

Server-Side Options

Copyright © 2024 by
Robert M. Dondero, Ph.D
Princeton University

Objectives

- We will cover:
 - Some server-side options
 - Java: Servlets
 - Java: The Spring web app framework
 - JavaScript: The Express web app framework

Agenda

- **Prelim: packages**
- Prelim: library management
- Java server-side pgmming
- Java server-side pgmming: Servlets
- Java server-side pgmming: Spring
- JavaScript server-side pgmming
- JavaScript server-side pgmming: Express

Packages

- **Python packages**
 - A ***package*** is a named group of modules (and other packages)
 - A *module* is stored in a *file*
 - A *package* is stored in a *directory*

Packages

- **Python packages (cont.)**
 - **Using a Python package**

```
import intmath.euclid
...
intmath.euclid.gcd(x, y)
...
```

Packages

- **Python packages (cont.)**
 - **Defining a Python package**

```
intmath
  __init__.py
  euclid.py
...
```

Packages

- **Java packages**
 - A **package** is a named group of classes/interfaces (and other packages)
 - A *class/interface* is stored in a file
 - A *package* is stored in a *directory*

Packages

- **Java packages (cont.)**
 - **Using a Java package**

```
...  
java.lang.Math.sqrt(x) ;  
...
```

```
import java.lang.Math ;  
...  
Math.sqrt(x) ;  
...
```


Packages

- **Java packages (cont.)**
 - **Defining a Java package**

```
// Book.java

package edu.princeton.penny;

// All names defined subsequently are
// in the edu.princeton.penny package.

public class Book { ... }
```

Defines class `edu.princeton.penny.Book`

`Book.java` **must be in** `edu/princeton/penny` directory

Agenda

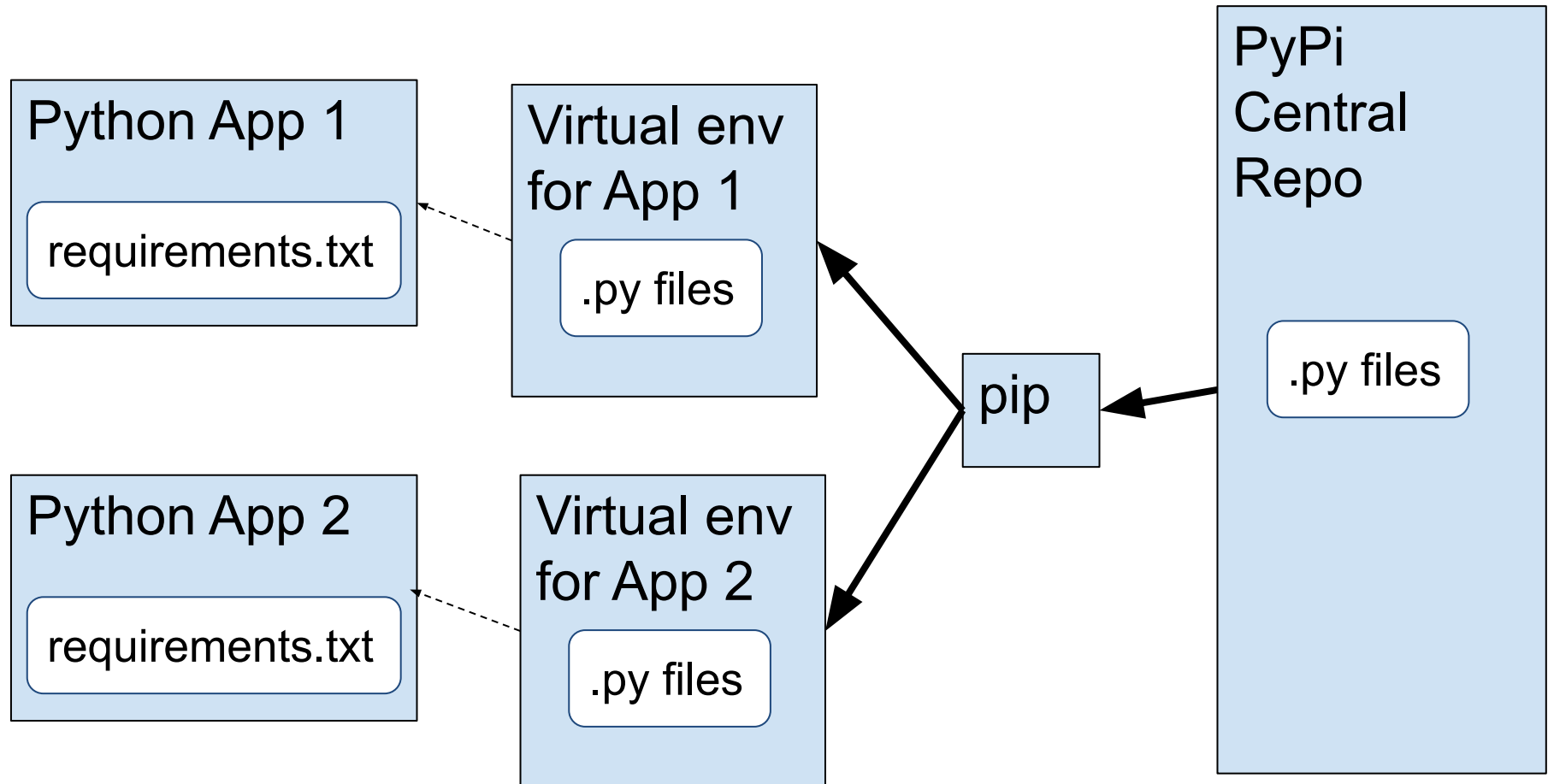
- Prelim: packages
- **Prelim: library management**
- Java server-side pgmming
- Java server-side pgmming: Servlets
- Java server-side pgmming: Spring
- JavaScript server-side pgmming
- JavaScript server-side pgmming: Express

Library Management

- **Python** library management
 - **pip** and **PyPi** central repo

Library Management

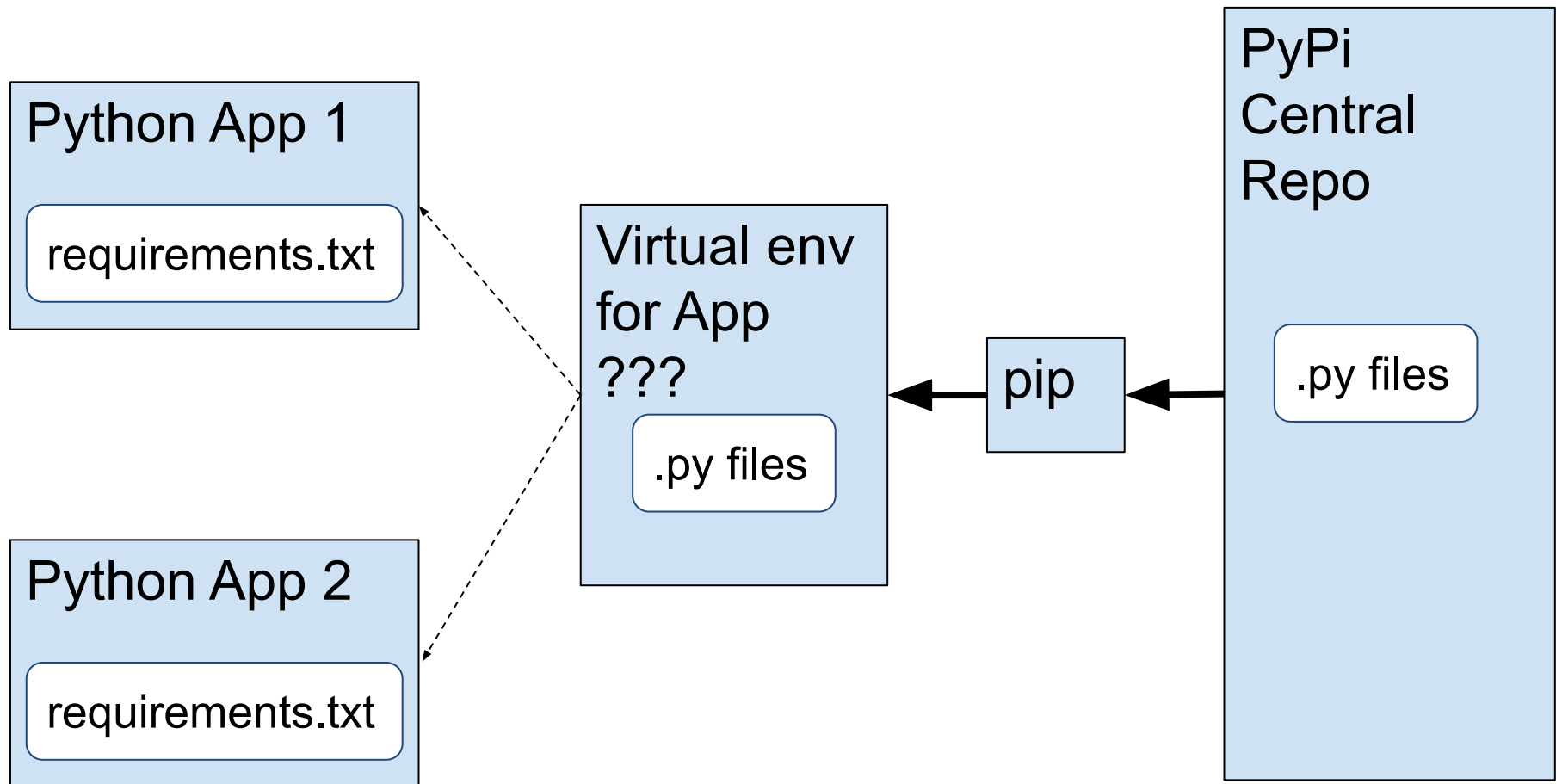
Python library management (cont.)



(Activate virtual env for App X)
`pip install -r requirements.txt`

Library Management

Python library management (cont.)

