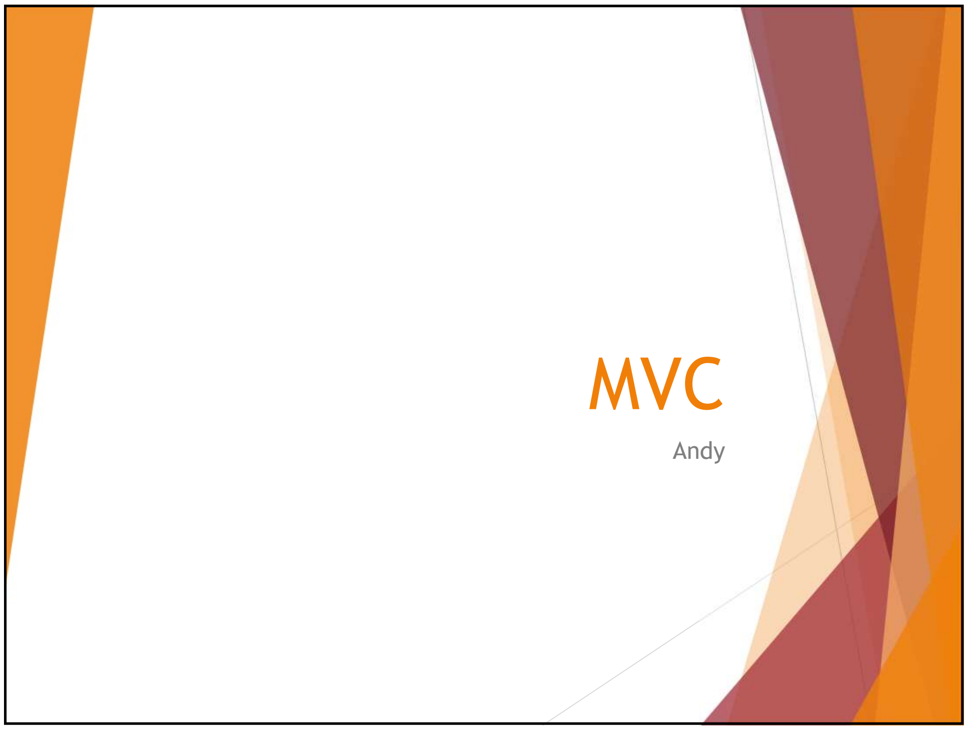
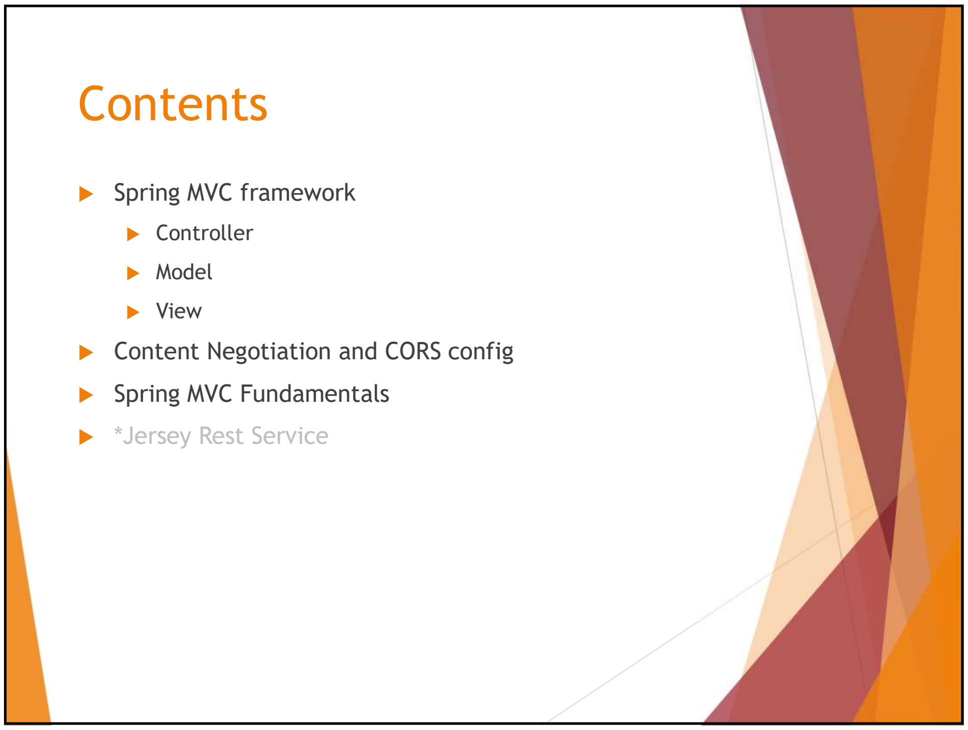


# MVC

Andy

1



## Contents

- Spring MVC framework
  - Controller
  - Model
  - View
- Content Negotiation and CORS config
- Spring MVC Fundamentals
- *Jersey Rest Service

2

## URL Composition

- URL composition
  - Example:
    - http://localhost:8080/userapp/user/{id}/load?minAge=20&lastName=Stark;
    - http://localhost:8080/userapp/user/2/load?minAge=20&lastName=Stark;
  - Interpretation:
    - **Domain:** http://localhost:8080/ **(http protocol, server, port)**
    - App route/name: /userapp (could be omitted as /)
    - Request route: /user/id/load
    - **Path variable**: id (part of the url / routes)
    - **Query parameter**: minAge, lastName (key=value pair after ?)

