

Web Frameworks: Spring

An Introduction

Application frameworks

- ❑ provides frozen spots
 - ❑ overall architecture
 - ❑ How the components interact
- ❑ allows to concentrate in hot spots to extend the behaviour of the framework
 - ❑ Hot spots are the functions written for the application
- ❑ A framework is not suitable for a problem when ...

Web Application Frameworks

- ❑ An application framework that is designed to support development of web applications that generally includes
 - ❑ Database support
 - ❑ Templating framework for generating dynamic web content
 - ❑ HTTP session management with middleware support
 - ❑ Built-in testing framework
- ❑ It can also support internationalization, security and privacy
- ❑ Consistent look and feel and consistent with database



Web Frameworks examples

- ☐ Ruby on Rails
- ☐ Play
- ☐ ASP.NET
- ☐ Django
- ☐ Symfony
- ☐ Spring
- ☐ Vue.js
- ☐ Angular js

Introducing Spring

Introduction

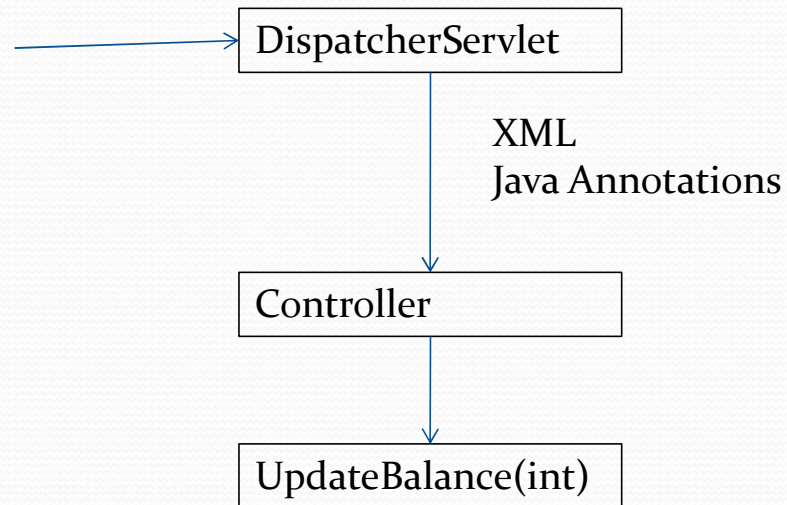
- Web.xml routes requests to the individual servlet's doGet or doPost methods
- doGet(...)
 - //extract parameters from request
 - //validation
 - //Construct objects with parameters
 - //do the processing



Spring

- In Spring
 - A specialized servlet-DispatcherServlet
 - One or more controllers having simple methods to process HTTP requests
 - The DispatcherServlet routes requests to appropriate controller-individual methods of the controllers
 - DispatcherServlet extracts request parameters, performs data validation and marshalling
 - Provides an extra layer of routing over web.xml

Spring



Routing is possible based on

1. Path like servlets
2. Request parameters using annotations
3. Data validation is taken care of

Routing through DispatcherServlet

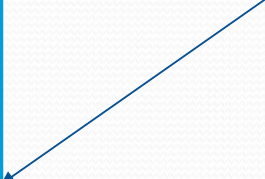
```
(  
public class ContactController {  
  
    public Contacts getContacts() {  
        //retrieve contacts  
        Contacts c=...  
        ...  
        return c;  
    }  
}
```

A Simple java class-no
framework code

Routing through DispatcherServlet

```
public class ContactController {  
    public Contacts getContacts() {  
        //retrieve contacts  
        Contacts c=...  
        ...  
        return c;  
    }  
  
    public void friends(){  
        ...  
    }  
}
```

A Simple java class-no
framework code



Introduction

- In EJB *public class HelloWorldBean implements SessionBean {*
- Spring avoids (as much as possible) littering your application code with its API
- Spring almost never forces you to implement a Spring-specific interface or extend a Spring-specific class
- Instead, the classes in a Spring-based application often have no indication that they're being used by Spring
- Spring has enabled the return of the plain old Java object (POJO) to enterprise development

