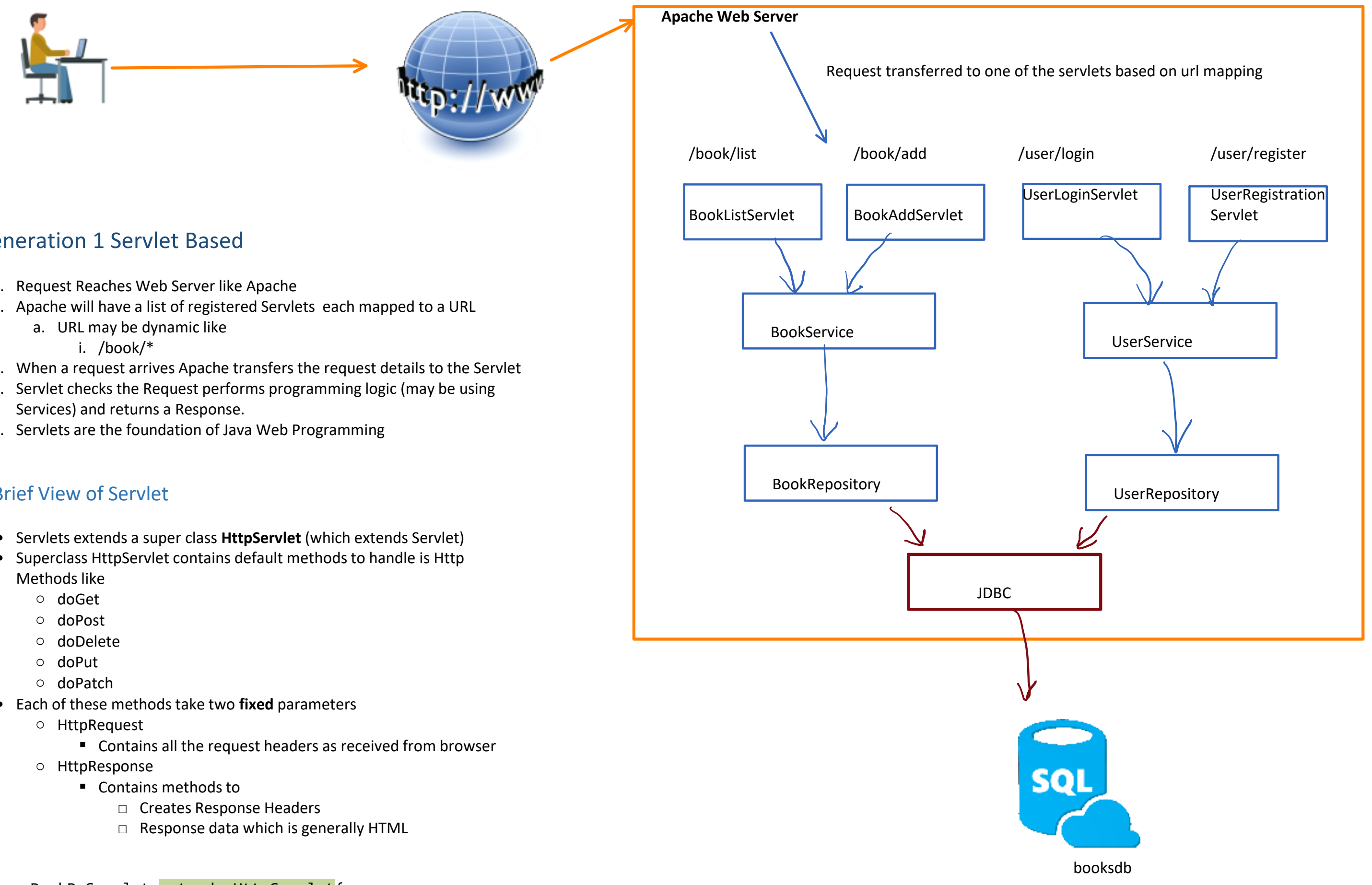
Web Application

Friday, November 6, 2020 10:19 AM

A web server doesn't know how to interact with a database

http://booksweb.org/books/by/jeffrey-archer



Generation 1 Servlet Based

1. Request Reaches Web Server like Apache
2. Apache will have a list of registered Servlets each mapped to a URL
    a. URL may be dynamic like
        i. /book/*
3. When a request arrives Apache transfers the request details to the Servlet
4. Servlet checks the Request performs programming logic (may be using Services) and returns a Response.
5. Servlets are the foundation of Java Web Programming