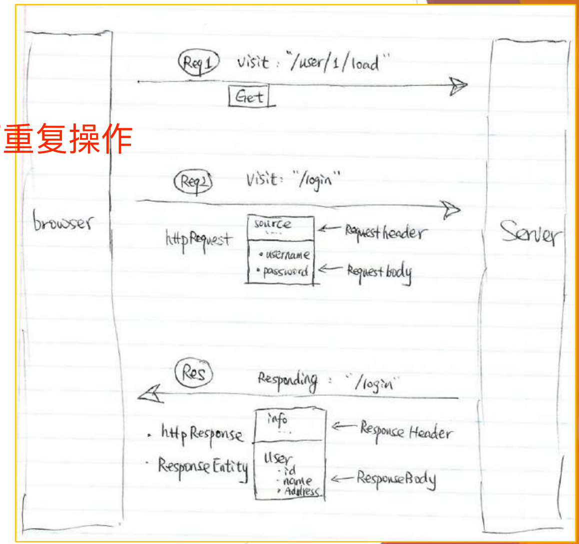
- What will happen after hitting this url:
  - 1. Based on domain, it will find the server
  - 2. Based on app route, server will allocate the application (userapp)
  - *3. the application will use "**controller**" to match the request route ("/user/id/load")
  - 4. path variable and query parameter will be passed as parameter to the controller
  - 5. controller returns result and display the view page to user

3

## URL Request & Response

- Request 可重复操作
  - Method:
    Get/Put/Post/Delete 不可重复操作
  - Route:
    RequestMapping
  - Data:
    - url: path variable, query parameter
    - Request body

- Response
  - Data: response body
  - View (with data)



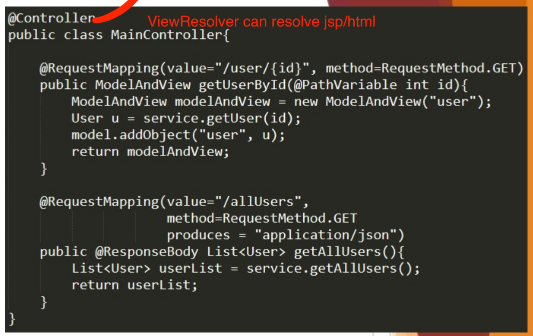
4

标注这是一个rest的controller，可以接受前端发送的请求

rest是一种前后端交流的协议

## Controller

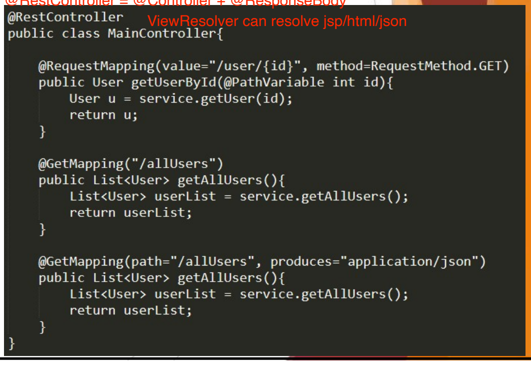
(Spring rest service)



```
@Controller
public class MainController{

    @RequestMapping(value="/user/{id}", method=RequestMethod.GET)
    public ModelAndView getUserById(@PathVariable int id){
        ModelAndView modelAndView = new ModelAndView("user");
        User u = service.getUser(id);
        model.addObject("user", u);
        return modelAndView;
    }

    @RequestMapping(value="/allUsers",
                    method=RequestMethod.GET
                    produces = "application/json")
    public @ResponseBody List<User> getAllUsers(){
        List<User> userList = service.getAllUsers();
        return userList;
    }
}
```

ViewResolver can resolve jsp/html

@RestController = @Controller + @ResponseBody



```
@RestController
public class MainController{

    @RequestMapping(value="/user/{id}", method=RequestMethod.GET)
    public User getUserById(@PathVariable int id){
        User u = service.getUser(id);
        return u;
    }

    @GetMapping("/allUsers")
    public List<User> getAllUsers(){
        List<User> userList = service.getAllUsers();
        return userList;
    }

    @GetMapping(path="/allUsers", produces="application/json")
    public List<User> getAllUsers(){
        List<User> userList = service.getAllUsers();
        return userList;
    }
}
```

