Spring MVC 技术分享

服务平台-工单组 王君明

目录

目录

Contents

1

MVC设计思想

MVC思想-标准MVC

1、发送请求

控制器C

2、改变状态

3、返回数据

模型M

5、选择视图

视图V

4、数据变化
触发

6、视图渲染

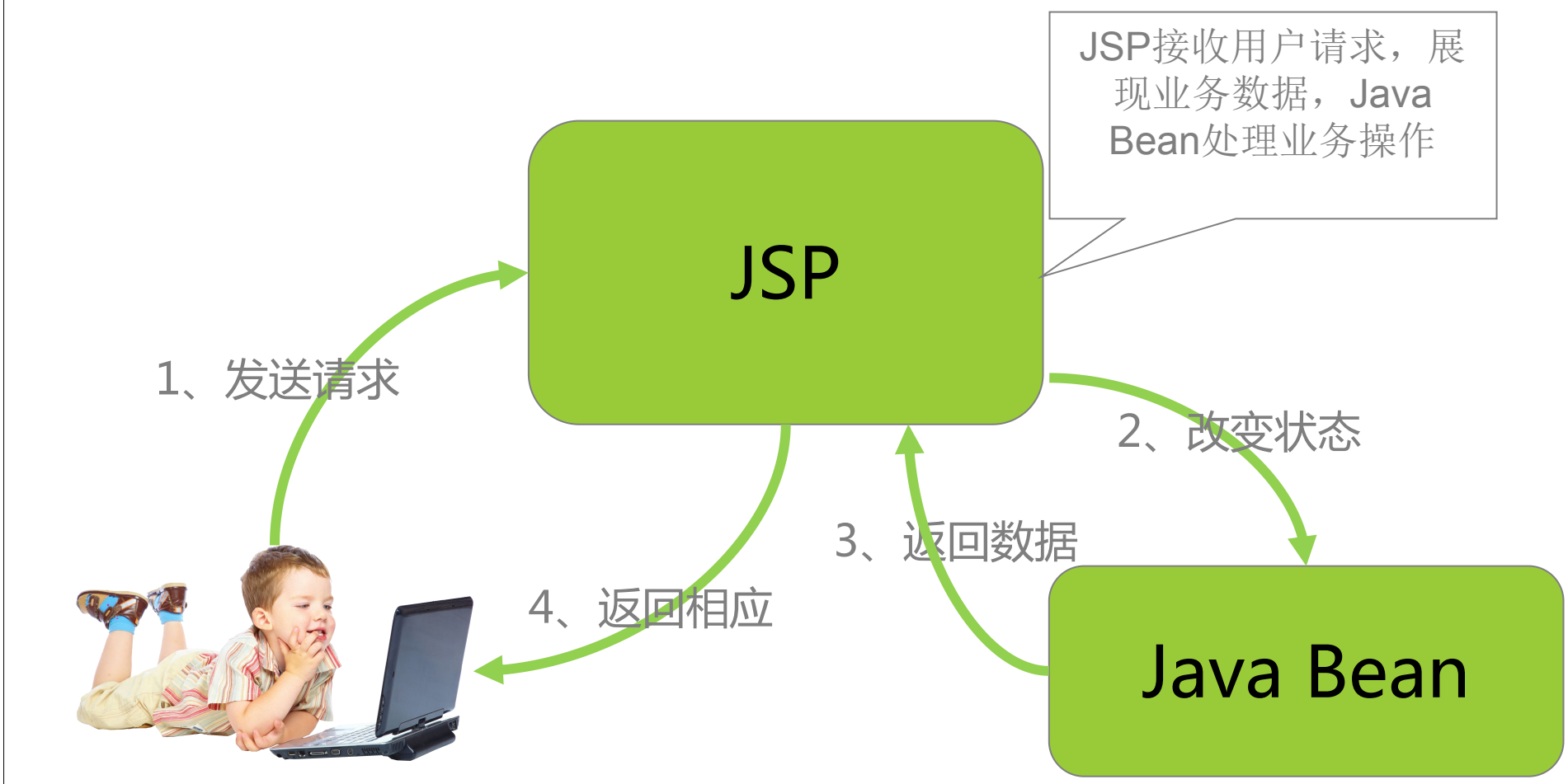
MVC设计思想-Java Web发展历程

Servlet

JSP → Model1

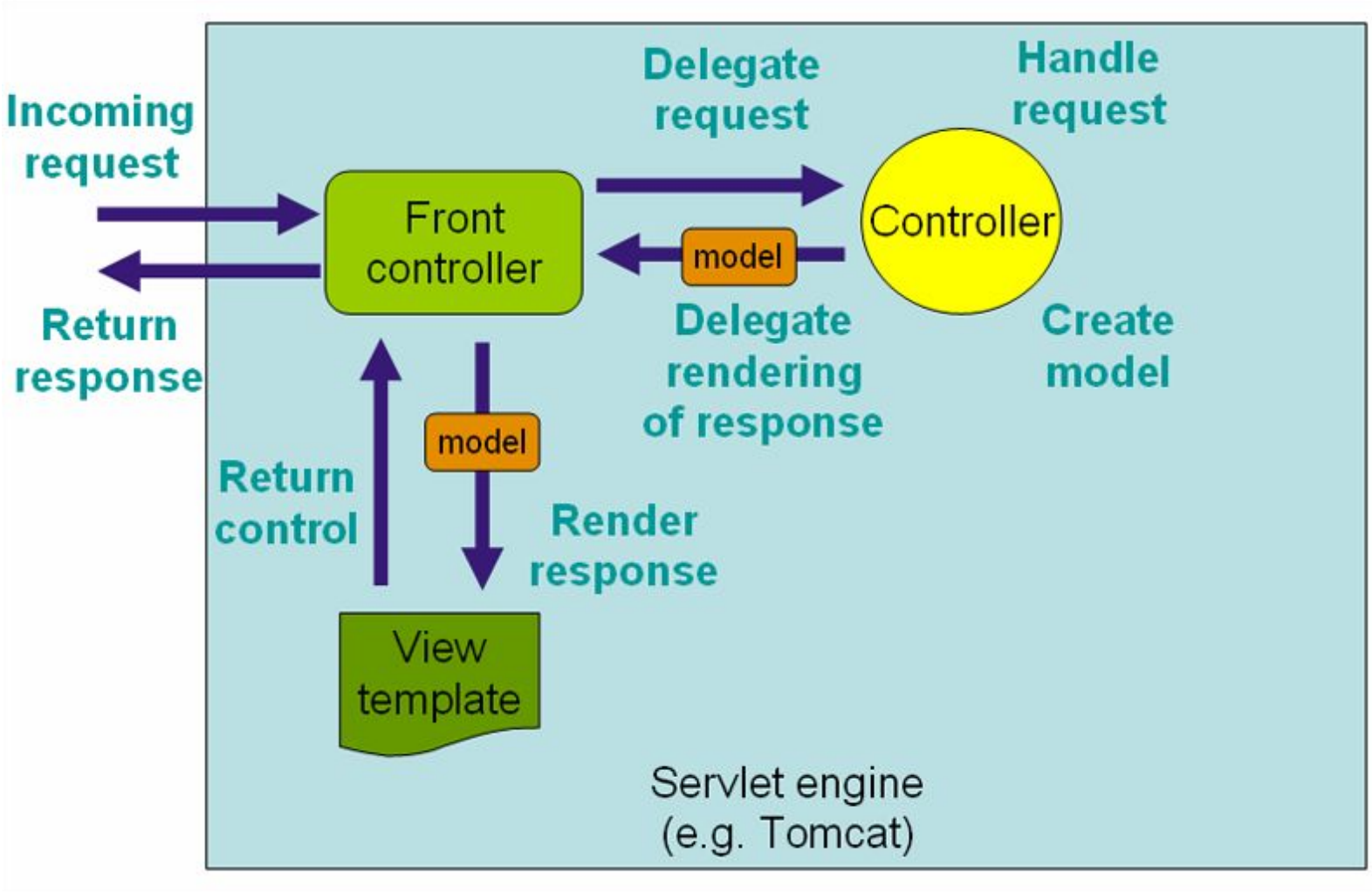
Model 2
(Web MVC) → Front controller
Page controller

