);
93:
94:       model.put("ampm", getAmPm());
95:       model.put("currentTime", getCurrentTime());
96:       model.put("author", prevAuthor);
97:       model.put("books", books);
98:       return new ModelAndView("searchresults", model);
99:    }
100: }
```

PennyExpress/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(
12:          'Usage: node ' + process.argv[1] + ' [port]\n');
13:       process.exit(1);
14:    }
15:
16:    let port = process.argv[2];
17:
18:    penny.app.listen(port,
19:       function() {
20:          process.stdout.write('Listening at port ' + port + '\n');
21:       }
22:    );
23: }
24:
25: if (require.main === module)
26:    main();
```

PennyExpress/package.json (Page 1 of 1)

```
1: {
2:   "dependencies": {
3:     "express": "^4.19.2",
4:     "mustache-express": "^1.3.2",
5:     "sqlite3": "^5.1.7"
6:   }
7: }
```

PennyExpress/database.js (Page 1 of 1)

```
//---------------------------------------------------------------------
// database.js
// Author: Bob Dondero
//---------------------------------------------------------------------

'use strict';

const sqlite3 = require('sqlite3').verbose();

//---------------------------------------------------------------------

function getBooks(author, callback) {
   let db = new sqlite3.Database('penny.sqlite', sqlite3.OPEN_READONLY);
   let stmt =
      'SELECT isbn, author, title FROM books WHERE author LIKE ?';
   db.all(stmt, [author + '%'],
      (err, data) => {db.close(); callback(err, data);}
   );
}

//---------------------------------------------------------------------

// For testing:

function writeRows(err, rows) {
   console.log('in writeRows');
   if (err)
      console.log(err.message);
   else
      console.log(rows);
}

function main() {
   try {
      getBooks('Kernighan', writeRows);
      getBooks('Sedgewick', writeRows);
      getBooks('Dondero', writeRows);
   }
   catch (err) {
      console.log(err.message);
   };
}

//---------------------------------------------------------------------

if (require.main === module)
   main();

//---------------------------------------------------------------------

module.exports = { getBooks };
```

PennyExpress/header.html (Page 1 of 1)

```
<hr>
Good {{ampm}} and welcome to <strong>Penny.com</strong>
<hr>
```

PennyExpress/footer.html (Page 1 of 1)

```
1: <hr>
2: Today is {{currentTime}}.<br>
3: Created by <a href="https://www.cs.princeton.edu/~rdondero">
4: Bob Dondero</a>
5: <hr>
```

PennyExpress/index.html (Page 1 of 1)

```
1: <!DOCTYPE html>
2:  <html>
3:    <head>
4:       <title>Penny.com</title>
5:       </head>
6:    <body>
7:       {{>header}}
8:       <br>
9:       Click here to <a href="/searchform">begin</a>.<br>
10:      <br>
11:      {{>footer}}
12:   </body>
13: </html>
```

PennyExpress/searchform.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      {{>header}}
      <h1>Author Search</h1>
      <form action="/searchresults" method="get">
         Please enter an author name:
         <input type="text" name="author" autofocus>
         <input type="submit" value="Go">
      </form>
      <br>
      <br>
      <strong>Previous author search: {{ prevAuthor }}</strong>
      <br>
      <br>
      {{>footer}}
   </body>
</html>
```


PennyExpress/searchresults.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>Penny.com</title>
   </head>
   <body>
      {{>header}}
      <h1>Author Search Results</h1>
      <h2>Books by {{author}}:</h2>
      {{#books}}
         {{isbn}}: <strong>{{author}}</strong>: {{title}}<br>
      {{/books}}
      {{^books}}
         (None)
      {{/books}}
      <br>
      <br>
      Click here to do another
      <a href="/searchform">author search</a>.
      <br>
      <br>
      <br>
      <br>
       {{>footer}}
   </body>
</html>
```

PennyExpress/penny.js (Page 1 of 2)

```
//---------------------------------------------------------------------
// penny.js
// Author: Bob Dondero
//---------------------------------------------------------------------

// Omit handling of server-side errors.

const express = require('express');
const mustacheExpress = require('mustache-express');
const database = require('./database');

//---------------------------------------------------------------------

function getAmPm() {
   let hours = new Date().getHours();
   return (hours < 12) ? 'morning' : 'afternoon';
}

function getCurrentTime() {
   return new Date();
}

//---------------------------------------------------------------------

function parseCookies(req) {
   let cookies = {};
   let cookieHeader = req.headers.cookie;
   if (! cookieHeader)
      return cookies;
   let cookieStrs = cookieHeader.split('; ');
   for (let cookieStr of cookieStrs) {
      let nameValue = cookieStr.split('=');
      let name = nameValue[0];
      let value = nameValue[1];
      cookies[name] = value;
   }
   return cookies;
}

//---------------------------------------------------------------------

function index(req, res) {
   process.stdout.write(req.originalUrl + '\n');

   let replacements = {
      ampm: getAmPm(),
      currentTime: getCurrentTime()
   };
   res.render('index', replacements);
}

//---------------------------------------------------------------------

function searchForm(req, res) {
   process.stdout.write(req.originalUrl + '\n');

   let cookies = parseCookies(req);
   let prevAuthor = cookies.prevAuthor;
   if (! prevAuthor)
      prevAuthor = '(None)';

   let replacements = {
      ampm: getAmPm(),
      currentTime: getCurrentTime(),
      prevAuthor: prevAuthor
   };
   res.render('searchform', replacements);
}

//---------------------------------------------------------------------

function searchResults(req, res) {
   function handleBooks(err, books) {  // Error handling omitted
      res.cookie('prevAuthor', author);
      let replacements = {
         ampm: getAmPm(),
         currentTime: getCurrentTime(),
         author: author,
         books: books
      };
      res.render('searchresults', replacements);
   }

   process.stdout.write(req.originalUrl + '\n');

   let author = req.query.author;
   if (! author)
      author = '';
   author = author.trim();

   if (author === '') {
      res.cookie('prevAuthor', '(None)');
      let replacements = {
         ampm: getAmPm(),
         currentTime: getCurrentTime(),
         author: '(None)',
         books: []
      };
      res.render('searchresults', replacements);
   }
   else
      database.getBooks(author, handleBooks);
}

//---------------------------------------------------------------------

let app = express();
app.engine('html', mustacheExpress());
app.set('view engine', 'html');
app.set('views', '.');

app.get('/', index);
app.get('/index', index);
app.get('/searchform', searchForm);
app.get('/searchresults', searchResults);

module.exports = { app };
```