ViewResolver can resolve jsp/html/json

5

## Request Mapping Methods

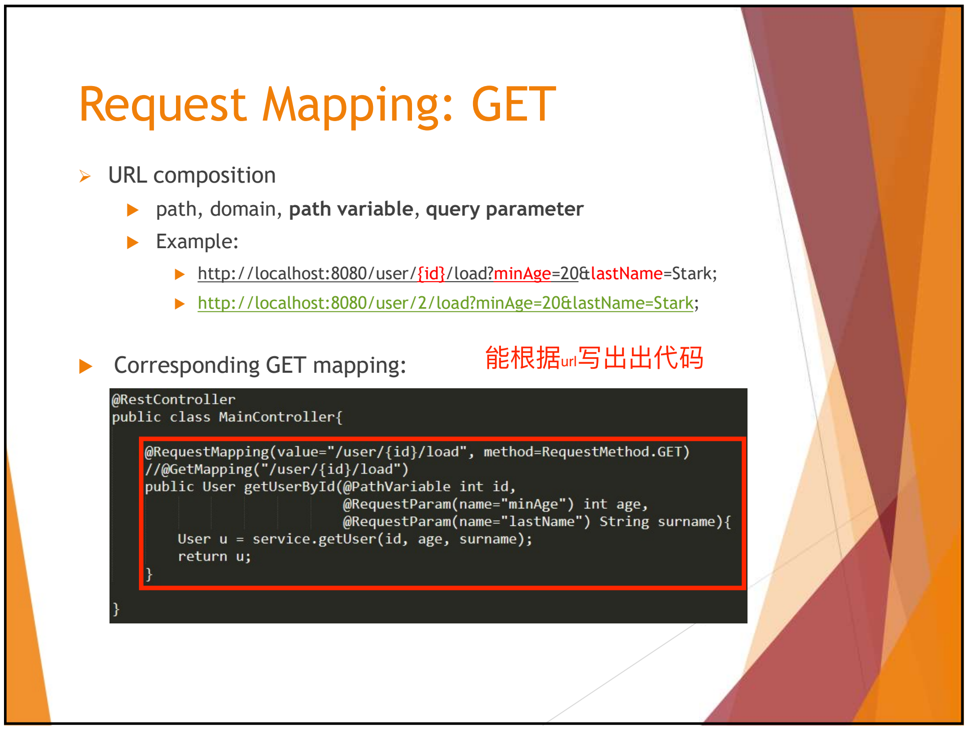
(rest service http methods)

- @RequestMapping(method = RequestMethod.GET)

- @GetMapping --- Retrieve/Read Data (R)
- @PutMapping --- Update data (U)
- @PostMapping --- Add new data (C)
- @DeleteMapping --- delete existing data (D)

Note: those methods are just contract based- they don't enforce the action. Meaning you can create new using Post method, but it is against the design principle.

6

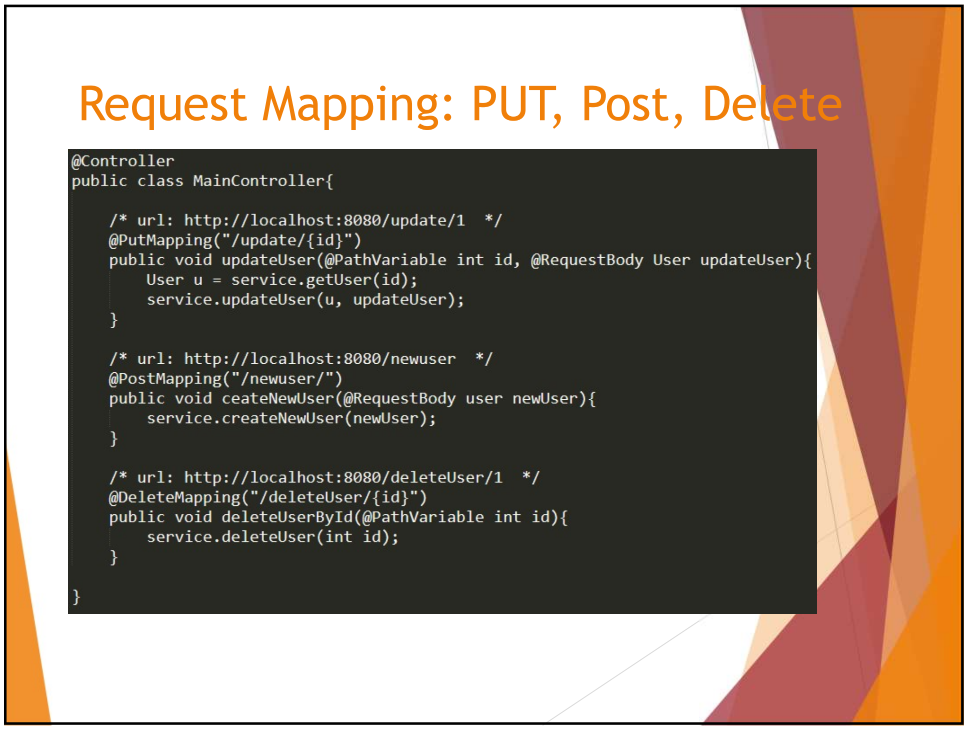


## Request Mapping: GET

- URL composition
  - path, domain, **path variable**, **query parameter**
  - Example:
    - http://localhost:8080/user/{id}/load?minAge=20&lastName=Stark;
    - http://localhost:8080/user/2/load?minAge=20&lastName=Stark;
- Corresponding GET mapping: 能根据url写出出代码

```
@RestController
public class MainController{

    @RequestMapping(value="/user/{id}/load", method=RequestMethod.GET)
    //@GetMapping("/user/{id}/load")
    public User getUserById(@PathVariable int id,
                            @RequestParam(name="minAge") int age,
                            @RequestParam(name="lastName") String surname){
        User u = service.getUser(id, age, surname);
        return u;
    }

}
```