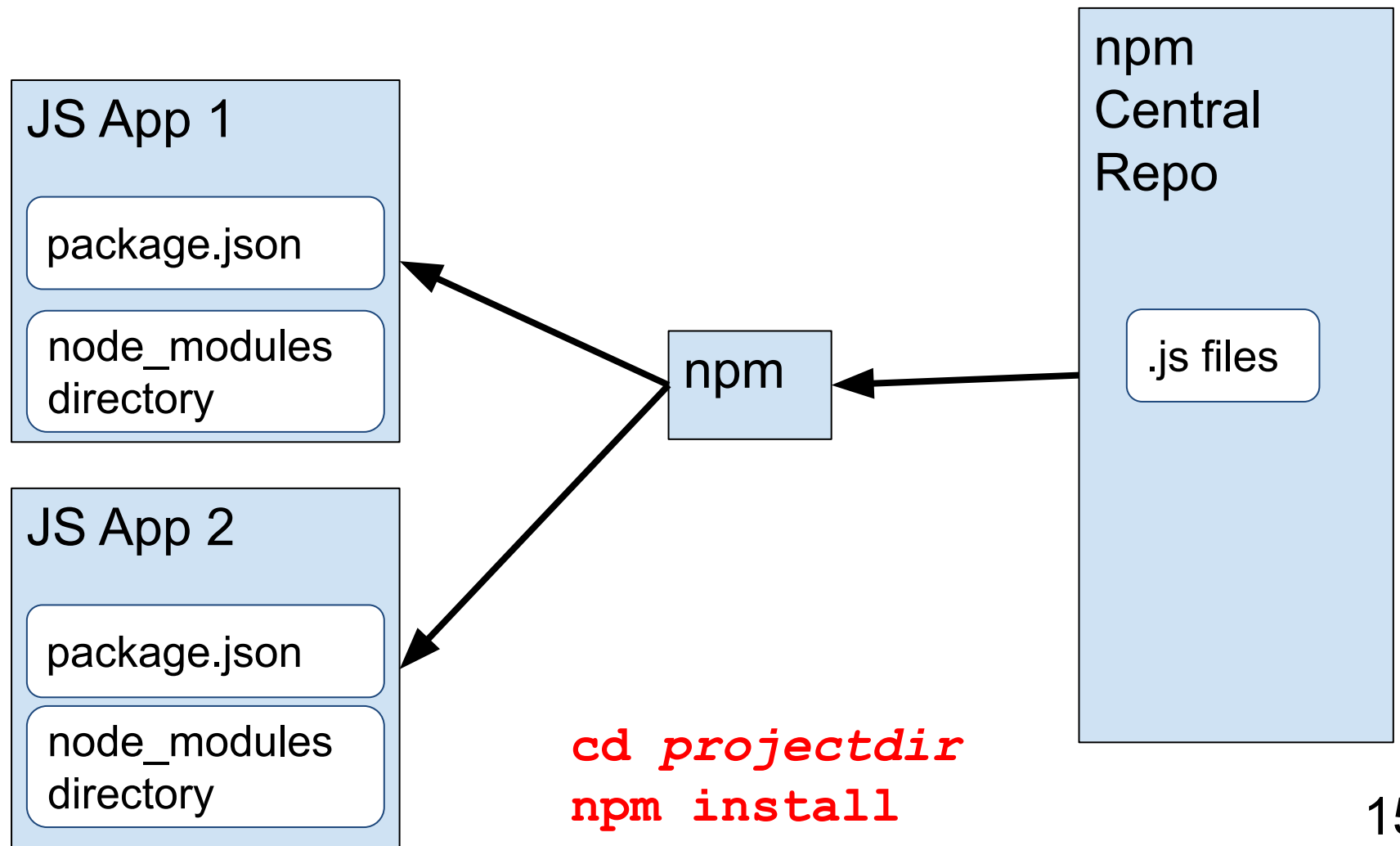


Library Management

- **JavaScript** library management
 - **npm** and **npm central repo**

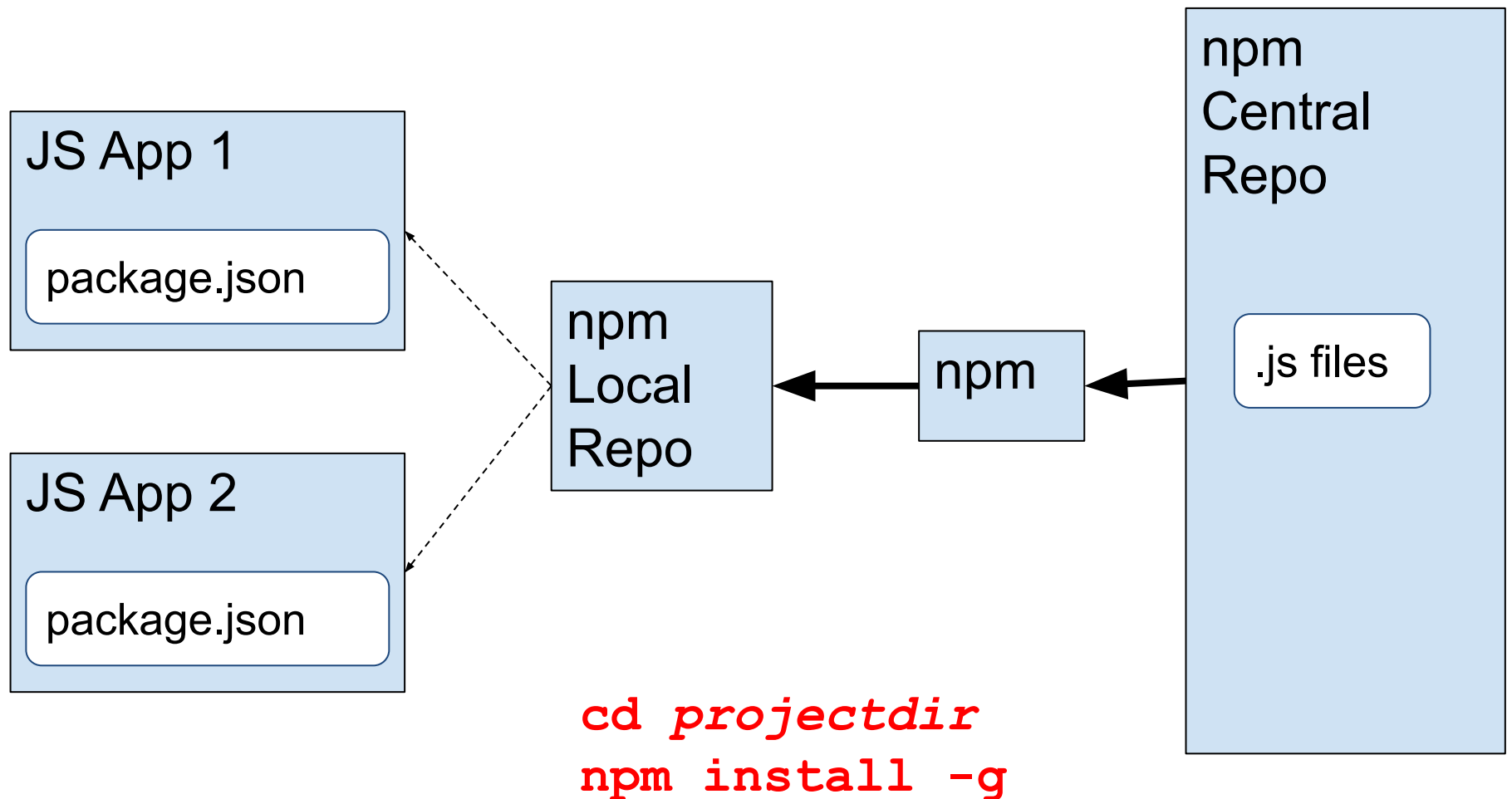
Library Management

JavaScript library management (cont.)



Library Management

JavaScript library management (cont.)