MVC设计思想-Model 1



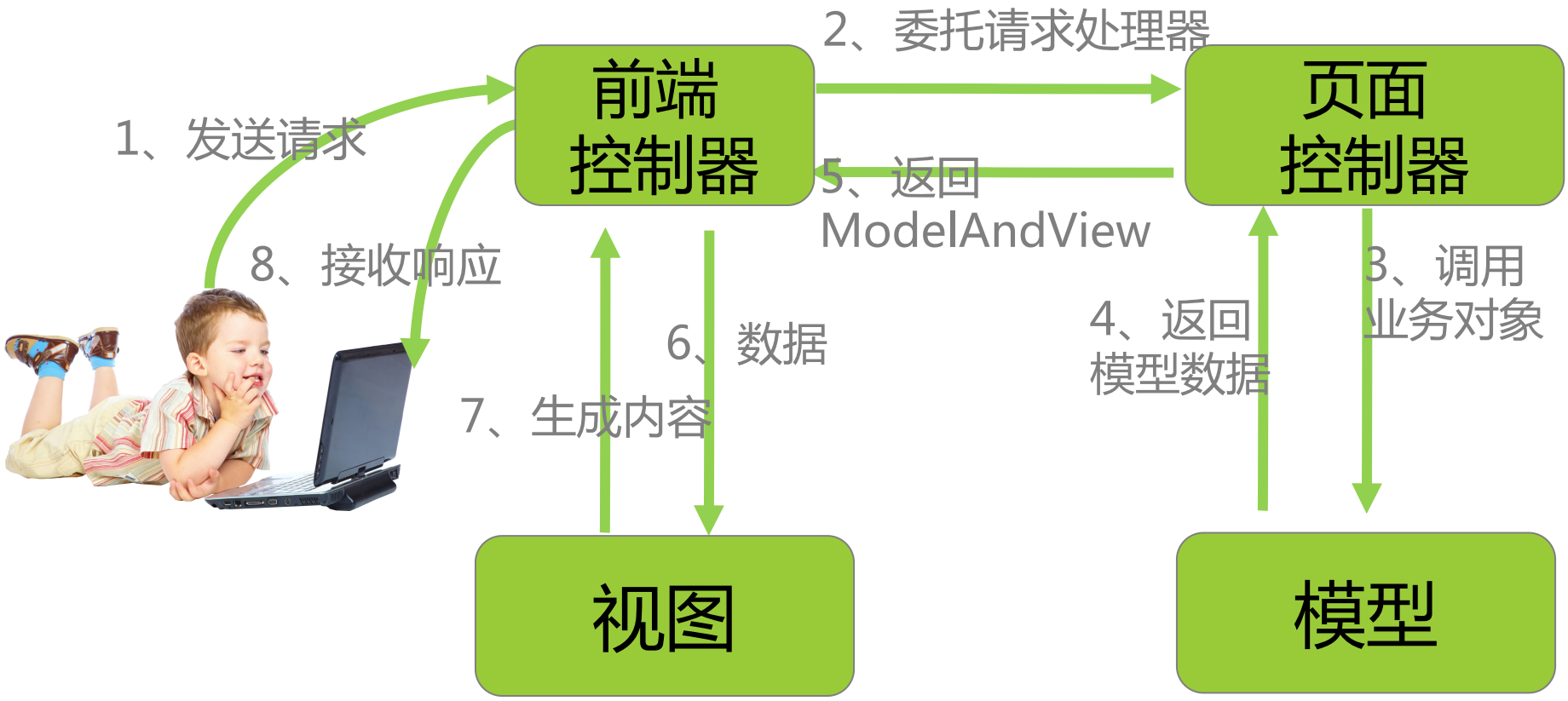
MVC设计思想-Model 2(Web MVC)

MVC设计思想-Spring MVC



The request processing workflow in Spring Web MVC (high level)