7



## Request Mapping: PUT, Post, Delete

```
@Controller
public class MainController{

    /* url: http://localhost:8080/update/1  */
    @PutMapping("/update/{id}")
    public void updateUser(@PathVariable int id, @RequestBody User updateUser){
        User u = service.getUser(id);
        service.updateUser(u, updateUser);
    }

    /* url: http://localhost:8080/newuser  */
    @PostMapping("/newuser/")
    public void ceateNewUser(@RequestBody user newUser){
        service.createNewUser(newUser);
    }

    /* url: http://localhost:8080/deleteUser/1  */
    @DeleteMapping("/deleteUser/{id}")
    public void deleteUserById(@PathVariable int id){
        service.deleteUser(int id);
    }

}
```

8

## Http Return Type

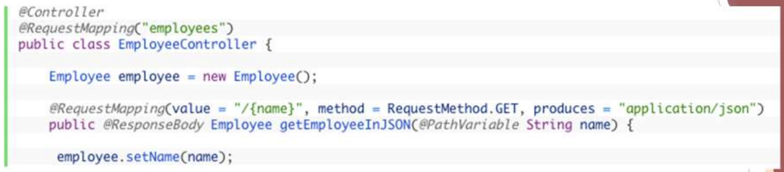
```
@GetMapping({"/user", "/"})
//@ResponseBody
public @ResponseBody User getUser(){
    return userManager.findUser();
}

@GetMapping({"/user", "/"})
public ResponseEntity<User> getUser(){
    User u = userManager.findUser();
    return new ResponseEntity<User>(u, HttpStatus.OK);
}

@GetMapping({"/user", "/"})
public ModelAndView getUser(){
    ModelAndView mv = new ModelAndView("user"); //user.jsp
    User u = userManager.findUser();
    mv.addObject("user", u);
    return mv;
}
```

```
@GetMapping({"/user", "/"})
public ModelAndView getUser(){
    ModelAndView mv = new ModelAndView("user"); //user.jsp
    User u = userManager.findUser();
    mv.addObject("user", u);
    return mv;
}

@GetMapping({"/user", "/"})
public String getUser(Model model){
    User u = userManager.findUser();
    model.addAttribute("user", u);
    return "user";  //user.jsp will use model
}

@GetMapping({"/user", "/"})
public User getUser(){
    User u = userManager.findUser();
    return u;
}
```

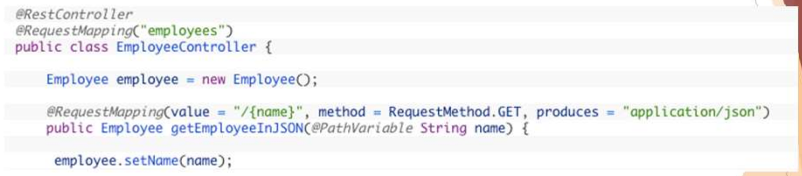
What if we just want to return simple json? use @ResponseBody