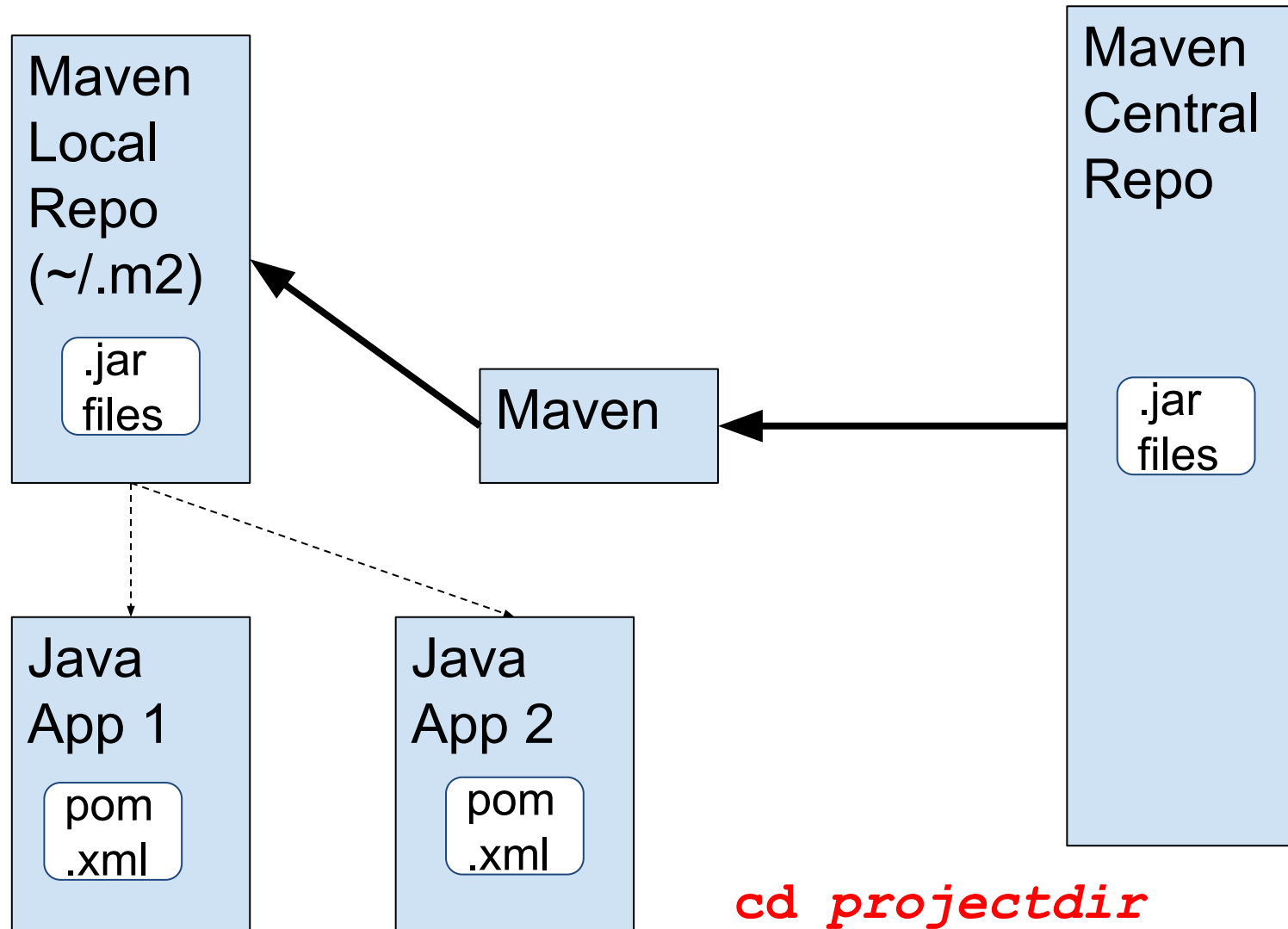


Library Management

- **Java** library management
 - *Maven* and central Maven repo
- Maven is a:
 - Library management tool
 - Build tool
 - C : make
 - Java : Maven

Library Management

Java library management (cont.)



cd projectdir
mvn compile

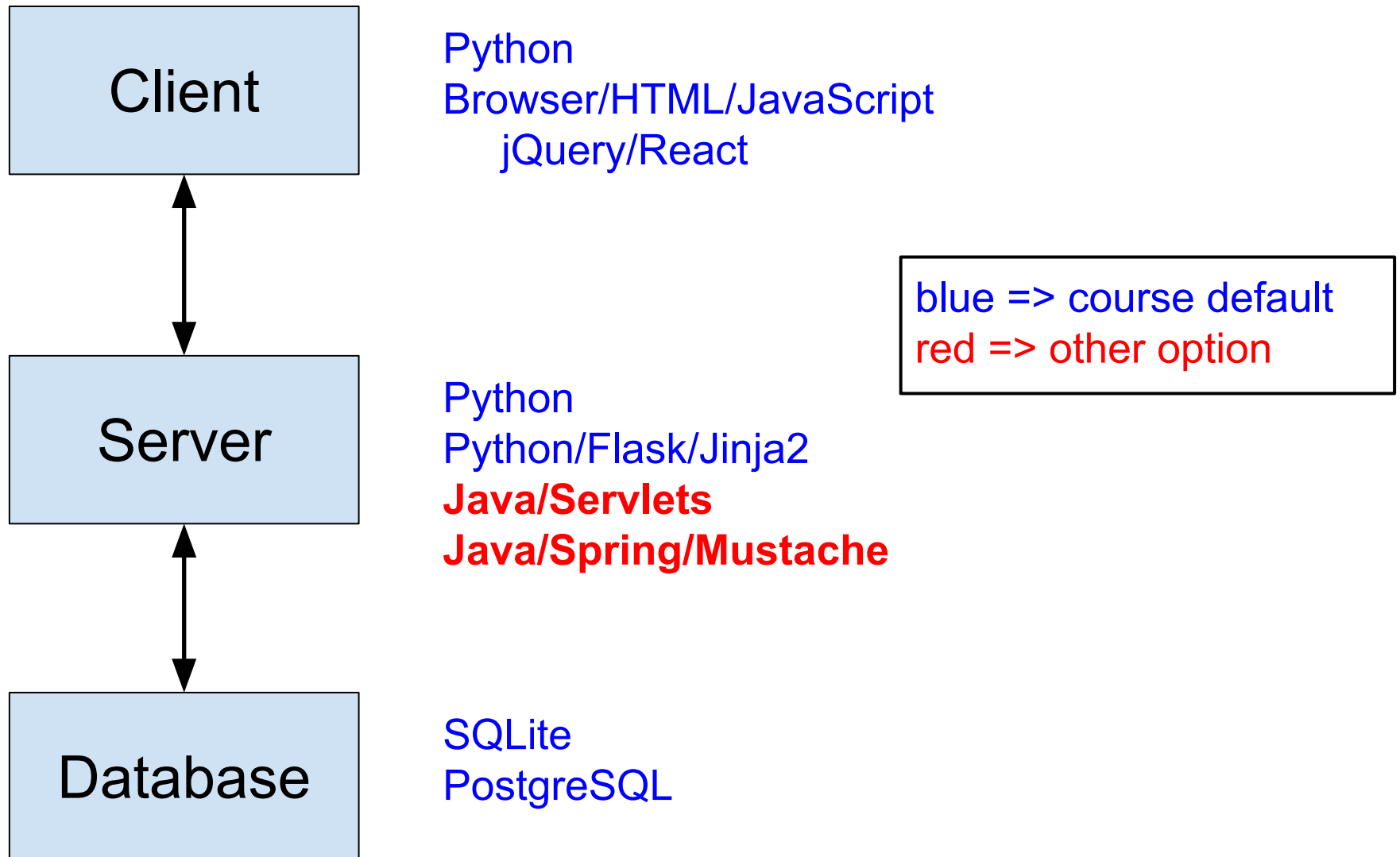
Library Management

- Installing Maven
 - Mac:
 - `brew install maven`
 - MS Windows
 - <https://maven.apache.org/guides/getting-started/windows-prerequisites.html>
 - Linux
 - Use your package mgr

Agenda

- Prelim: packages
- Prelim: library management
- **Java server-side pgmming**
- Java server-side pgmming: Servlets
- Java server-side pgmming: Spring
- JavaScript server-side pgmming
- JavaScript server-side pgmming: Express

Java Server-Side Web Pgmming



Java Server-Side Web Pgmming

- **Question:** Why study Java server-side web programming?
- **Answer 1:** Help you better understand server-side web programming
- **Answer 2:** Give you a sense of how Java is used in the “real world”

Java Server-Side Web Pgmming

- **Question:** Why study Java server-side web programming (cont.)?
- **Answer 3:** Show realistic use of some advanced features of Java
- **Answer 4:** Popular and important...

Java Server-Side Web Pgmming

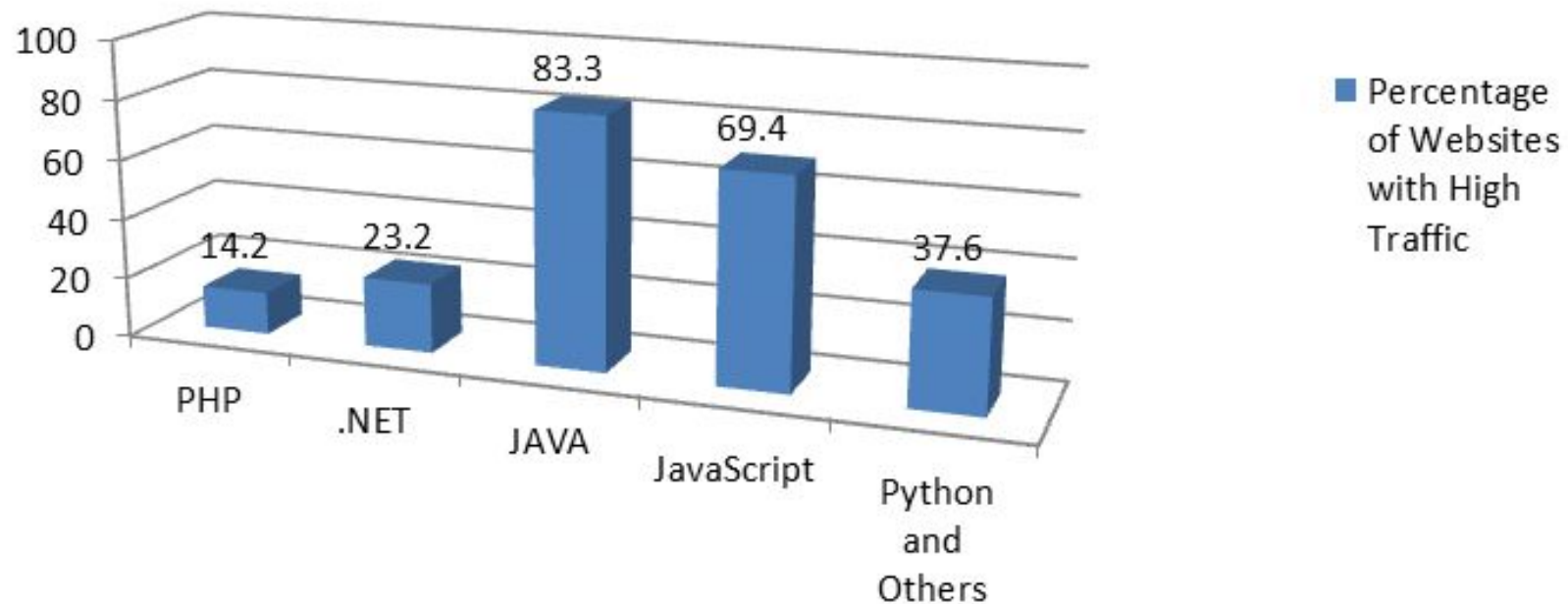
Website	Front End	Back End
Google	JavaScript	C, C++, Go, Java , Python, Node
Facebook	JavaScript TypeScript Flow	Hack, HHVM, Python, C++, Java , Erlang, D, Haskell
YouTube	JavaScript TypeScript	Python, C, C++, Java , Go
Yahoo	JavaScript	PHP
Etsy	JavaScript	PHP
Amazon	JavaScript	Java , C++, Perl
Wikipedia	JavaScript	PHP
Fandom	JavaScript	PHP
X	JavaScript	C++, Java , Scala, Ruby
Bing	JavaScript	C++, C#
eBay	JavaScript	Java , JavaScript, Scala