MVC设计思想-Spring MVC



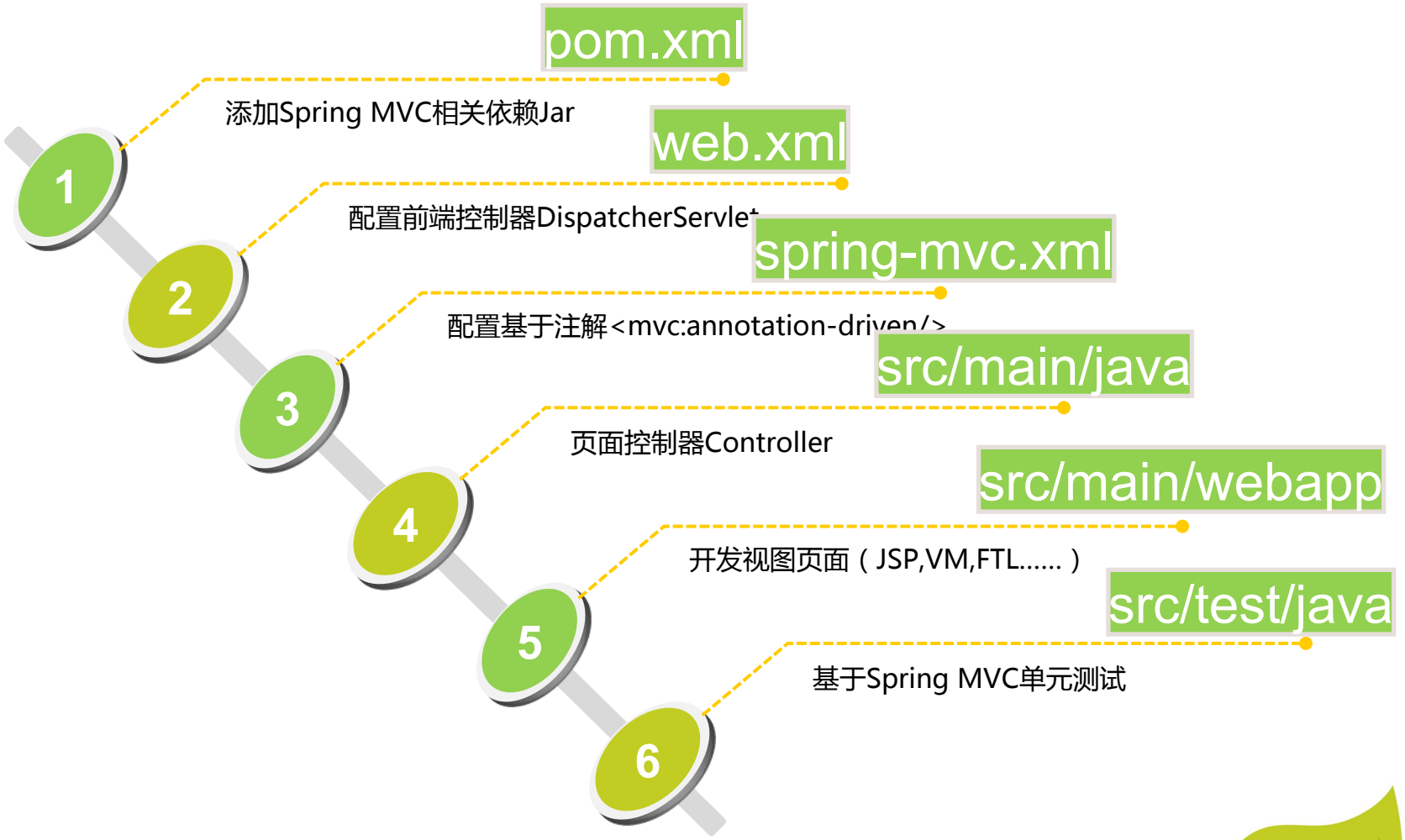
MVC设计思想-Spring MVC优势

编号	优势	描述
1	角色划分清晰	前端控制器（DispatcherServlet） 处理器映射（HandlerMapping） 处理器适配器（HandlerAdapter） 视图解析器（ViewResolver）,视图(View) 页面控制器（Controller）业务处理
2	扩展灵活	可以方便扩展各个角色，增强使用，通过HandlerAdapter可以支持任意的类作为处理器
3	Spring嫡系MVC	Spring在java世界应用太广泛。
4	方便单元测试	利用Spring提供的Mock对象能够非常简单的进行Web层单元测试
5	多视图集成	非常容易与其他视图技术集成，如Velocity、FreeMarker等等

