中山大學
SUN YAT-SEN UNIVERSITY

《分布式计算》

J2EE 开发入门

学 院 名 称 ： 数据科学与计算机学院

成　　员 ： 陈伟宸

时　　间 ： 2016 年 9 月 17 日

1. Java 2 Platform Enterprise Edition，J2EE 是一种利用 Java 2 平台来简化企业解决方案的开发、部署和管理相关的复杂问题的体系结构，其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构。

2. 这四层模型分别为：

- 运行在客户端机器上的客户层组件
- 运行在 J2EE 服务器上的 Web 层组件
- 运行在 J2EE 服务器上的业务逻辑层组件
- 运行在 EIS 服务器上的企业信息系统(Enterprise information system)层软件

MVC 属于 Web 层，JPA 属于业务逻辑层

3. MVC 由三个组件构成，它们的作用分别是：

- Model（模型）

  模型包含应用程序的核心功能。模型封装了应用程序的状态。有时它包含的唯一功能就是状态。它对视图或控制器一无所知。

- View（视图）

  视图提供模型的表示。它是应用程序的 外观。视图可以访问模型的读方法，但不能访问写方法。此外，它对控制器一无所知。当更改模型时，视图应得到通知。

- Controller（控制器）

控制器对用户的输入作出反应。它创建并设置模型。

4. MVC 模型的执行过程：

HTTP 请求由 web 应用服务器分配（ web.xml ）给框架提供的 Servlet 或 Filter;

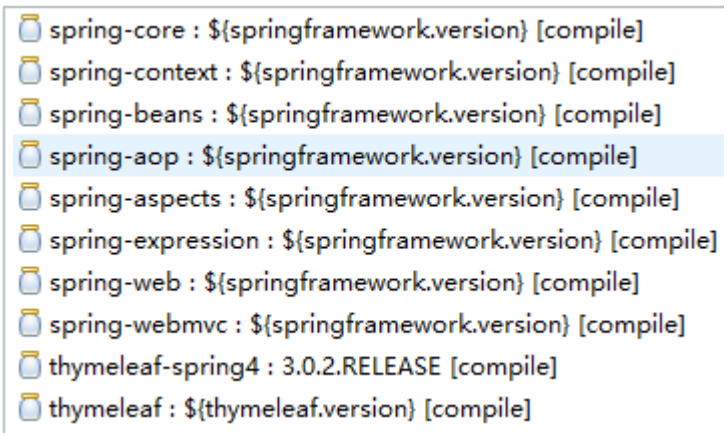
框架将请求按应用程序 注解元数据（annatation）或 如 struts.xml 将请求分派给 URI 对应的处理程序

控制器处理请求输入、验证、从业务组件（层）获取数据、装配数据模型，最后，选择合适输出模板

模板读取数据，产生输出流

5.

（1）在 pox.xml 添加 spring 和 thymeleaf-spring 的相关依赖



（2）在 src/WEB-INF 目录下新建 spring-gtvg-servlet.xml，在其中配置好模板引擎和模板解析器等

```
<!-- scan the package and the sub package -->
<context:component-scan base-package="thymeleafexamples.gtvg"/>
<!-- don't handle the static resource -->
<mvc:default-servlet-handler />
<!-- if you use annotation you must configure following setting -->
<mvc:annotation-driven />

<bean id="templateResolver"
   class="org.thymeleaf.templateresolver.ServletContextTemplateResolver">
  <property name="prefix" value="/WEB-INF/templates/" />
  <property name="suffix" value=".html" />
</bean>

<bean id="templateEngine"
      class="org.thymeleaf.spring4.SpringTemplateEngine">
  <property name="templateResolver" ref="templateResolver" />
</bean>

<bean class="org.thymeleaf.spring4.view.ThymeleafViewResolver">
  <property name="templateEngine" ref="templateEngine" />
</bean>
```

（3）在 web.xml 中配置好 org.springframework.web.servlet.DispatcherServlet

```
<servlet>
    <servlet-name>spring-gtvg</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

（4）使用 @Controller 和 @RequestMapping 修改所有的 controller

HomeController

```
@Controller
public class HomeController {
    @RequestMapping("/")
    public String home(Model model) {
        model.addAttribute("Today", Calendar.getInstance());
        return "home";
    }
}
```

OrderDetailsController

```java
@Controller
public class OrderDetailsController  {
    @RequestMapping("/order/details")
    public String orderDetails(@RequestParam(value="orderId") int orderId, Model model) {
        final OrderService orderService = new OrderService();
        final Order order = orderService.findById(orderId);
        model.addAttribute("order", order);
        return "order/details";
    }
}
```

OrderListController

```java
@Controller
public class OrderListController {
    @RequestMapping("order/list")
    public String orderList(Model model) {
        final OrderService orderService = new OrderService();
        final List<Order> allOrders = orderService.findAll();
        model.addAttribute("orders", allOrders);
        return "order/list";
    }
}
```

ProductCommentsController