<https://en.wikipedia.org/wiki/>

Programming_languages_used_in_most_popular_websites (July 2024)

Java Server-Side Web Pgmming

Programming Language vs High Traffic Websites



-- <http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/>

Agenda

- Prelim: packages
- Prelim: library management
- Java server-side pgmming
- **Java server-side pgmming: Servlets**
- Java server-side pgmming: Spring
- JavaScript server-side pgmming
- JavaScript server-side pgmming: Express

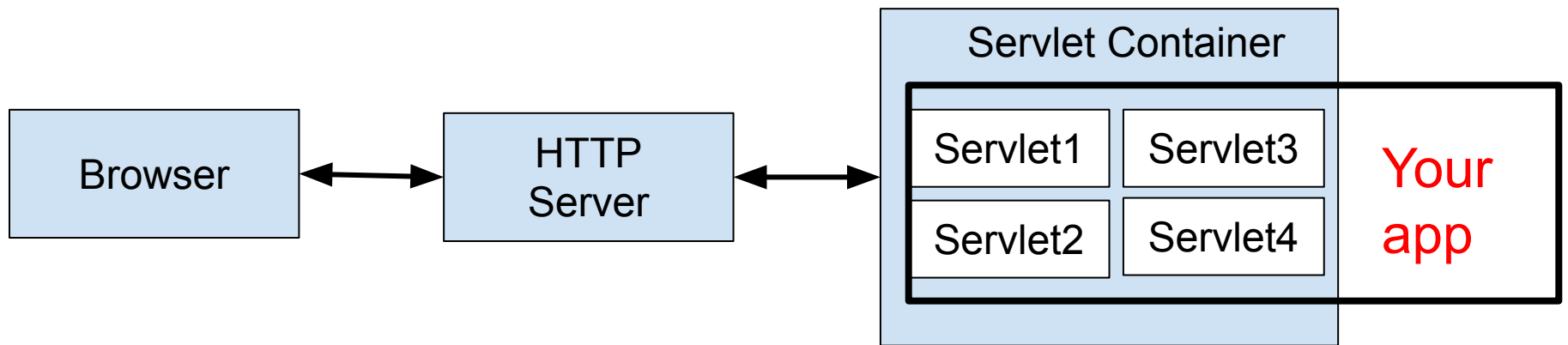
Servlet Programming

- **Question:** How does one do web programming in Java?
- **Answer 1:** Use Servlets
- **Answer 2:** Use a Java web app framework

Servlet Programming

- **Servlets**
 - “A servlet is a Java technology-based Web component, managed by a **container**, that generates dynamic content.”
 - Servlet specification at:
 - http://download.oracle.com/.../servlet-3_0-final-spec.pdf

Servlet Programming

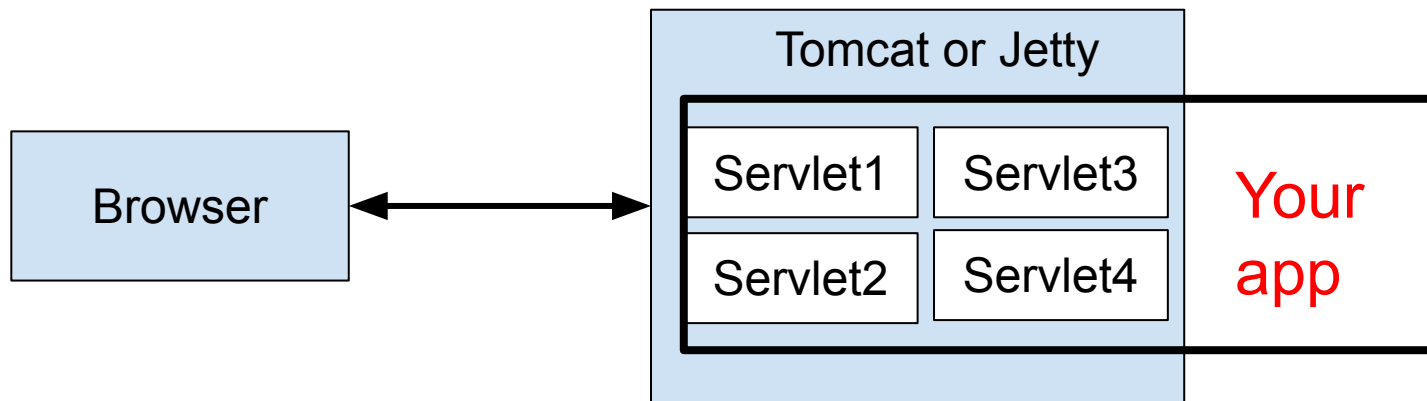


Popular servlet containers:

Apache Tomcat

Eclipse Jetty

Servlet Programming



Tomcat or Jetty can be both servlet container and HTTP server

We'll use that configuration

We'll use Jetty for Servlets, Tomcat for Spring

Servlet Programming

- **Python:**
 - *WSGI specification* connects HTTP server and your Python app
- **Java:**
 - *Servlet specification* connects HTTP server and your Java app

Servlet Programming

- See **PennyServlets** app

On Linux or Mac:

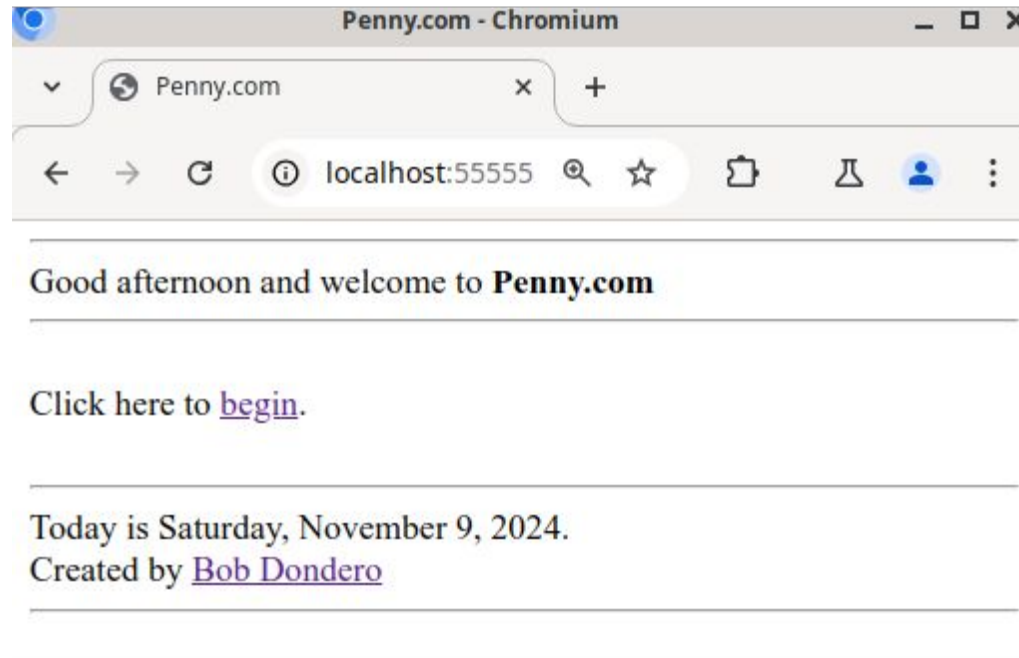
```
$ ./runserver 55555
```

On MS Windows:

```
C:\>runserver.bat 55555
```

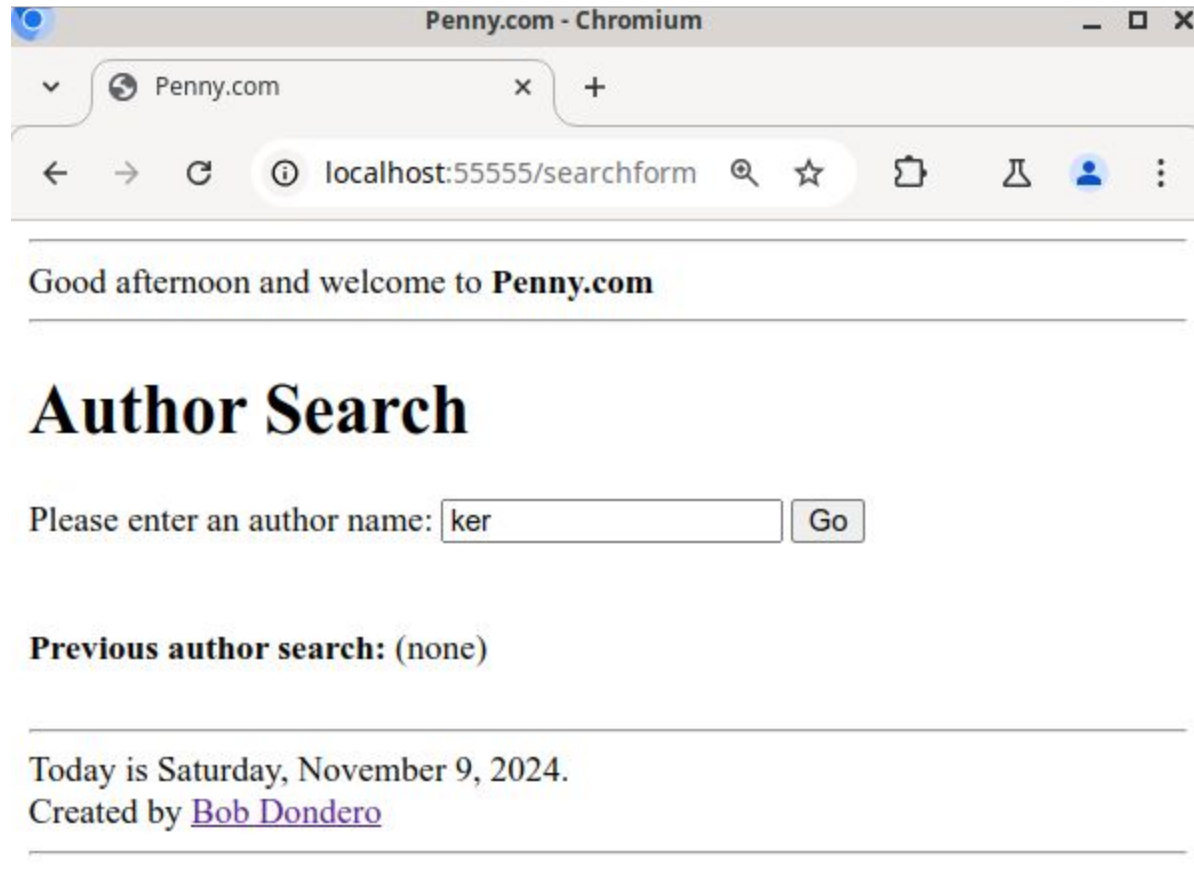

Servlet Programming

- See **PennyServlets** app (cont.)