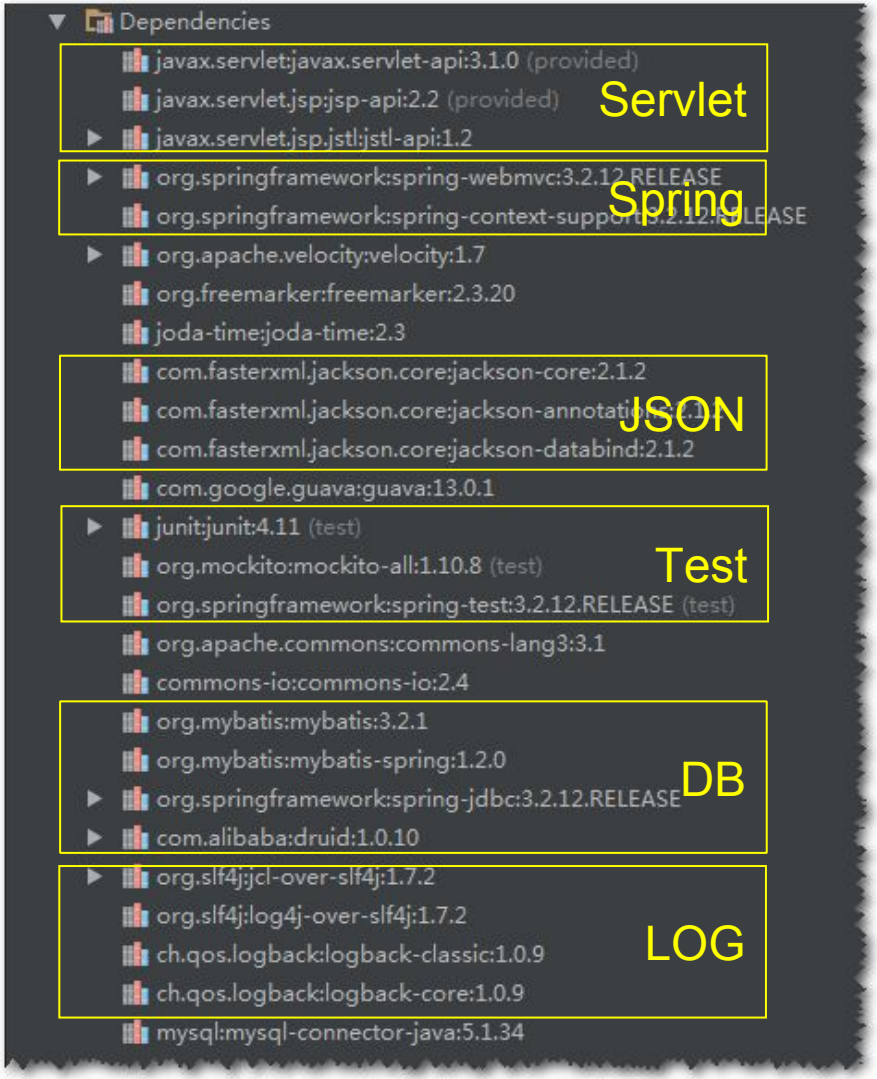
二

2 Spring MVC开发实践

Spring MVC开发实践



Spring MVC开发实践-POM



添加配置依赖

必选、可选依赖根据功能使用，推荐最小配置。

```
<plugin>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>9.2.5.v20141112</version>
    <configuration...>
</plugin>
```

推荐配置jetty插件

可以 mvn jetty:run 启动运行

Spring MVC开发实践-web.xml

```
28    <servlet>
29        <servlet-name>showcase</servlet-name>
30        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
31        <init-param>
32            <param-name>contextConfigLocation</param-name>
33            <param-value>
34                classpath*:/spring/mvc/mvc.xml
35            </param-value>
36        </init-param>
37        <load-on-startup>1</load-on-startup>
38    </servlet>
39    <servlet-mapping>
40        <servlet-name>showcase</servlet-name>
41        <url-pattern>/showcase/*</url-pattern>
42    </servlet-mapping>
```

Servlet3.x 中 web.xml 也不是必须的了！

```
21    <listener>