A Brief View of Servlet

- Servlets extends a super class **HttpServlet** (which extends Servlet)
- Superclass HttpServlet contains default methods to handle is Http Methods like
    - doGet
    - doPost
    - doDelete
    - doPut
    - doPatch
- Each of these methods take two **fixed** parameters
    - HttpRequest
        - Contains all the request headers as received from browser
    - HttpResponse
        - Contains methods to
            - Creates Response Headers
            - Response data which is generally HTML

```
class BookByServlet extends HttpServlet{

    @Override
    public void doGet(HttpRequest request, HttpResponse
    resp){

        //1. get user request
        String author= req.getQueryString("author")

        //2. get data from the server

        List<Book>
        books=bookService.getAllBooksBy(author);

        //3. create html for the browser
        String html= htmlGenerator.buildHtmlFor(books)

        //4. send response to the browser
        resp.getWriter().write(html);
    }

}
```

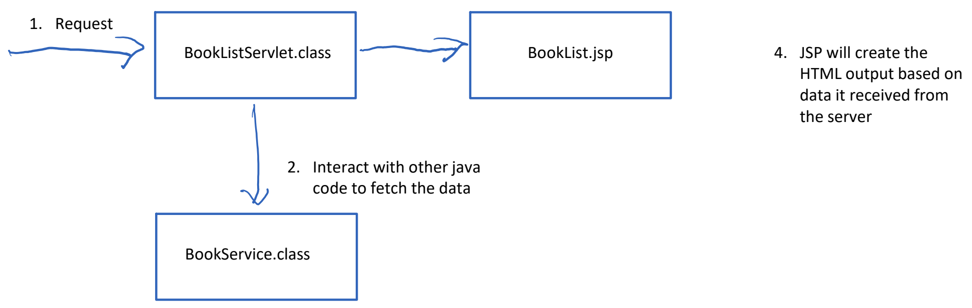
Problems

1. We need One Servlet Per URL
    - A servlet can serve wild card url
    - But generally we design different servlet for different funcionality
        - Web don't want a single BookServlet doing everything about book
        - It needs to check url, do a switch case
        - It will increase the code
2. We need to generate HTML programmatically using String manipulation that will become complex for large UI
3. There is no support for dependency injection of services and repository here.

Generation 2 — JSP and Servlet

- Java introduced JSP or Java Server Pages
- They are like HTML pages where you can embed Java code or data from a Java Program
- Internally a JSP will be translated and compiled as a Servlet
- This approach solves One Major Problem —>
    - we don't need to write HTML inside Java
- But we can't replace Servlet with JSP
    - They we need to write java logic in html like page
- So We use a combination mechanism

3. Servlet will dispatch request with data to JSP


1. Request
BookListServlet.class
BookList.jsp
4. JSP will create the HTML output based on data it received from the server
2. Interact with other java code to fetch the data
BookService.class

```
class BookByServlet extends HttpServlet{

    @Override
    public void doGet(HttpRequest request, HttpResponse resp){

        //1. get user request
        String author= req.getQueryString("author")

        //2. get data from the server

        List<Book> books=bookService.getAllBooksBy(author);

        //3. create html for the browser
        req.addAttribute("books", books);
        req.getDispatcher("booklist.jsp").dispatch(req,resp)
    }

}
```

Problems

We still have our previous unsolved problems

1. No dependency injection
2. Too many Servlets created one for each functionality

Generation 3 — Spring MVC (Controller Based)

- Spring Introduces its own framework to deal with web programming
- **Spring provides its own Servlet that would catch all the URLs for a web application**
    - **Every URL will reach Spring's pre-created Servlet called DispatcherServlet**
- **We will not create a single servlet in our project**
- When any request is made, the web server will transfer the request to SpringServlet
- Once Spring Receives the request it
    1. Checks the URL and HTTP Method
    2. Based on URL and HTTP Methods, it transfers the control to one of the user defined functions present in a controller

Controller

- A Controller is a plain Java class with @Controller annotation
    - @Control is also an stereotype for @Component
- This class doesn't extend any other class
- It can have different user defined methods to handle different URLS
- User defined methods can take parameters based on their needs
    - It may be something from
        - Http request
        - HTTP header
        - Spring context
- Controllers returns are processed by Spring Dispatcher Servlet to produce the final response

Controller vs Servlet code

Functionality	Servlet

Get All Books	`class BookListServlet extends HttpServlet{`
	
	`  public doGet(HttpRequest req, HttpResponse resp){`
	
	`  }`
	`}`

Get Book By Id	`class BookByIdServlet extends HttpServlet{`
	
	`  public doGet(HttpRequest req, HttpResponse resp){`
	`    String id=req.getUrl().substring(req.getUrl.lastIndexOf("/")+1);`
	`    …`
	`  }`
	`}`