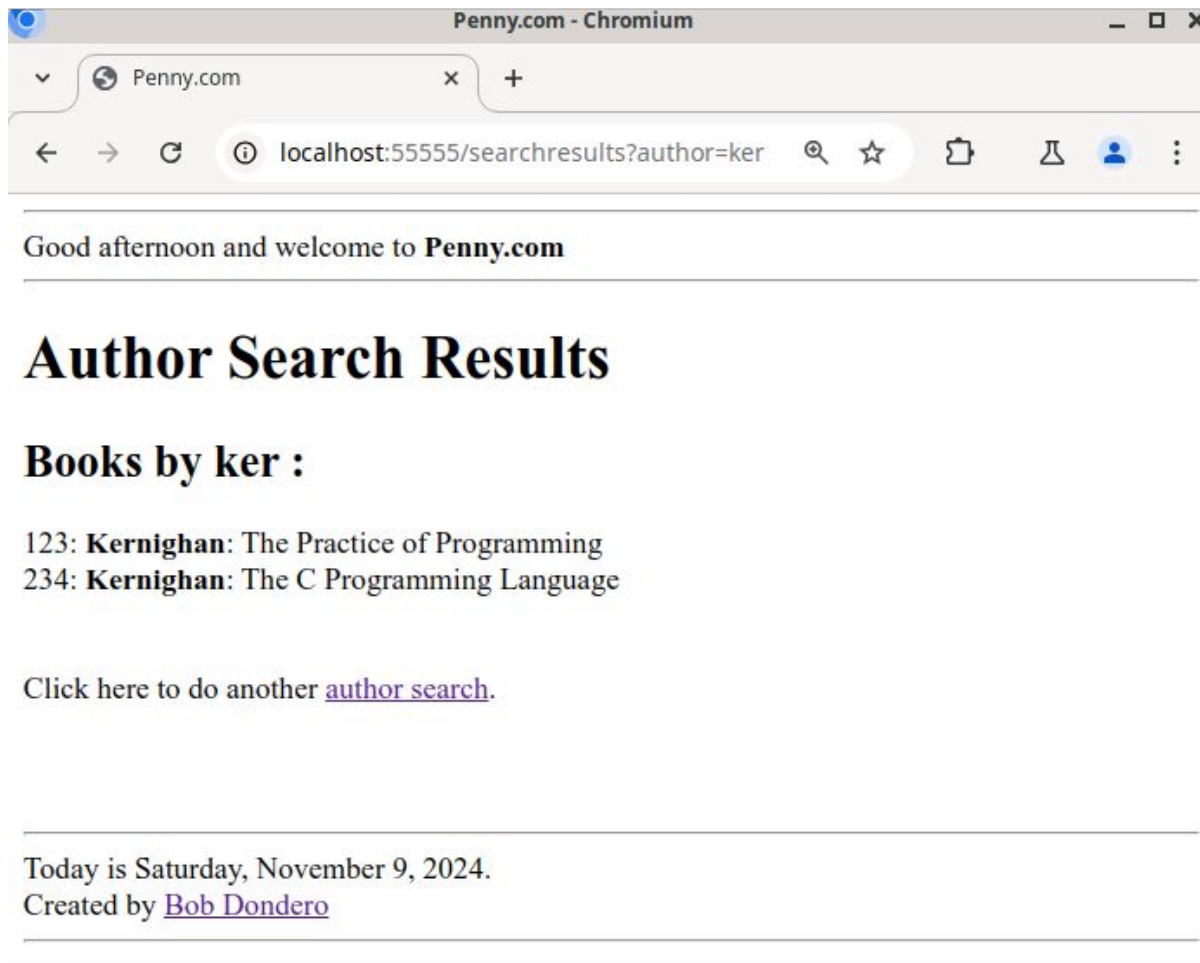
Servlet Programming

- See **PennyServlets** app (cont.)



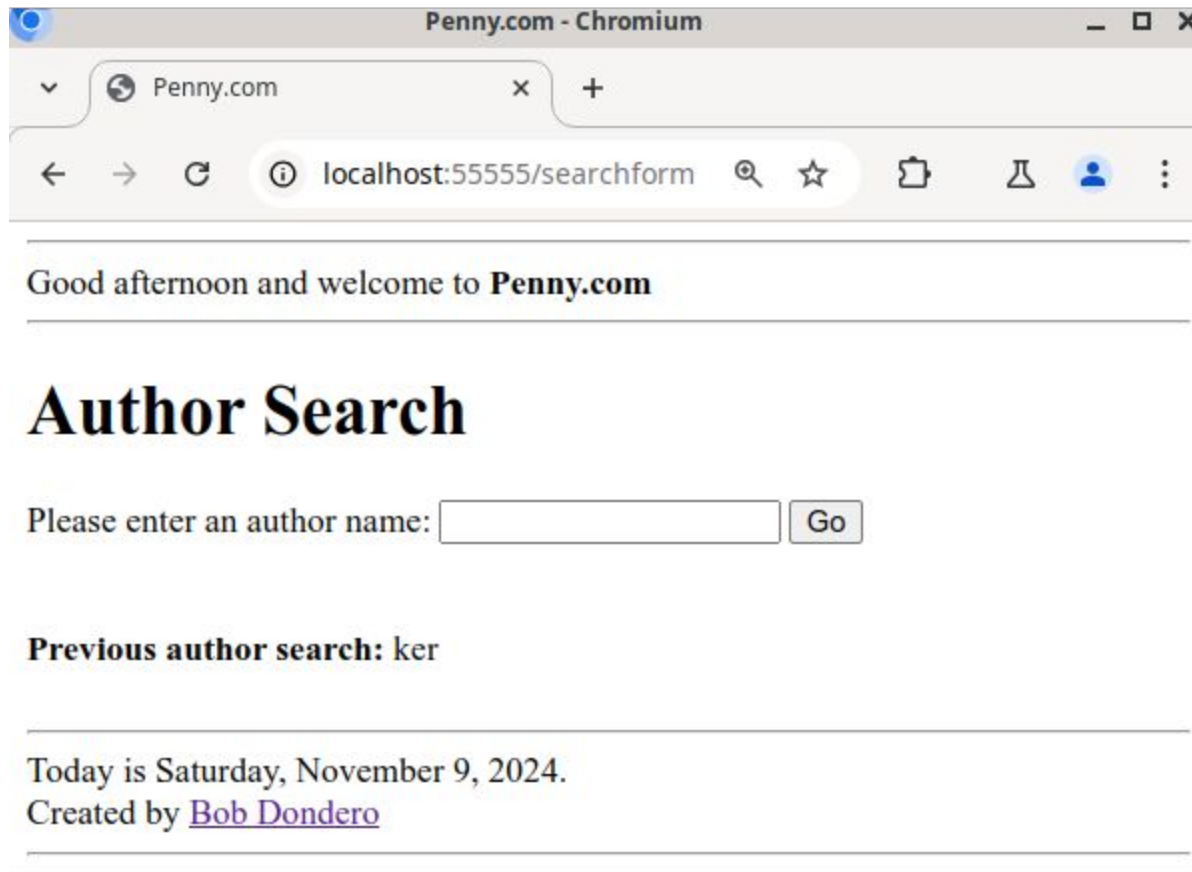
Servlet Programming

- See **PennyServlets** app (cont.)



Servlet Programming

- See **PennyServlets** app (cont.)



The screenshot shows a web browser window titled "Penny.com - Chromium". The address bar displays "localhost:55555/searchform". The page content includes a welcome message, a search form, and a footer with the date and author information.

Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name:

Previous author search: ker

Today is Saturday, November 9, 2024.
Created by [Bob Dondero](#)

Servlet Programming

```
PennyServlets/  
  pom.xml  
  runserver  
  runserver.bat  
  src/  
    main/  
      java/  
        edu/  
          princeton/  
            penny/  
              Book.java  
              Common.java  
              Database.java  
              IndexServlet.java  
              Penny.java  
              SearchFormServlet.java  
              SearchResultsServlet.java  
              Util.java  
    resources/  
      penny.sql  
      penny.sqlite
```

Servlet Programming

- See **PennyServlets** app (cont.)
 - **runserver, runserver.bat**
 - **pom.xml**
 - **penny.sql, penny.sqlite**
 - **Book.java, Database.java**
 - **Common.java, Util.java**
 - **IndexServlet.java**
 - **SearchFormServlet.java**
 - **SearchResultsServlet.java**
 - **Penny.java**

PennyServlets/runserver (Page 1 of 1)

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

PennyServlets/runserver.bat (Page 1 of 1)

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

PennyServlets/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:
28:    <dependency>
29:      <groupId>org.eclipse.jetty</groupId>
30:      <artifactId>jetty-server</artifactId>
31:      <version>9.4.18.v20190429</version>
32:    </dependency>
33:
34:    <dependency>
35:      <groupId>org.eclipse.jetty</groupId>
36:      <artifactId>jetty-webapp</artifactId>
37:      <version>9.4.18.v20190429</version>
38:    </dependency>
39:
40:    <dependency>
41:      <groupId>org.eclipse.jetty.websocket</groupId>
42:      <artifactId>websocket-server</artifactId>
43:      <version>9.4.18.v20190429</version>
44:    </dependency>
45:
46:    <dependency>
47:      <groupId>org.eclipse.jetty.websocket</groupId>
48:      <artifactId>websocket-servlet</artifactId>
49:      <version>9.4.18.v20190429</version>
50:    </dependency>
51:    <dependency>
52:      <groupId>org.apache.commons</groupId>
53:      <artifactId>commons-text</artifactId>
54:      <version>1.9</version>
55:    </dependency>
56:
57:  </dependencies>
58:
59: </project>

```

blank (Page 1 of 1)

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)