22        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
23    </listener>
24    <context-param>
25        <param-name>contextConfigLocation</param-name>
26        <param-value>classpath*:spring/app.xml</param-value>
27    </context-param>
```

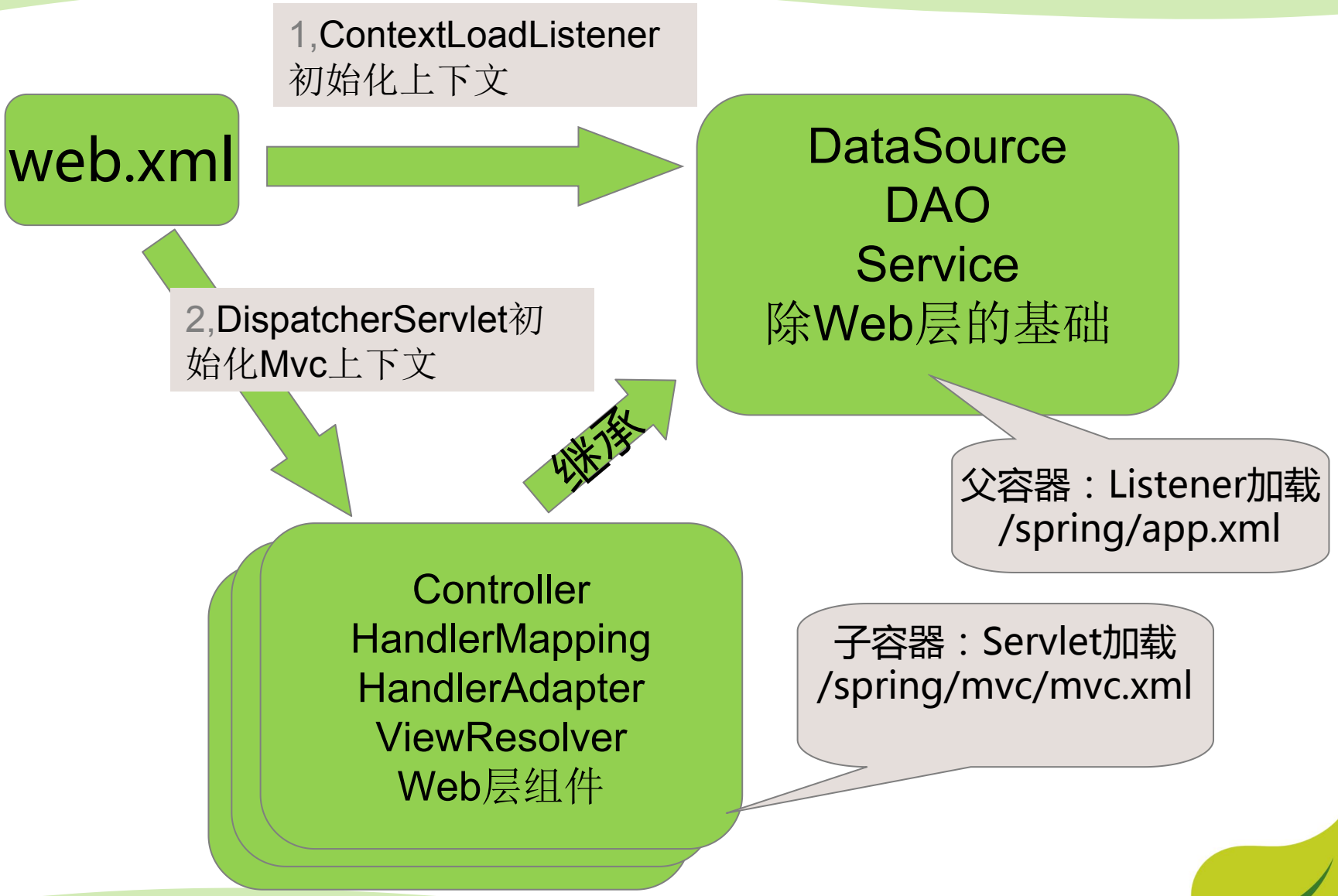
Spring MVC开发实践-code base Servlet

```
26  public class DemoWebApplicationInitializer implements WebApplicationInitializer {
27
28      public void onStartup(ServletContext servletContext) throws ServletException {
29          XmlWebApplicationContext context = new XmlWebApplicationContext();
30          context.setConfigLocation("classpath*:/spring/mvc/mvc.xml");
31          ServletRegistration.Dynamic registration =servletContext
32                  .addServlet("dispatcher",new DispatcherServlet(context));
33          registration.setLoadOnStartup(1);
34          registration.addMapping("/showcase/*");
35      }
36  }
```