AddBook	`class BookAddServlet extends HttpServlet{`

Controller

@Controller
public class BookController{
 	
@GetMapping("/book/list")
public String getAllBooks(){
return "list.jsp";
}
 	
@GetMapping("/books/by/{id}")
public String getBookById(String id){
return "details.jsp"
}
 	
@GetMapping("/books/add")

	`…`
	`}`
	`}`

AddBook	`class BookAddServlet extends HttpServlet{`
	`public doGet(HttpRequest req, HttpResponse resp){`
	`//Show Book Create Form`
	`…`
	`}`
	`public doPost(HttpRequest req, HttpResponse resp){`
	`//Handle Form submission and database insert`
	`…`
	`}`
	`}`

```
        return "details.jsp"
    }


    @GetMapping("/books/add")
    public String showBookForm(){
        return "bookform.jsp";
    }

    @PostMapping("/book/add")
    public String addBook(Book book){
        //add the book
        return "list.jsp";
    }

}
```

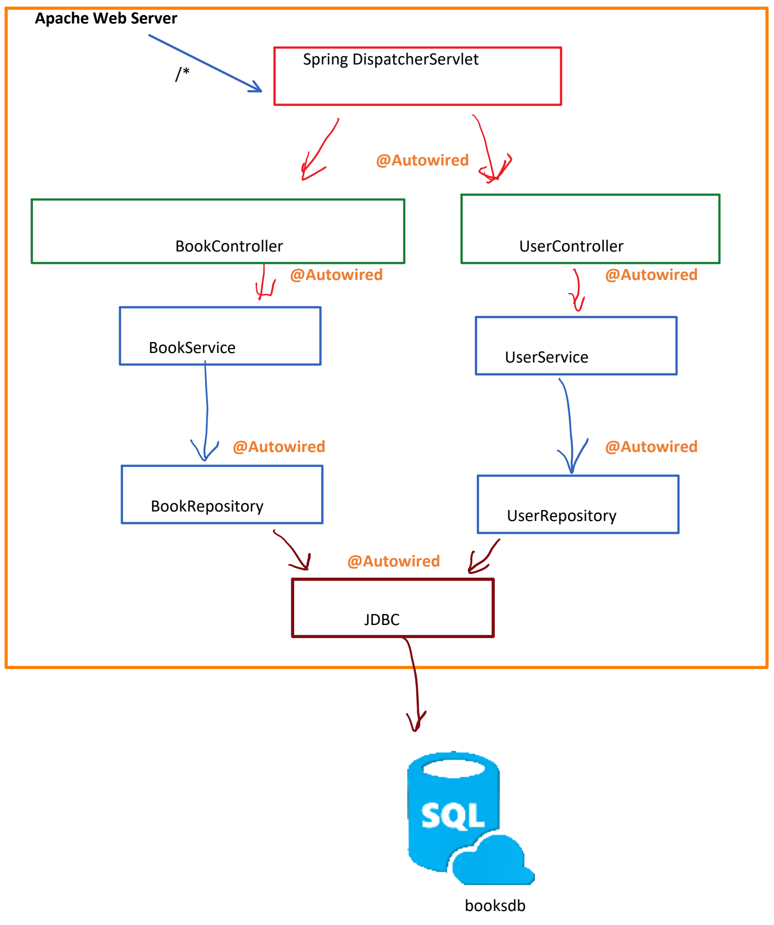
Controller Advantage over Servlets

- Compared to Servlets, Controllers are simple classes
  - They don't need to extends any superclass
  - They are generally not required to directly interact with request and response objects

- A controller can process multiple functionationlity or URL in different user defined functions
  - This functions can have more meanigful names
- A controller can use all the Spring Framework features like
  - Dependency Injection
  - JPA support
  - AOP support
- A controller is based on simple functions which allows
  - Fewer classes
    - Instead of creating a class for each functionality you create multiple functions in same class
  - It is easier to unit test
    - It takes simple parameters and returns simple parameters
  - No need to manually process the request and response objects
-

Summary!

- A controller provides a much shorter, simpler and Springfield web programming which is more testable and scalable
- It allows you to leverage other features of spring
  - DI
  - AOP
  - JPA
  - Security
  - Testing


Apache Web Server
/*
Spring DispatcherServlet
@Autowired
BookController
UserController
@Autowired
@Autowired
BookService
UserService
@Autowired
@Autowired
BookRepository
UserRepository
@Autowired
JDBC
SQL
booksdb