```

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) PennyServlets/src/main/java/edu/princeton/penny/Util.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: }

```

```

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 2) PennyServlets/src/main/java/edu/princeton/penny/SearchFormServlet.java (Page 2 of 2)

```

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: }
```

```

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: }
```

```

PennyServlets/src/main/java/edu/princeton/penny/SearchResultsServlet.java PennyServlets/src/main/java/edu/princeton/penny/SearchResultsServlet.java
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:
67:         pw.println("<h1>Author Search Results</h1>");
68:         pw.println("<h2>Books by ");
69:         pw.println(StringEscapeUtils.escapeHtml4(prevAuthor));
70:         pw.println("</h2>");
71:
72:         if (books.size() == 0)
73:             pw.println("(None)<br>");
74:         else
75:             for (Book book : books)
76:                 pw.println(
77:                     StringEscapeUtils.escapeHtml4(book.getIsbn())
78:                     + ": "
79:                     + "<strong>"
80:                     + StringEscapeUtils.escapeHtml4(book.getAuthor())
81:                     + "</strong>: "
82:                     + StringEscapeUtils.escapeHtml4(book.getTitle())
83:                     + "<br>");
84:         pw.println("<br>");
85:         pw.println("<br>");
86:         pw.println("<a href=\"/searchform\">Click here to do another ");
87:         pw.println("<a href=\"/searchform\">author search</a>.</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) blank (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: }

```

1: This page is intentionally blank.

Agenda

- Prelim: packages
- Prelim: library management
- Java server-side pgmming
- Java server-side pgmming: Servlets
- **Java server-side pgmming: Spring**
- JavaScript server-side pgmming
- JavaScript server-side pgmming: Express

Spring

- **Question:** How does one do web programming in Java?
- **Answer 1:** Use Servlets
- **Answer 2:** Use a Java web app framework

Spring



Rod
Johnson

Spring

- ***Spring Boot***

- Wikipedia: “Spring Boot is a convention-over-configuration extension for the **Spring** Java platform intended to help minimize configuration concerns while creating Spring-based applications.”

Spring

- Why study Spring / Spring Boot?
 - (Instead of some other framework)

Spring

Some popular server-side web app frameworks:

Rank	Framework	GitHub Stars
1	Laravel (PHP)	74.9K
2	Django (Python)	73.4K
3	Flask (Python)	64.5K
4	Express (JavaScript)	62.0K
5	Ruby on Rails (Ruby)	53.6K
6	Spring (Java)	53.3K
7	Meteor (JavaScript)	43.8K
8	Symfony (PHP)	28.7K

According to GitHub as of 10/4/23

Spring

- See **PennySpring** app

On Linux or Mac:

```
$ ./runserver
```

On MS Windows:

```
C:\>runserver.bat
```

[All pages identical to those of PennySpring app]

Spring

- See **PennySpring** app (cont.)
 - Browse to <https://start.spring.io/>
 - Complete as shown on following slides, and click *Generate* to download `PennySpringBoot.zip`

Spring

Spring Initializr — Mozilla Firefox

COS 333, Spring 2024: Gene X Spring Initializr

https://start.spring.io

Import bookmarks... Getting Started CS Department Web ... COS217 COS333 IntroCsBook IndepWork Princeton Health Linux Mint Forums Blog Most Visited News Other Bookmarks

≡ spring initializr

⚙️ 🌙

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ **Java** ☐ Kotlin ☐ Groovy

☒ **Maven**

Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M1) ☐ 3.3.3 (SNAPSHOT) ☒ **3.3.2**

☐ 3.2.9 (SNAPSHOT) ☐ 3.2.8

Project Metadata

Group

Artifact

Name

Description

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Mustache TEMPLATE ENGINES

Logic-less templates for both web and standalone environments. There are no if statements, else clauses, or for loops. Instead there are only tags.

🐙

GENERATE CTRL + G EXPLORE CTRL + SPACE SHARE...

Spring

Spring Initializr — Mozilla Firefox

COS 333, Spring 2024: Gene X Spring Initializr

https://start.spring.io

Import bookmarks... Getting Started CS Department Web ... COS217 COS333 IntroCsBook IndepWork Princeton Health Linux Mint Forums Blog Most Visited News Other Bookmarks

☰

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.4.0 (SNAPSHOT) ☐ 3.4.0 (M1) ☐ 3.3.3 (SNAPSHOT) ☒ 3.3.2

☐ 3.2.9 (SNAPSHOT) ☐ 3.2.8

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Mustache TEMPLATE ENGINES

Logic-less templates for both web and standalone environments. There are no if statements, else clauses, or for loops. Instead there are only tags.

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

Spring

- See **PennySpring** app (cont.)
 - Unzip `PennySpringBoot.zip` to generate this directory structure:

Spring

```
demo
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
  .gitignore
  src/
    main/
      java/
        edu/
          princeton/
            demo/
              DemoApplication.java
    resources/
      static/
      templates/
      application.properties
    test/
      ...
  .mvn/
  ...
```

Spring

- See **PennySpring** app (cont.)
 - Edit to create this directory structure:

Spring

PennySpring

```
HELP.md
mvnw
mvnw.cmd
pom.xml
.gitignore
src/
```

```
    main/
```

```
        java/
```

```
            edu/
```

```
                princeton/
```

```
                    penny/
```

```
                        Book.java
```

```
                        Database.java
```

```
                        PennyApplication.java
```

```
                        PennyController.java
```

```
resources/
```

```
resources/
    static/
        templates/
```

```
            footer.html
```

```
            header.html
```

```
            index.html
```

```
            searchform.html
```

```
            searchresults.html
```

```
application.properties
```

```
penny.sql
```

```
penny.sqlite
```

```
test/
```

```
    ...
```

```
.mvn/
```

```
    ...
```

Spring

- See **PennySpring** app (cont.)
 - runserver, runserver.bat
 - pom.xml
 - application.properties
 - header.html, footer.html, index.html, searchform.html, searchresults.html
 - penny.sql, penny.sqlite
 - Book.java
 - Database.java
 - **PennyApplication.java**
 - **PennyController.java**

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)

```

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>3.3.1</version>
11:    <relativePath/> <!-- lookup parent from repository -->
12:  </parent>
13:  <groupId>edu.princeton</groupId>
14:  <artifactId>demo</artifactId>
15:  <version>0.0.1-SNAPSHOT</version>
16:  <name>demo</name>
17:  <description>Penny</description>
18:  <url/>
19:  <licenses>
20:    <license/>
21:  </licenses>
22:  <developers>
23:    <developer/>
24:  </developers>
25:  <scm>
26:    <connection/>
27:    <developerConnection/>
28:    <tag/>
29:    <url/>
30:  </scm>
31:  <properties>
32:    <java.version>21</java.version>
33:  </properties>
34:  <dependencies>
35:    <dependency>
36:      <groupId>org.springframework.boot</groupId>
37:      <artifactId>spring-boot-starter-mustache</artifactId>
38:    </dependency>
39:    <dependency>
40:      <groupId>org.springframework.boot</groupId>
41:      <artifactId>spring-boot-starter-web</artifactId>
42:    </dependency>
43:    <dependency>
44:      <groupId>org.springframework.boot</groupId>
45:      <artifactId>spring-boot-starter-test</artifactId>
46:      <scope>test</scope>
47:    </dependency>
48:    <dependency>
49:      <groupId>org.xerial</groupId>
50:      <artifactId>sqlite-jdbc</artifactId>
51:      <version>3.34.0</version>
52:    </dependency>
53:    <dependency>
54:      <groupId>org.apache.commons</groupId>
55:      <artifactId>commons-text</artifactId>
56:      <version>1.9</version>
57:    </dependency>
58:  </dependencies>
59:  <build>
60:    <plugins>
61:      <plugin>
62:        <groupId>org.springframework.boot</groupId>
63:        <artifactId>spring-boot-maven-plugin</artifactId>
64:      </plugin>
65:    </plugins>

```

PennySpring/pom.xml (Page 2 of 2)

```

66:   </build>
67: </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/~rdontero">
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) PennySpring/src/main/java/edu/princeton/penny/PennyApplication.java (Pag

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <{>header</>
8:     <h1>Author Search Results</h1>
9:     <h2>Books by <{author}</>:</h2>
10:    <{#books}</>
11:    <{isbn}</>: <strong><{author}</></strong>: <{title}</><br>
12:    <{/books}</>
13:    <{^books}</>
14:    <(None)</>
15:    <{/books}</>
16:    <br>
17:    <br>
18:    Click here to do another
19:    <a href="/searchform">author search</a>.
20:    <br>
21:    <br>
22:    <br>
23:    <br>
24:    <{>footer</>
25:  </body>
26: </html>