```java
@Controller
public class ProductCommentsController {
    @RequestMapping("product/comments")
    public String productComments(@RequestParam(value="prodId") int prodId, Model model) {
        final ProductService productService = new ProductService();
        final Product product = productService.findById(prodId);
        model.addAttribute("prod", product);
        return "product/comments";
    }
}
```

ProductListController

```java
@Controller
public class ProductListController {
    @RequestMapping("product/list")
    public String productList(Model model) {
        final ProductService productService = new ProductService();
        final List<Product> allProducts = productService.findAll();
        model.addAttribute("prods", allProducts);
        return "product/list";
    }
}
```

SubscribeController

```
@Controller
public class SubscribeController {
    @RequestMapping("/subscribe")
    public String subscribe(Model model) {
        return "subscribe";
    }
}
```

UserProfileController

```
@Controller
public class UserProfileController {
    @RequestMapping("/userprofile")
    public String userProfile(Model model) {
        return "userprofile";
    }
}
```

（5）运行后发现，引擎无法解析

#aggregates.sum(o.orderLines.{purchasePrice * amount})等类似语句，于是在 Order 中增加一个 getTotal()函数用于计算总数

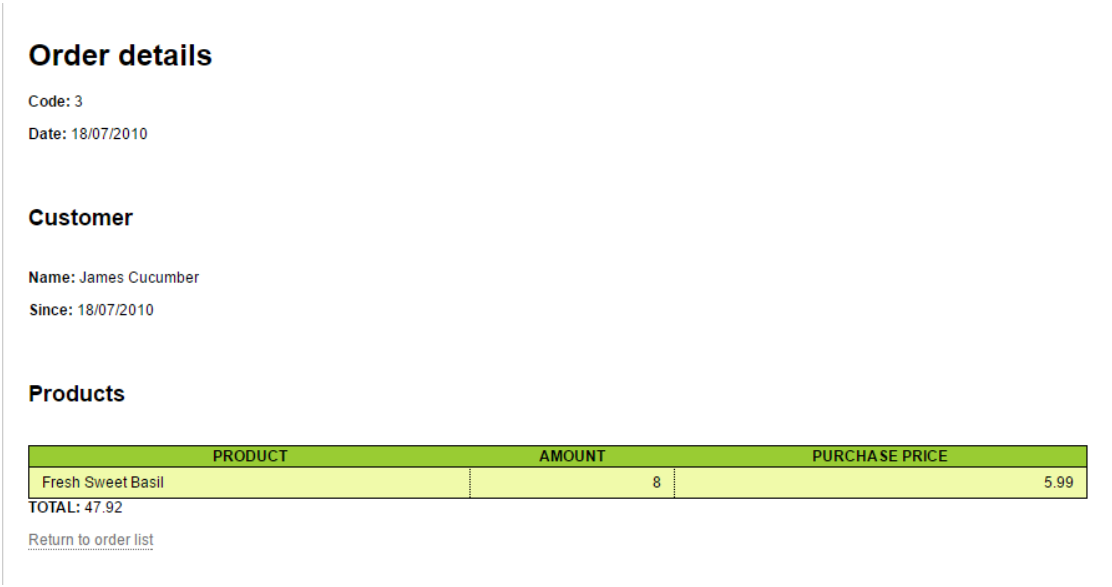
```
public BigDecimal getTotal() {
    BigDecimal total = new BigDecimal(0);
    for (OrderLine orderLine : getOrderLines())
        total = total.add(orderLine.getPurchasePrice().multiply(new BigDecimal(orderLine.getAmount())));
    return total;
}
```

并在 html 模板中将相应字段为：

```
<div>
  <b>TOTAL:</b> <span th:text="${order.getTotal()}">35.23</span>
</div>

<p>
```

运行结果：



6. @PostMapping 和 @RequestMapping(method = RequestMethod.POST)

7. 可以，需要在控制器中指定使用的 HTML 文件或直接在控制器中编写好使用的 HTMP 文件

8. WebApplicationInitializer

10. 对于 IBA JPA 需要留意如何继承接口，如何创建查询，对于 Spring Thyme Seed Starter Manager，需要留意@Controller 等元数据的使用方式，已经在 HTML 中的 thymeleaf 引擎的语法

小结

1. 学习了 MVC 设计模式
2. 学习了 Spring 的 mvc 框架
3. 学习了 Thymeleaf 模板引擎
4. 学习了 Spring 与 Thymeleaf 的集成
5. 即使都是最新版，没有集成 Spring 的 Thymeleaf 与集成了 Spring 的 Thymeleaf 对 HTML 模板的解析引擎并不完全相同，在移植时有时要根据需要修改模板