Servlet 3.0以后 web 容器本身支持基于Java代码的Servlet初始化

SpringServletContainerInitializer
ServletContainerInitializer
@HandlesTypes

Spring MVC开发实践-Context Hierarchy



Spring MVC开发实践-mvc.xml

```
13      <!-- mvc 注解驱动配置 使用默认的 HandlerMapping 和 HandlerAdapter -->
14      <mvc:annotation-driven>
15          <mvc:argument-resolvers>
16              <bean class="org.jimlgx.showcase.spring.support.RowBoundsArgumentResolver"></bean>
17          </mvc:argument-resolvers>
18      </mvc:annotation-driven>
19      <!-- 扫描 Controller 的包结构 -->
20      <context:component-scan base-package="org.jimlgx.showcase.spring.web.**"/>
21      <!-- 自动ClassName 映射 如：WelcomeController 映射为 /welcome -->
22      <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping"></bean
23      <!--映射默认请求到 welcome-->
24      <mvc:view-controller path="/" view-name="welcome"/>
25
```

14~18:mvc注解驱动 why mvc:?
16:自定义的参数解析器,解析RowBounds
20:扫描加载业务Controller的包前缀
22:扩展其它的HandlerMapping
24:定义controller路径映射

schema/mvc/spring-mvc.xsd

spring-webmvc:jar
/META-INF/spring.handlers
spring.schemas

MvcNamespaceHandler

Spring MVC开发实践-mvc-vm.xml

```
8   <bean id="velocityConfig" class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">
9       <property name="resourceLoaderPath" value="/WEB-INF/vm/"></property>
10      <property name="velocityProperties">
11          <props>
12              <prop key="input.encoding">UTF-8</prop>
13              <prop key="output.encoding">UTF-8</prop>
14          </props>
15      </property>
16  </bean>
17  <bean class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">
18      <property name="contentType" value="text/html;charset=UTF-8"/>
19      <property name="cache" value="true"/>
20      <property name="prefix" value=""/>
21      <property name="suffix" value=".vm"/>
22  </bean>
```

视图解析器Velocity

8~16:Velocity配置项，vm模板加载的位置
17~22:VelocityViewResolver配置

Spring MVC开发实践-WelcomeController

```
25  @org.springframework.stereotype.Controller("/welcome")
26  public class WelcomeController implements Controller{
27
28      @Override
29      public ModelAndView handleRequest(HttpServletRequest request,
30                                        HttpServletResponse response) throws Exception {
31          ModelAndView modelAndView = new ModelAndView();
32          String plaint = "Welcome to here,enjoy spring mvc 分享 !";
33          PlaintStringView view = new PlaintStringView(plaint);
34          modelAndView.setView(view);
35          return modelAndView;
36      }
37  }
```

实现接口的"欢迎Controller"

25:配置注解beanName为"/welcome"，可以访问/welcome
26:一种古老的方式，实现Controller的接口
29:输入request和response,输出ModelAndView
33:创建一个自定义的PlaintStringView，输出纯文本的中英混合体