```

```

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: }

```

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

```

```

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: }

```

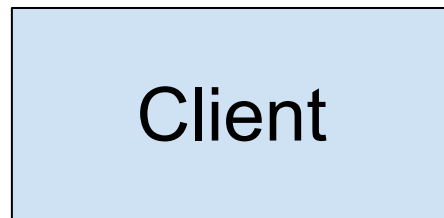
Spring

- There is much more to Spring and Mustache
 - Spring website:
 - <https://spring.io/>
 - Spring Boot website:
 - <https://spring.io/projects/spring-boot>
 - Mustache website:
 - <https://www.baeldung.com/mustache>

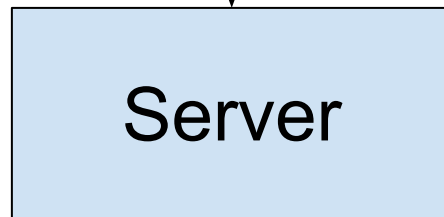
Agenda

- Prelim: packages
- Prelim: library management
- Java server-side pgmming
- Java server-side pgmming: Servlets
- Java server-side pgmming: Spring
- **JavaScript server-side pgmming**
- JavaScript server-side pgmming: Express

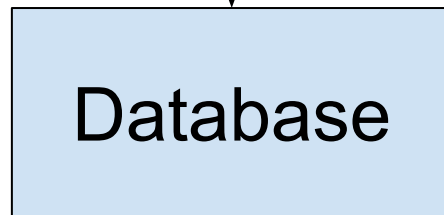
JavaScript Server Web Pgmming



Python
Browser/HTML/JavaScript
jQuery/React



Python
Python/Flask/Jinja2
Java/Servlets
Java/Spring/Mustache
JavaScript/Express/Mustache



SQLite
PostgreSQL

blue => course default
red => other option

JavaScript Server Web Pgmming

- Why study JavaScript server-side pgmming?

JavaScript Server Web Pgmming

Website	Front End	Back End
Google	JavaScript	C, C++, Go, Java, Python, Node
Facebook	JavaScript TypeScript Flow	Hack, HHVM, Python, C++, Java, Erlang, D, Haskell
YouTube	JavaScript TypeScript	Python, C, C++, Java, Go
Yahoo	JavaScript	PHP
Etsy	JavaScript	PHP
Amazon	JavaScript	Java, C++, Perl
Wikipedia	JavaScript	PHP
Fandom	JavaScript	PHP
X	JavaScript	C++, Java, Scala, Ruby
Bing	JavaScript	C++, C#
eBay	JavaScript	Java, JavaScript, Scala

<https://en.wikipedia.org/wiki/>

Programming_languages_used_in_most_popular_websites (July 2024)

JavaScript Server Web Pgmming

- Why study server-side JavaScript pgmming (cont.)?
 - Introduces additional aspects of JavaScript
 - Database programming
 - Becoming more popular

Agenda

- Prelim: packages
- Prelim: library management
- Java server-side pgmming
- Java server-side pgmming: Servlets
- Java server-side pgmming: Spring
- JavaScript server-side pgmming
- **JavaScript server-side pgmming:
Express**

Express

- Why study server-side JavaScript programming using **Express** (instead of some other framework)?

Express

Some popular server-side web app frameworks:

Rank	Framework	GitHub Stars
1	Laravel (PHP)	74.9K
2	Django (Python)	73.4K
3	Flask (Python)	64.5K
4	Express (JavaScript)	62.0K
5	Ruby on Rails (Ruby)	53.6K
6	Spring (Java)	53.3K
7	Meteor (JavaScript)	43.8K
8	Symfony (PHP)	28.7K

According to GitHub as of 10/4/23

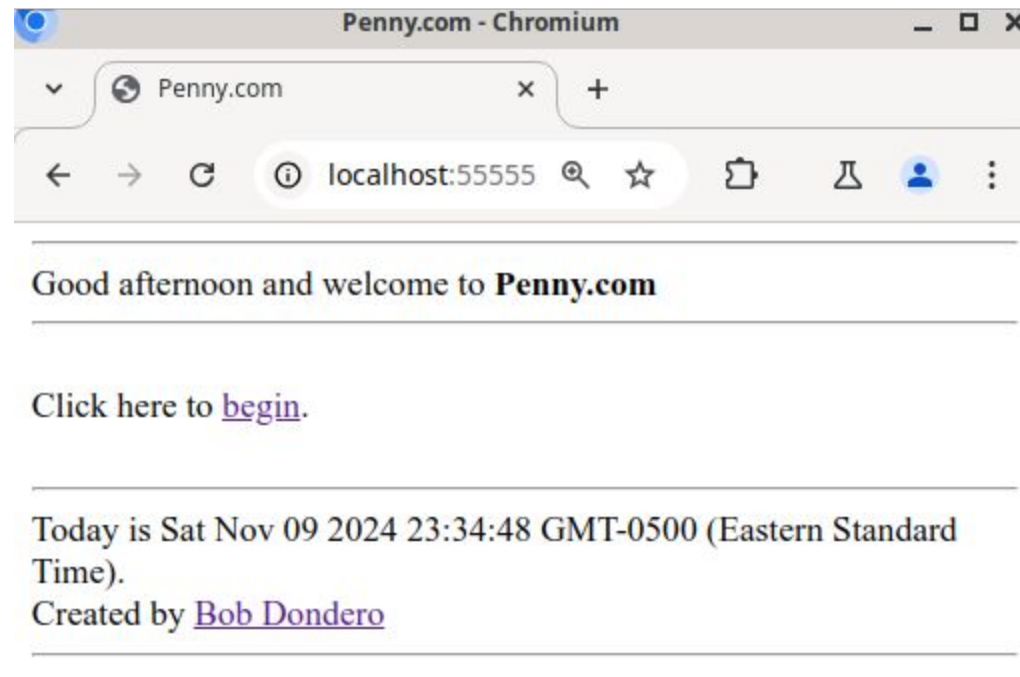
Express

- See **PennyExpress** app