Mapping Request parameters to method parameters

@Controller

```
public class ContactController {
```

@RequestMapping("/search")

```
public Contacts searchContacts(
```

```
    @RequestParam searchstr String SearchStr) {
```

```
    //retrieve contacts
```

```
    Contacts c=...
```

```
    ...
```

```
    return c;
```

```
}
```

Retrieves request parameters and performs basic data validation so that value of *searchstr* can be mapped to *SearchStr*

Mapping Request parameters to method parameters

@Controller

```
public class ContactController {
```

```
@RequestMapping(value={"/search"}, method = RequestMethod.GET)
```

```
public Contacts searchContacts(
```

```
    @RequestParam searchstr String SearchStr) {
```

```
    //retrieve contacts
```

```
    Contacts c=...
```

```
    ...
```

```
    return c;
```


```
}
```


Mapping Requests

```
@RestController
@RequestMapping(value = "/demo")
public class LoginController {

    @RequestMapping(value = "/login")
    public String sayHello1() {
        return "Hello World ";
    }

    @RequestMapping(value = "/dummy")
    public String sayHello() {
        return "Hello World dummy";
    }
}
```


- 
- No need to worry about
 - how that request got to the server,
 - what format it got there in,
 - how all the data got extracted from it.
 - It simplifies the methods and write cleaner, simpler methods, by using request parameters in the request mapping to extract that data and pass it into the method



@Controller

```
public class ContactController {
```

```
@RequestMapping("/search/{str}")
```

```
    public Contacts searchContacts(
```

```
        Search s) {
```

```
        //retrieve contacts
```

```
        Contacts c=...
```

```
        ...
```

```
        return c;
```

```
    }
```

Path variable provides a nicer way of parsing the request parameters rather than
?<key>=value

@Controller

```
public class ContactController {
```

@RequestMapping("/search/")

```
    public Contacts searchContacts(
```

```
        Search s) {
```

```
        //retrieve contacts
```

```
        Contacts c=...
```

```
        ...
```

```
        return c;
```

```
}
```

```
public class Search {
```

```
    private String fname;
```

```
    private String lname;
```

```
    public String
```

```
    getFname() {..}
```

```
    public setFname(String  
        name) {..}
```

```
    ...}
```

Automatic data marshalling
through HTTP message
converters

Response

- `@RequestMapping(value = "/prod", produces = {"application/JSON"})`
- `@ResponseBody StringResponse getProduces() {`
- `StringResponse s= new StringResponse();`
- `s.setResponse("Attribute!");`
- `return s;`
- `}`

```
{"response": "Attribute!"}
```



Key Features of Spring

- Inversion of Control
- Dependency Injection
- Aspect Oriented Programming

Inversion of Control

- Inversion of Control (or IoC) covers a broad range of techniques that allow an object to become a passive participant in the system

```
public class BankAccount {  
    public void transfer(BigDecimal amount, BankAccount recipient) {  
  
        recipient.deposit(this.withdraw(amount));  
    }  
  
    public void closeOut() {  
  
        this.open = false;  
    }  
  
    public void changeRates(BigDecimal newRate) {  
  
        this.rate = newRate;  
    }  
}
```

