CS544 EA

Applications

Spring MVC: Data Output

Data Output

- There are two main ways to output data:
  - Render a view
    - Several ways to specify a view name
    - Providing it ‘model’ data to render

  For web pages

  - Output an object
    - Use @ResponseBody on return type
    - Message converter transform to desired format
    - View name can be used to specify a transformer

  For web services

Return String View Name

```
@Bean
public ViewResolver viewResolver() {
     InternalResourceViewResolver bean = new InternalResourceViewResolver();

     bean.setViewClass(JstlView.class);
     bean.setPrefix("/WEB-INF/view/");
     bean.setSuffix(".jsp");

     return bean;
}
```

Many other types of ViewResolvers and Views are supported out of the box: Tiles, Velocity, PDF, Excel, Jasper Reports, XSLT

```
@GetMapping(value="/cars/{id}")
public String get(@PathVariable int id, Model model) {
     model.addAttribute("car", carDao.get(id));
     return "carDetail";
}
```

Model is an OUT param

Add data to model

Specify view

What is the name of our view? / Where will Spring MVC look for it?

View

```
<!DOCTYPE html>
<html>
<head><title>Add a Car</title></head>
<body>
  <form action="../cars/${car.id}" method="post">
  <table>
    <tr>
      <td>Make:</td>
      <td><input type="text" name="make" value="${car.make}" /> </td>
    </tr>
    <tr>
      <td>Model:</td>
      <td><input type="text" name="model" value="${car.model}" /> </td>
    </tr>
    <tr>
      <td>Year:</td>
      <td><input type="text" name="year" value="${car.year}" /> </td>
    </tr>
    <tr>
      <td>Color:</td>
      <td><input type="text" name="color" value="${car.color}" /> </td>
    </tr>
  </table>
  <input type="submit" value="update"/>
  </form>
  <form action="delete?carId=${car.id}" method="post">
    <button type="submit">Delete</button>
  </form>
</body>
</html>
```

ModelAndView

```xml
<!-- Resolves views selected for rendering by @Controllers to .jsp resources
     in the /WEB-INF/views directory -->
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />
  <property name="prefix" value="/WEB-INF/views/" />
  <property name="suffix" value=".jsp" />
</bean>
```

```java
@RequestMapping(value="/cars", params="id", method=RequestMethod.GET)
public ModelAndView get(@RequestParam int id) {
  ModelAndView mav = new ModelAndView();
  mav.setViewName("carDetail");
  mav.addObject("car", carDao.get(id));
  return mav;
}
```

ModelAndView is old pre-spring 3

Use .setViewName() not .setView() !

@ModelAttribute

- Critical for form data
  - Especially if you want to show an empty form
  - The view 2 slides ago can only display as empty form with the following code:

```
@GetMapping(value="/addCar")
public String get(@ModelAttribute("car") Car car) {
    return "addCar";
}
```

Places an empty Car Object in the Model with key “car”

Implicit View Name

If you've configured it

- You can omit (not specify) a view name
  - Convention: convert the request URL to view name

```
@Bean
public DefaultRequestToViewNameTranslator defaultRequestToViewNameTranslator() {
     return new DefaultRequestToViewNameTranslator();
}
```

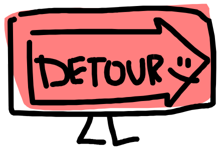
```
<!-- when using XML config -->_
<bean class="org.springframework.web.servlet.view.DefaultRequestToViewNameTranslator" />
```

Can be declared in addition to normal ViewResolver

```
@GetMapping(value="/cars")
public void getAll(Model model) {
  model.addAttribute("cars", carDao.getAll());
}
```

No view name – what view is called?

Redirects



- Redirects are important!
  - After processing POST (input) → **always redirect**!
  - Known as POST/REDIRECT/GET **pattern**
  - Separation of concerns
  - No problem with refresh
  - No duplicate submissions
  - No problem with bookmarks

See: http://en.wikipedia.org/wiki/Post/Redirect/Get

Redirects

```
@PostMapping(value="/cars")
public String add(Car car, Model model) {
  carDao.add(car);
  model.addAttribute("id", car.getId());
  return "redirect:/cars/{id}";
}
```

Redirect can contain URI Template

```
@PostMapping(value = "/list")
public RedirectView addItem(@RequestBody Item item)
{
  shoppingListService.addToList(item);
  return new RedirectView("list");
}
```

Pre Spring 3