```
$ cd PennyExpress  
$ npm install  
$ node runserver.js 55555  
Listening at port 55555
```

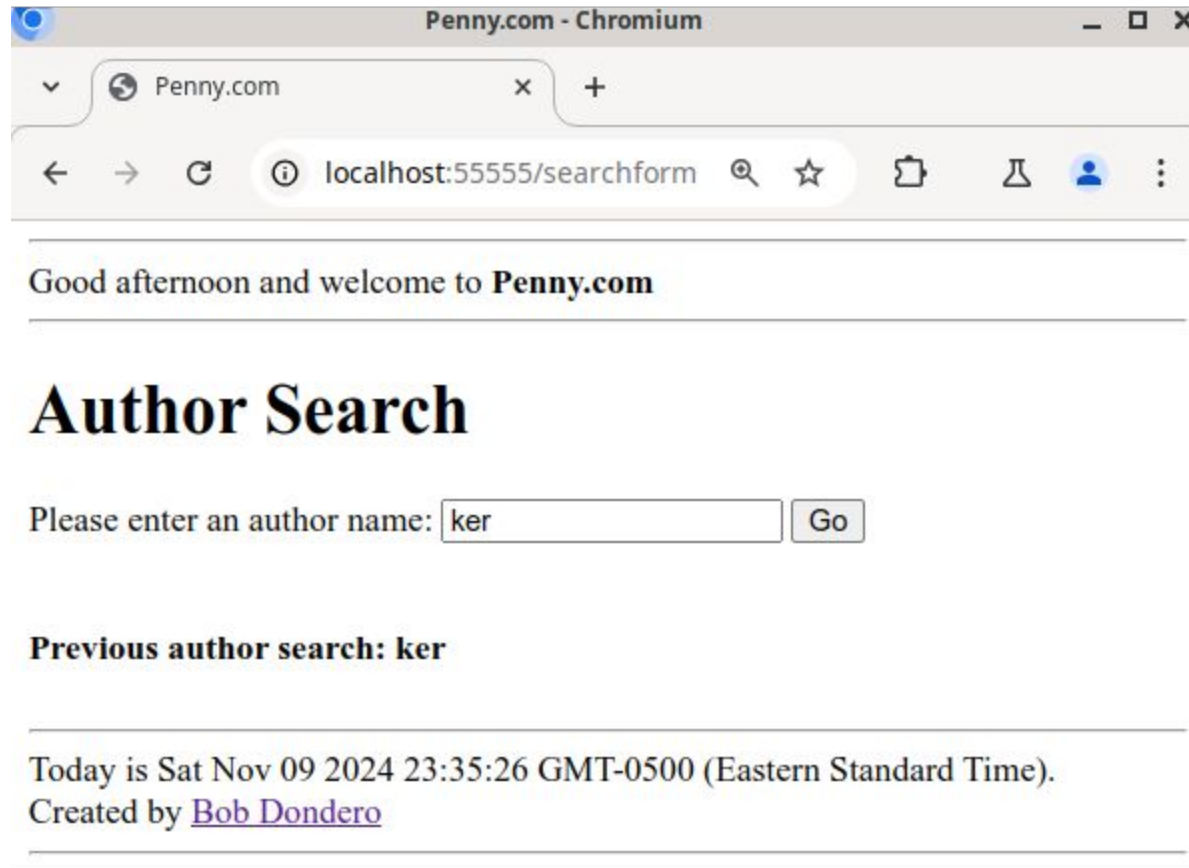
Express

- See PennyExpress app (cont.)



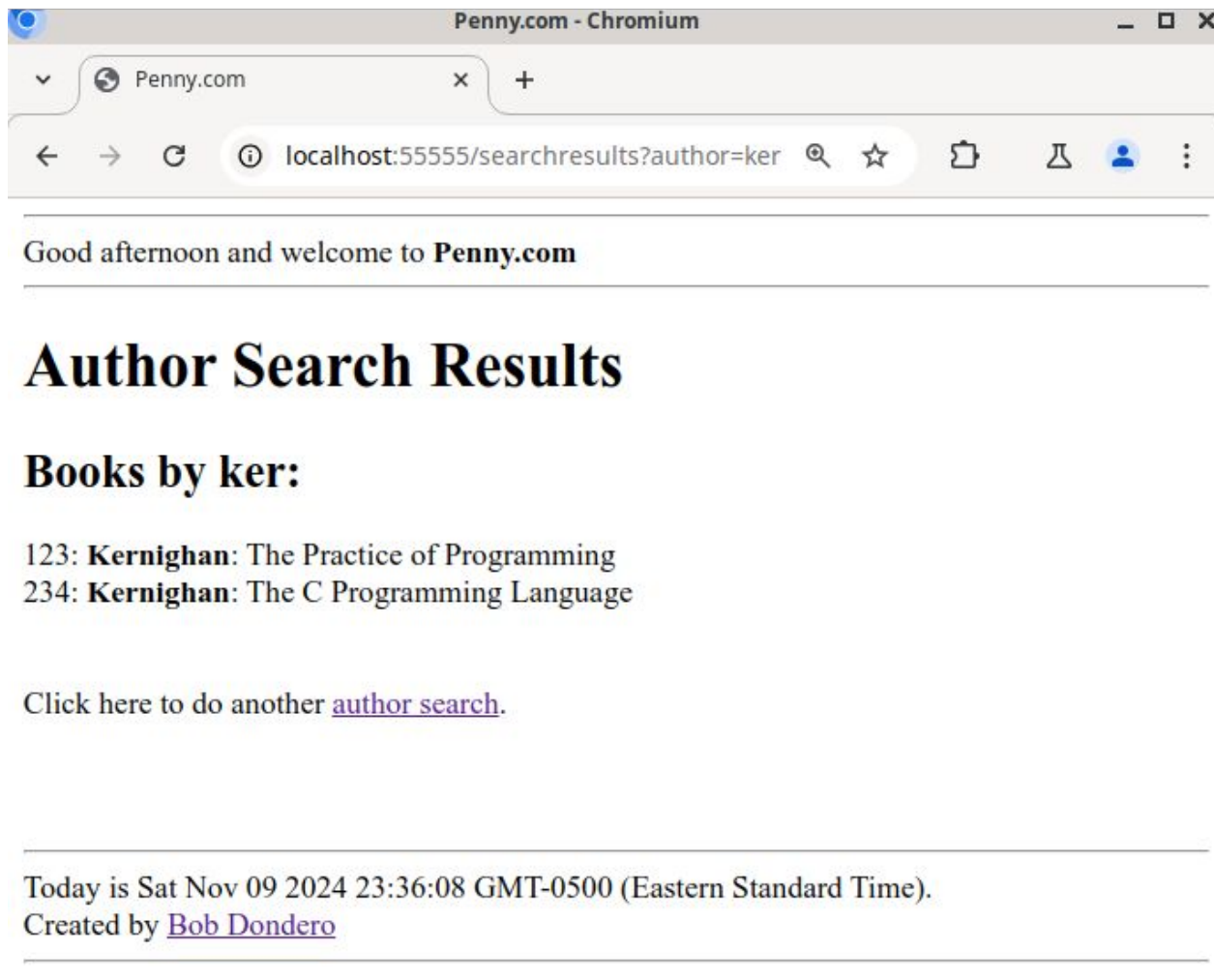
Express

- See **PennyExpress** app (cont.)



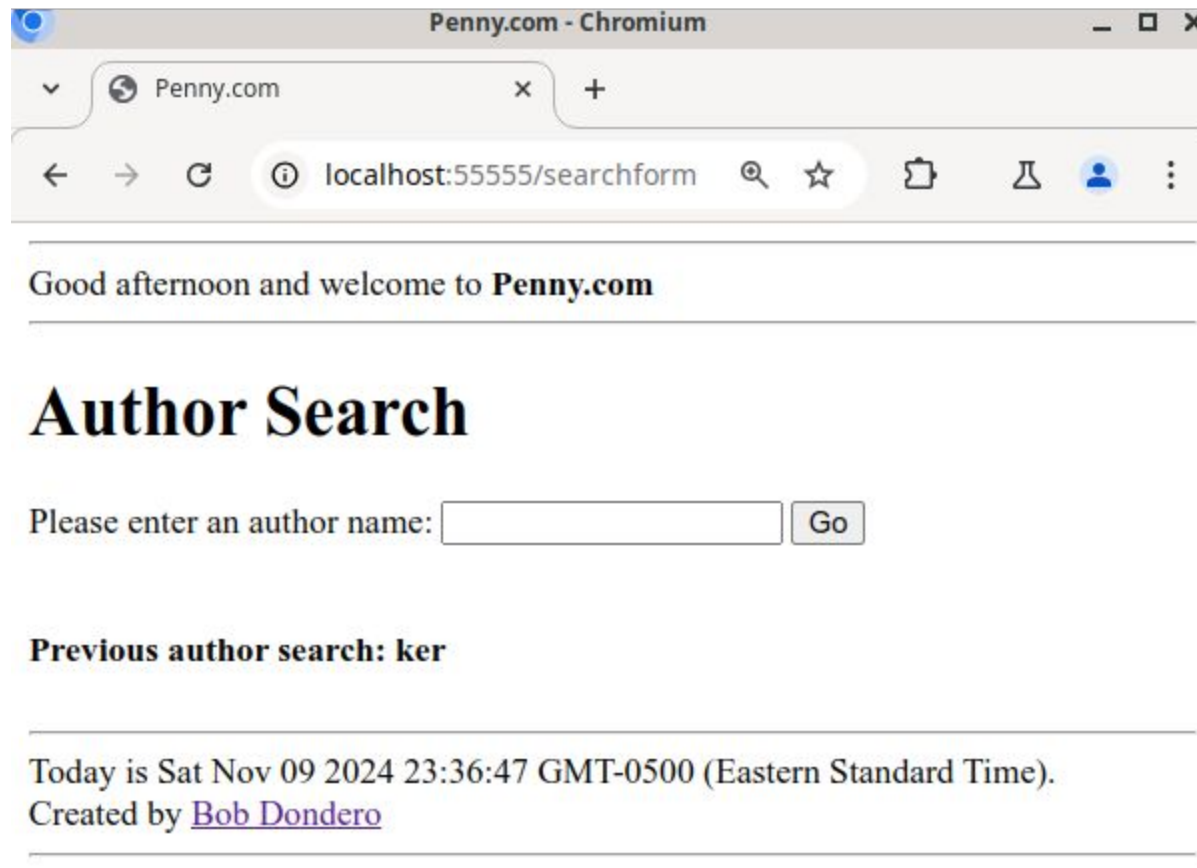
Express

- See **PennyExpress** app (cont.)



Express

- See **PennyExpress** app (cont.)



Express

- See **PennyExpress** app (cont.)
 - **runserver.js**
 - **package.json**
 - penny.sql, penny.sqlite
 - **database.js**
 - **header.html, footer.html, index.html, searchform.html, searchresults.html**
 - **penny.js**

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)

```

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 };

```

PennyExpress/header.html (Page 1 of 1)

```

1: <hr>
2: Good {{ampm}} and welcome to <strong>Penny.com</strong>
3: <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/~rdontero">
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)

```

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: {{ prevAuthor }}</strong>
17:     <br>
18:     <br>
19:     {{>footer}}
20:   </body>
21: </html>

```

PennyExpress/searchresults.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 Results</h1>
9:     <h2>Books by {{author}}:</h2>
10:    {{#books}}
11:      {{isbn}}: <strong>{{author}}</strong>: {{title}}<br>
12:    {{/books}}
13:    {{^books}}
14:      (None)
15:    {{/books}}
16:    <br>
17:    <br>
18:    Click here to do another
19:    <a href="/searchform">author search</a>.
20:    <br>
21:    <br>
22:    <br>
23:    <br>
24:    {{>footer}}
25:  </body>
26: </html>

```

PennyExpress/penny.js (Page 1 of 2)

```

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 mustacheExpress = require('mustache-express');
10: const database = require('./database');
11:
12: //-----
13:
14: function getAmPm() {
15:   let hours = new Date().getHours();
16:   return (hours < 12) ? 'morning' : 'afternoon';
17: }
18:
19: function getCurrentTime() {
20:   return new Date();
21: }
22:
23: //-----
24:
25: function parseCookies(req) {
26:   let cookies = {};
27:   let cookieHeader = req.headers.cookie;
28:   if (! cookieHeader)
29:     return cookies;
30:   let cookieStrs = cookieHeader.split('; ');
31:   for (let cookieStr of cookieStrs) {
32:     let nameValue = cookieStr.split('=');
33:     let name = nameValue[0];
34:     let value = nameValue[1];
35:     cookies[name] = value;
36:   }
37:   return cookies;
38: }
39:
40: //-----
41:
42: function index(req, res) {
43:   process.stdout.write(req.originalUrl + '\n');
44:
45:   let replacements = {
46:     ampm: getAmPm(),
47:     currentTime: getCurrentTime()
48:   };
49:   res.render('index', replacements);
50: }
51:
52: //-----
53:
54: function searchForm(req, res) {
55:   process.stdout.write(req.originalUrl + '\n');
56:
57:   let cookies = parseCookies(req);
58:   let prevAuthor = cookies.prevAuthor;
59:   if (! prevAuthor)
60:     prevAuthor = ' (None)';
61:
62:   let replacements = {
63:     ampm: getAmPm(),
64:     currentTime: getCurrentTime(),
65:     prevAuthor: prevAuthor

```

PennyExpress/penny.js (Page 2 of 2)

```

66:   };
67:   res.render('searchform', replacements);
68: }
69:
70: //-----
71:
72: function searchResults(req, res) {
73:   function handleBooks(err, books) { // Error handling omitted
74:     res.cookie('prevAuthor', author);
75:     let replacements = {
76:       ampm: getAmPm(),
77:       currentTime: getCurrentTime(),
78:       author: author,
79:       books: books
80:     };
81:     res.render('searchresults', replacements);
82:   }
83:
84:   process.stdout.write(req.originalUrl + '\n');
85:
86:   let author = req.query.author;
87:   if (! author)
88:     author = '';
89:   author = author.trim();
90:
91:   if (author === '') {
92:     res.cookie('prevAuthor', ' (None)');
93:     let replacements = {
94:       ampm: getAmPm(),
95:       currentTime: getCurrentTime(),
96:       author: ' (None)',
97:       books: []
98:     };
99:     res.render('searchresults', replacements);
100:   }
101:   else
102:     database.getBooks(author, handleBooks);
103: }
104:
105: //-----
106:
107: let app = express();
108: app.engine('html', mustacheExpress());
109: app.set('view engine', 'html');
110: app.set('views', '.');
111:
112: app.get('/', index);
113: app.get('/index', index);
114: app.get('/searchform', searchForm);
115: app.get('/searchresults', searchResults);
116:
117: module.exports = { app };

```

Express

- There is much more to Express and Mustache-express
 - Express website:
 - <https://expressjs.com/>
 - Mustache-express website:
 - <https://www.npmjs.com/package/mustache-express>

Commentary

	Small Apps	Medium Apps	Large Apps
PHP	✓		
Python/Flask		✓	
JavaScript/Express		✓	
Java/Spring			✓

Summary

- We have covered:
 - Some server-side options
 - Java: Servlets
 - Java: The Spring web app framework
 - JavaScript: The Express web app framework