9

## @Controller vs @RestController



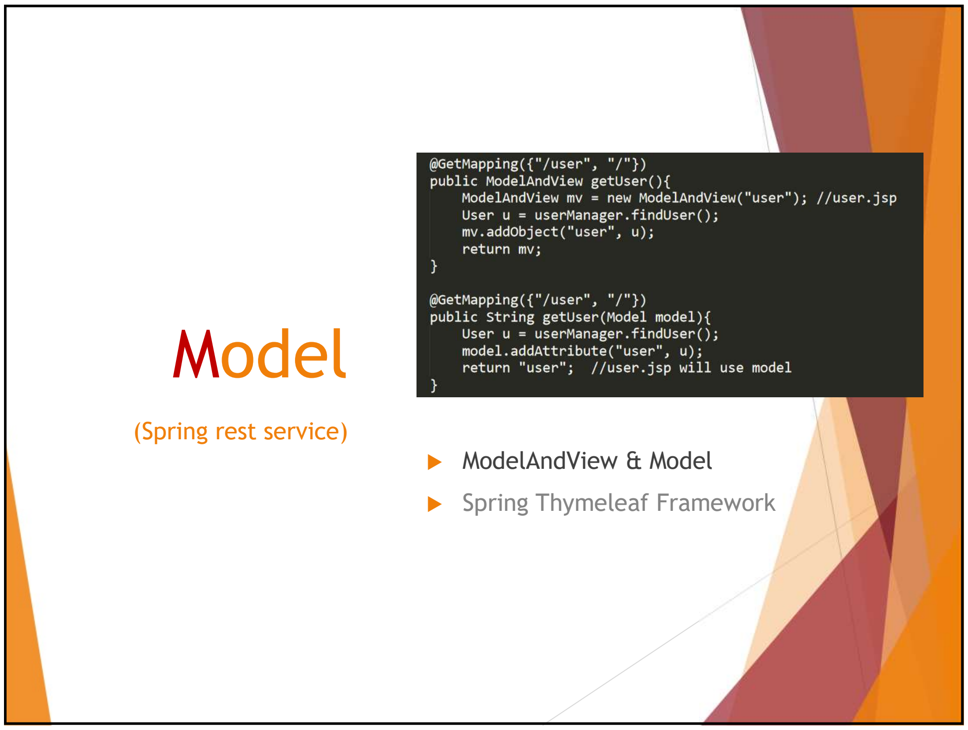
```
@Controller
@RequestMapping("employees")
public class EmployeeController {

    Employee employee = new Employee();

    @RequestMapping(value = "/{name}", method = RequestMethod.GET, produces = "application/json")
    public @ResponseBody Employee getEmployeeInJSON(@PathVariable String name) {

     employee.setName(name);
```



```
@RestController
@RequestMapping("employees")
public class EmployeeController {

    Employee employee = new Employee();

    @RequestMapping(value = "/{name}", method = RequestMethod.GET, produces = "application/json")
    public Employee getEmployeeInJSON(@PathVariable String name) {

     employee.setName(name);
```

@RestController = @Controller + @ResponseBody

10



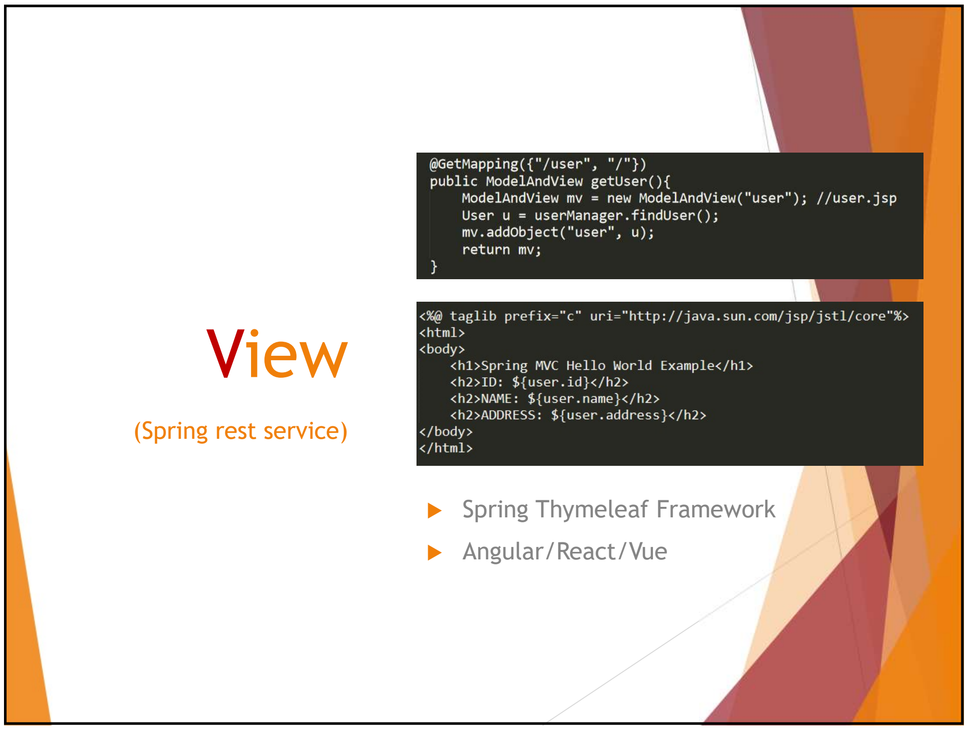
## Model

(Spring rest service)

```
@GetMapping({"/user", "/"})
public ModelAndView getUser(){
    ModelAndView mv = new ModelAndView("user"); //user.jsp
    User u = userManager.findUser();
    mv.addObject("user", u);
    return mv;
}

@GetMapping({"/user", "/"})
public String getUser(Model model){
    User u = userManager.findUser();
    model.addAttribute("user", u);
    return "user";  //user.jsp will use model
}
```

- ModelAndView & Model
- Spring Thymeleaf Framework

11



## View

(Spring rest service)

```
@GetMapping({"/user", "/"})
public ModelAndView getUser(){
    ModelAndView mv = new ModelAndView("user"); //user.jsp
    User u = userManager.findUser();
    mv.addObject("user", u);
    return mv;
}
```