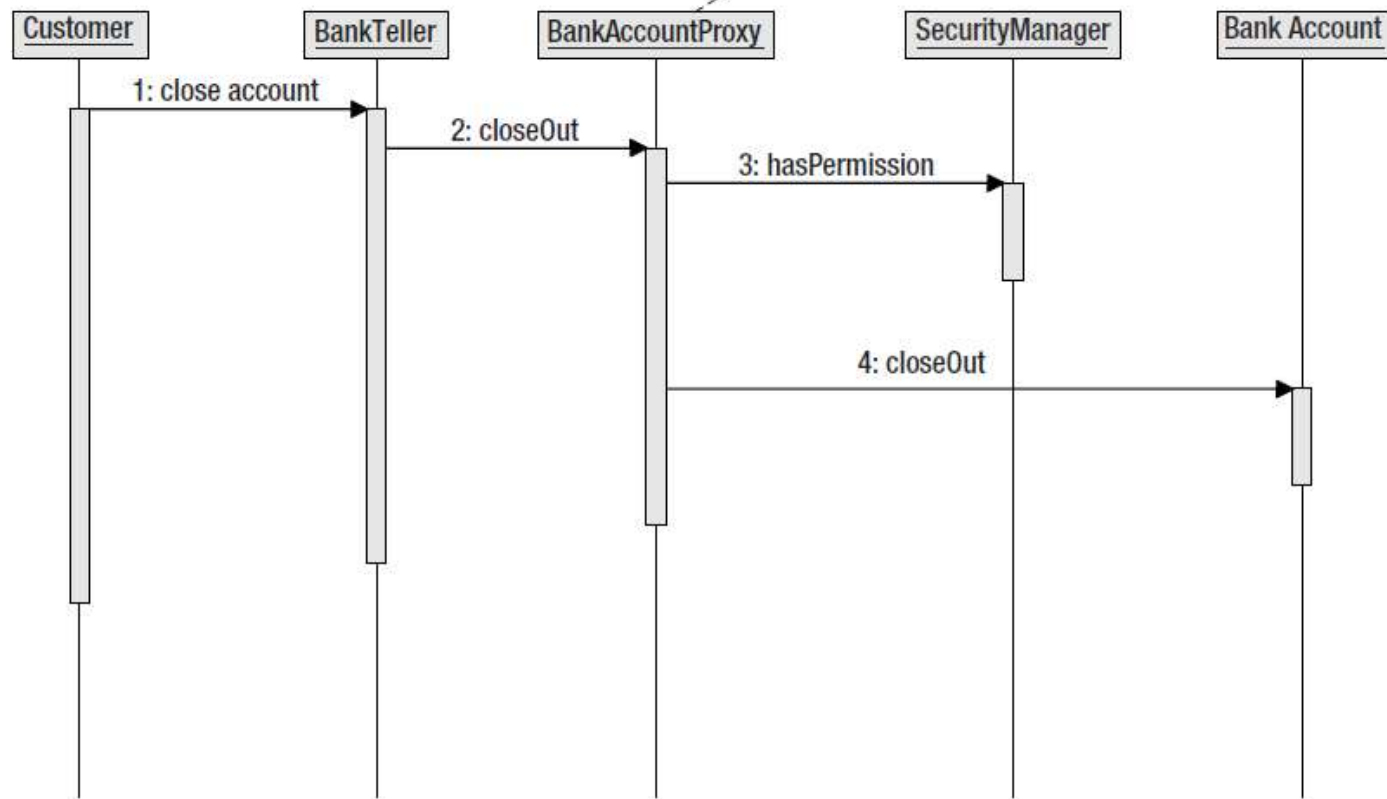

IoC Enabled code

```
public class BankAccount {  
    public void transfer(BigDecimal amount, BankAccount recipient) {  
        recipient.deposit(this.withdraw(amount));  
    }  
  
    public void closeOut() {  
        this.open = false;  
    }  
  
    public void changeRates(BigDecimal newRate) {  
        this.rate = newRate;  
    }  
}
```

- You can add the authorization mechanism into the execution path with a type of IoC implementation called *aspect-oriented programming (AOP)*

AOP

- Aspects are concerns of the application that apply themselves across the entire system
- The SecurityManager is one example of a system-wide aspect, as its hasPermission methods are used by many methods
- Other typical aspects include logging, and auditing of events
- While we may have removed calls to the SecurityManager from the BankAccount, the deleted code will still be executed in the AOP framework either at compile time or at runtime
- Runtime-proxy based AOP (simple)
- Compile time- weaving (stronger than proxies)



Inversion of Control is the broad concept of giving control back to the framework
This control can be control over creating new objects, control over transactions, or
control over the security implementation