DemoModelController

```
45  @Controller
46  @RequestMapping("/demo")
47  public class DemoModelController {
48      /**...*/
58      @RequestMapping(value = "/list",
59              method = {RequestMethod.GET},
60              consumes = {MediaType.ALL_VALUE},
61              produces = {MediaType.TEXT_HTML_VALUE})
62      @ModelAttribute("list")
63      public List<DemoModel> listView(@RequestParam Map<String,String> map,
64                                      RowBounds rowBounds) {...}
```

熟悉的注解方式Controller

45~46:你懂的·
58:定义改方法的URL: /demo/list
59~61:Mapping映射范围缩小的条件
62:声明Model中会有一个key=list, value是List<DemoModel>的对象
63:请求参数自动绑定到map中了
64:RowBounds很熟悉，也可以作为参数了吗？

Spring MVC开发实践-Controller

```
83    /**...*/
91    @RequestMapping(value = "/", method = {RequestMethod.POST})
92    @ResponseBody
93    public DemoModel save(@ModelAttribute("demo")DemoModel model) { return demoModelService.save(model); }
96    /**...*/
104   @RequestMapping(value = "/", method = {RequestMethod.GET},produces = {MediaType.APPLICATION_JSON_VALUE
105   @ResponseBody
106   public List<DemoModel> list(RowBounds rowBounds) {...}
113   /**...*/
124   @RequestMapping(value = "/{id}", method = {RequestMethod.GET})
125   @ResponseBody
126   public DemoModel get(@PathVariable(value = "id") Long id) { return demoModelService.findById(id); }
129   /**...*/
137   @RequestMapping(value = "/{id}", method = {RequestMethod.PUT})
138   @ResponseBody
139   public DemoModel udate(@PathVariable(value = "id") Long id,@ModelAttribute("demo")DemoModel model) {..
144   /**...*/
152   @RequestMapping(value = "/{id}", method = {RequestMethod.DELETE})
153   @ResponseBody
154   public int delete(@PathVariable(value = "id") Long id) {...}
```

POST 创建

GET 列表

GET 对象

PUT 修改

DELETE 删除

支持Restful风格Controller

CRUD操作

Spring MVC开发实践-list.vm

```
<bean id="velocityConfig" class="org.springframework.web.servlet.view.velocity.Vel
    <property name="resourceLoaderPath" value="/WEB-INF/vm/"></property>
    <property name="velocityProperties"...>
</bean>
<bean class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">
    <property name="contentType" value="text/html;charset=UTF-8"/>
    <property name="cache" value="true"/>
    <property name="prefix" value=""/>
    <property name="suffix" value=".vm"/>
</bean>
```

Velocity模板视图

请求 /demo/list 经过解析 -> /WEB-INF/vm/demo/list.vm

Spring MVC开发实践-Junit

```
32  @ContextConfiguration(locations = {
33          "classpath:spring/mockito_service.xml",
34          "classpath:spring/mvc/mvc-vm.xml",
35          "classpath:spring/mvc/mvc.xml"})
36  @WebAppConfiguration
37  public class DemoModelControllerTest extends AbstractJUnit4SpringContextTests {
38      private static final Logger LOG = LoggerFactory
39              .getLogger(DemoModelControllerTest.class);
40      @Autowired
41      private WebApplicationContext wac;
42      @Autowired
43      private DemoModelService demoModelService;
44      private MockMvc mockMvc;
45      @Before
46      public void setUp() {
47          this.mockMvc = MockMvcBuilders.webAppContextSetup(this.wac).build();
48      }
49      @Test
```

Spring MVC Test

构建MockMvc对象，按需加载需要的spring/**.xml

Spring MVC开发实践-Junit

```
51      @Test
52      public void testList1() throws Exception {
53          //Mock service服务
54          List<DemoModel> list = getDemoModels();
55          RowBounds rowBounds = RowBoundsArgumentResolver.getRowBounds();
56          Mockito.when(demoModelService.findAll(rowBounds)).thenReturn(list);
57          //Mock request请求
58          MockHttpServletRequestBuilder builder = get("/demo/list");
59          builder.param("name","Qunar");
60          builder.param("offset","0");
61          builder.param("limit","5");
62          builder.accept(MediaType.TEXT_HTML);
63          ResultActions ac = mockMvc.perform(builder);
64          //输出测试信息
65          ac.andDo(print());
66          //断言校验
67          ac.andExpect(status().isOk());
68          ac.andExpect(MockMvcResultMatchers.view().name("demo/list"));
69          ac.andExpect(MockMvcResultMatchers.model().attributeExists("list"));
70      }
```