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<body>
    <h1>Spring MVC Hello World Example</h1>
    <h2>ID: ${user.id}</h2>
    <h2>NAME: ${user.name}</h2>
    <h2>ADDRESS: ${user.address}</h2>
</body>
</html>
```

- Spring Thymeleaf Framework
- Angular/React/Vue

12

## HttpHeader & Content Negotiation

- **RequestHeader**

  **accept** --- Accept: application/json

  **content-type** --- Content-Type: application/xml

  **cookie** --- Cookie: access_token=eyJhbGciOiJIUzI1NiIsI.eyJpc3MiOiJodHRwczotcGxlL.mFrs3Zo8eaSNcxiNfvRh9dqKP4F1cB;

  **Origin** --- Origin: http://www.example-social-network.com

- **Content Negotiation**

  **Front-End: RequestHeader contains: accept vs content-type**

  **Back-End: produces->accept && consumes <- content-type**

```
@RequestMapping(value="/allUsers",
                method=RequestMethod.GET
                produces = "application/json",
                consumes = "application/xml")
public @ResponseBody List<User> getAllUsers(){
    List<User> userList = service.getAllUsers();
    return userList;
}
```

13

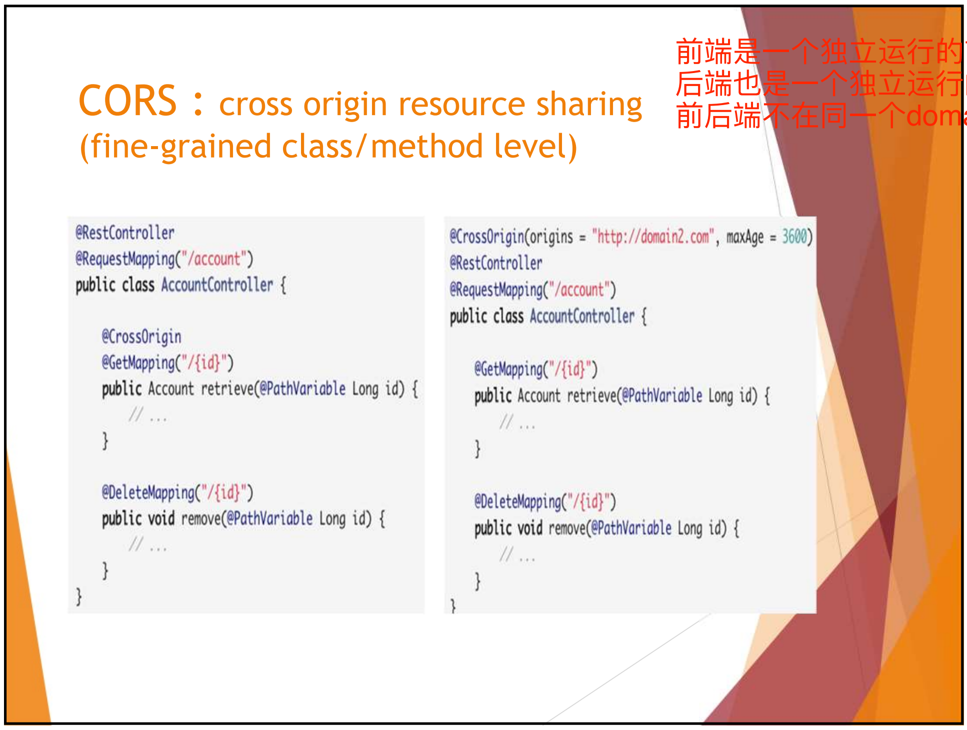
## Cookie vs Session

- Cookie: front end storage, small size, can be used with every http request, popular for application with independent front end.
- Session: back end storage, has expiration time (~20mins), corresponding to front end tab. Expires if time is reached without user interactions or the tab is closed. Usually used with jsp.
- Get Mapping for Cookie and Session:

```
@GetMapping("/demo")
public void handle(@CookieValue("JSESSIONID") String cookie) {
    //...
}

@RequestMapping("/")
public String handle(@SessionAttribute User user) {
    // ...
}
```

14

## BQ里面的challenge答案！



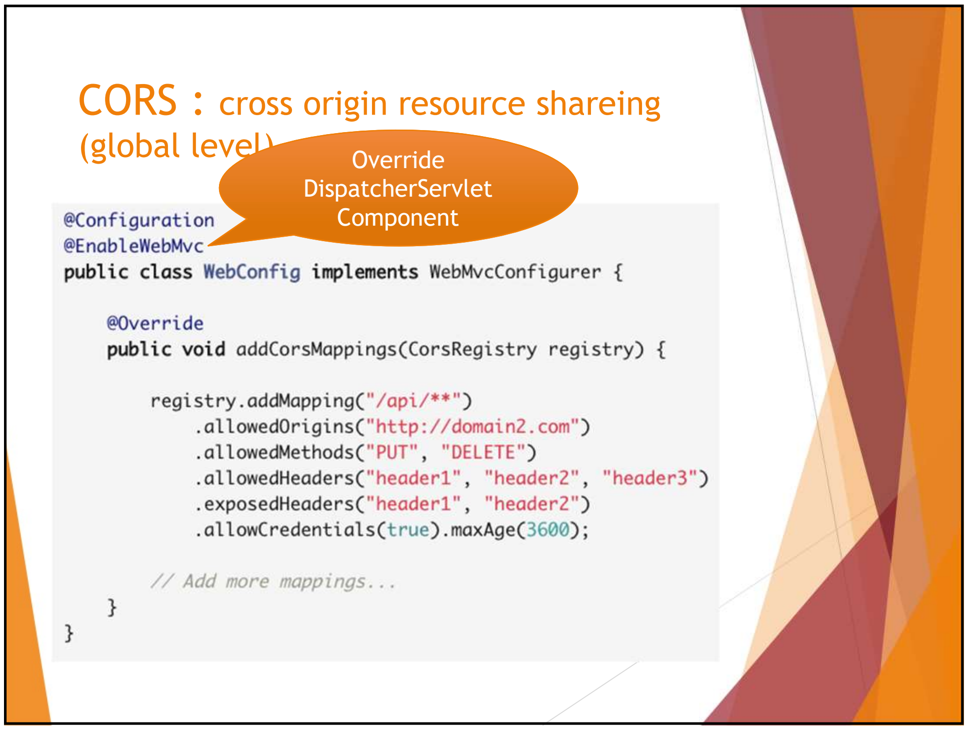
前端是一个独立运行的前端，
后端也是一个独立运行的后端，
前后端不在同一个domain上面

### CORS：cross origin resource sharing (fine-grained class/method level)

```
@RestController
@RequestMapping("/account")
public class AccountController {

    @CrossOrigin
    @GetMapping("/{id}")
    public Account retrieve(@PathVariable Long id) {
        // ...
    }

    @DeleteMapping("/{id}")
    public void remove(@PathVariable Long id) {
        // ...
    }
}
```

```
@CrossOrigin(origins = "http://domain2.com", maxAge = 3600)
@RestController
@RequestMapping("/account")
public class AccountController {

    @GetMapping("/{id}")
    public Account retrieve(@PathVariable Long id) {
        // ...
    }

    @DeleteMapping("/{id}")
    public void remove(@PathVariable Long id) {
        // ...
    }
}
```

15



### CORS：cross origin resource shareing (global level)

Override DispatcherServlet Component

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {

        registry.addMapping("/api/**")
            .allowedOrigins("http://domain2.com")
            .allowedMethods("PUT", "DELETE")
            .allowedHeaders("header1", "header2", "header3")
            .exposedHeaders("header1", "header2")
            .allowCredentials(true).maxAge(3600);

        // Add more mappings...
    }
}
```

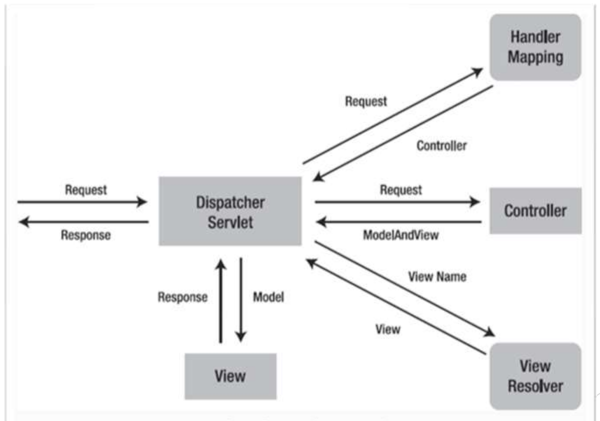
16

## Http Error Code

| Code | Meaning |
|---|---|
| 400 Bad Request | User sent invalid request – syntax or parameter |
| 401 Unauthorized | User is not authenticated |
| 403 Forbidden | User is not allowed/permitted |
| 404 Not Found | User sent wrong request URL |
| 500 Internal Server Error | Server has internal configure error |
| 502 Bad Gateway | Server cannot access certain resource |
| 503 Service Unavailable | Server under maintenance |
| 504 Gateway Timeout | Server not responding in time |

17

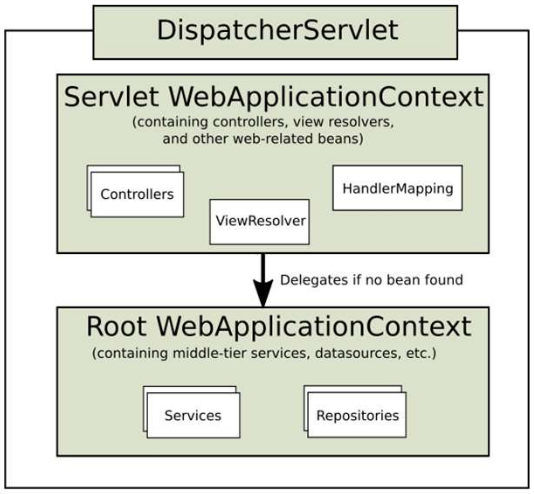
## Spring MVC Fundamentals: DispatcherServlet

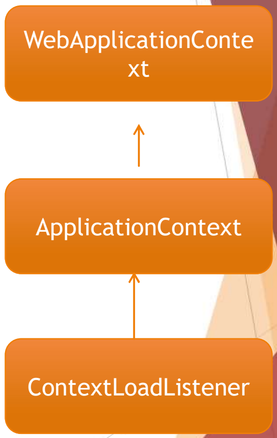


Spring dispatcher servlet

18

## DispatcherServlet


DispatcherServlet
Servlet WebApplicationContext
(containing controllers, view resolvers, and other web-related beans)
Controllers
ViewResolver
HandlerMapping
Delegates if no bean found
Root WebApplicationContext
(containing middle-tier services, datasources, etc.)
Services
Repositories



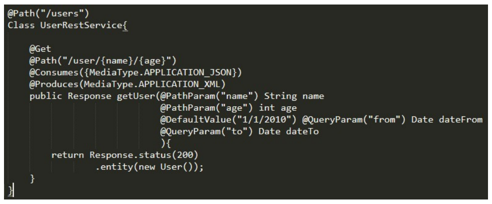
19

## Web.xml

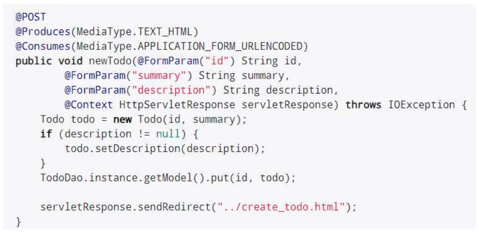
```
<web-app>

  <listener>  <!-Not required for new spring version-→
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/app-context.xml</param-value>
  </context-param>

   <servlet>
    <servlet-name>app</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value></param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>app</servlet-name>
        <url-pattern>/app/*</url-pattern>
  </servlet-mapping>

</web-app>
```

20

## *Jersey Rest Service



```
@Path("/users")
Class UserRestService{

    @Get
    @Path("/user/{name}/{age}")
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces(MediaType.APPLICATION_XML)
    public Response getUser(@PathParam("name") String name
                            @PathParam("age") int age
                            @DefaultValue("1/1/2010") @QueryParam("from") Date dateFrom
                            @QueryParam("to") Date dateTo
                            ){
        return Response.status(200)
                .entity(new User());
    }
}
```

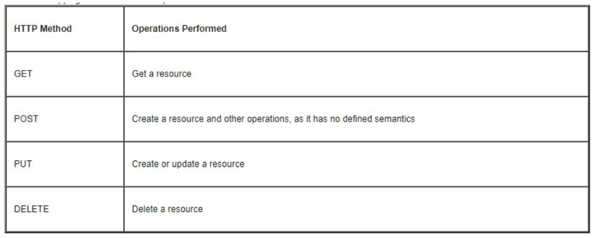


```
@POST
@Produces(MediaType.TEXT_HTML)
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
public void newTodo(@FormParam("id") String id,
        @FormParam("summary") String summary,
        @FormParam("description") String description,
        @Context HttpServletResponse servletResponse) throws IOException {
    Todo todo = new Todo(id, summary);
    if (description != null) {
        todo.setDescription(description);
    }
    TodoDao.instance.getModel().put(id, todo);

    servletResponse.sendRedirect("../create_todo.html");
}
```

Get vs
Post

21

## *Jersey Rest Service



| HTTP Method | Operations Performed |
|---|---|
| GET | Get a resource |
| POST | Create a resource and other operations, as it has no defined semantics |
| PUT | Create or update a resource |
| DELETE | Delete a resource |

22

## *Jersey Rest Service

```
@PUT
@Consumes(MediaType.APPLICATION_XML)
public Response putTodo(JAXBElement<Todo> todo) {
    Todo c = todo.getValue();
    return putAndGetResponse(c);
}

@DELETE
public void deleteTodo() {
    Todo c = TodoDao.instance.getModel().remove(id);
    if(c==null)
        throw new RuntimeException("Delete: Todo with " + id +  " not found");
}
```

Put vs
Delete

23

## *Rest: Spring vs Jersey

| Spring Annotation | JAX-RS Annotation |
|---|---|
| `@RequestMapping(path = "/troopers"` | `@Path("/troopers")` |
| `@RequestMapping(method = RequestMethod.POST)` | `@POST` |
| `@RequestMapping(method = RequestMethod.GET)` | `@GET` |
| `@RequestMapping(method = RequestMethod.DELETE)` | `@DELETE` |
| `@ResponseBody` | N/A |
| `@RequestBody` | N/A |
| `@PathVariable("id")` | `@PathParam("id")` |
| `@RequestParam("xyz")` | `@QueryParam('xyz")` |
| `@RequestParam(value="xyz"` | @FormParam("xyz") |
| `@RequestMapping(produces = {"application/json"})` | `@Produces("application/json")` |
| `@RequestMapping(consumes = {"application/json"})` | `@Consumes("application/json")` |

24