Mock Service

Mock Request

Spring MVC Test

请求 /demo/list 借助Mock对象执行单元测试

Spring MVC开发实践-Junit

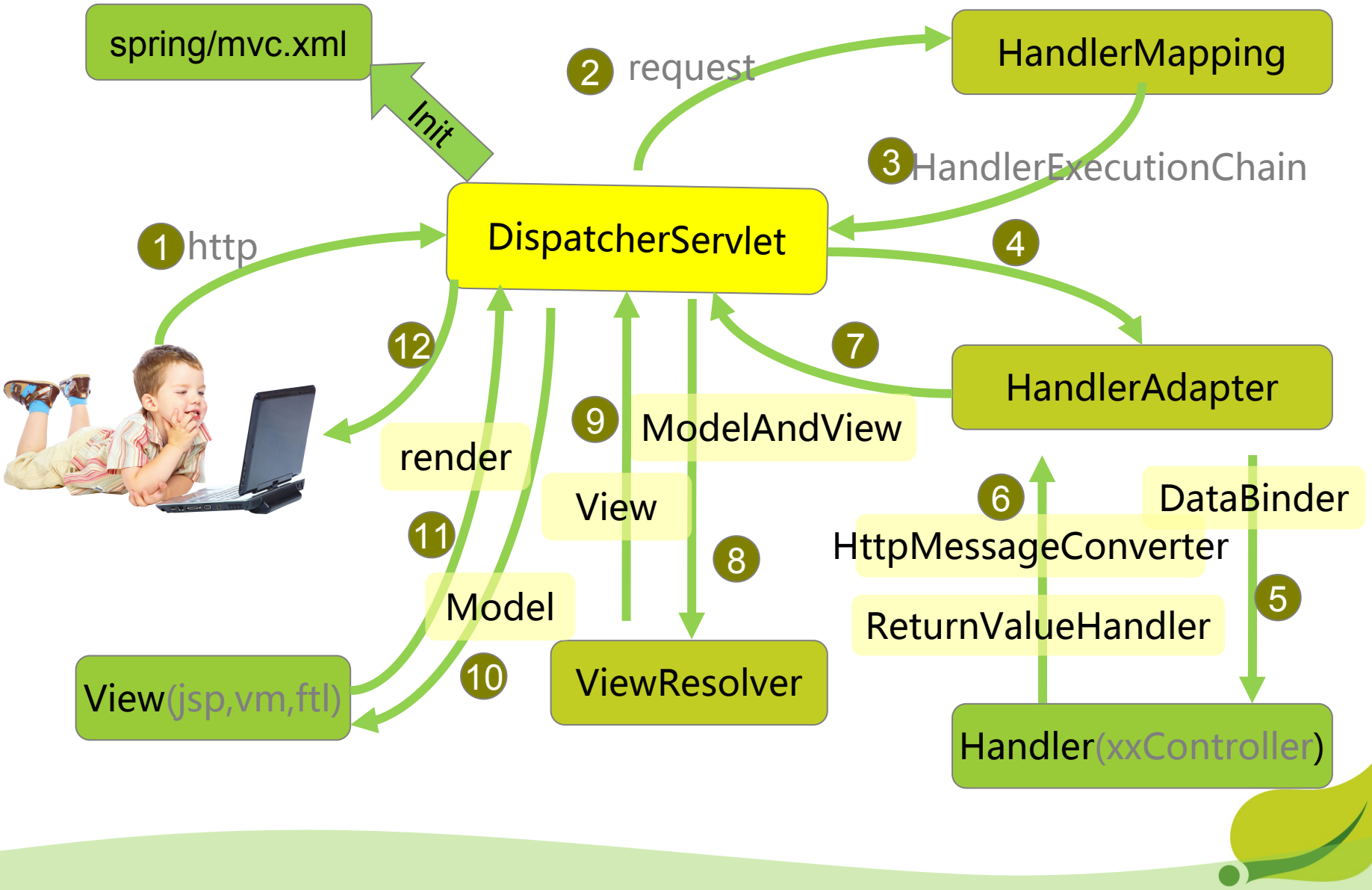
```
MockHttpServletRequest:
          HTTP Method = GET
          Request URI = /demo/list
           Parameters = {name=[Qunar], offset=[0], limit=[5]}
              Headers = {Accept=[text/html]}

        ModelAndView:
            View name = demo/list
                 View = null
            Attribute = list

MockHttpServletResponse:
               Status = 200
        Error message = null
              Headers = {Content-Type=[text/html;charset=UTF-8]}
         Content type = text/html;charset=UTF-8
                 Body = <h3>list.vm showcase</h3>
                        <h4>hello 世界！ </h4>
                        <p>
                            <li>testName</li>
                            <li>testName</li>
                            <li>testName</li>
                            <li>testName</li>
                            <li>testName</li>
                        </p>
```

Body Html

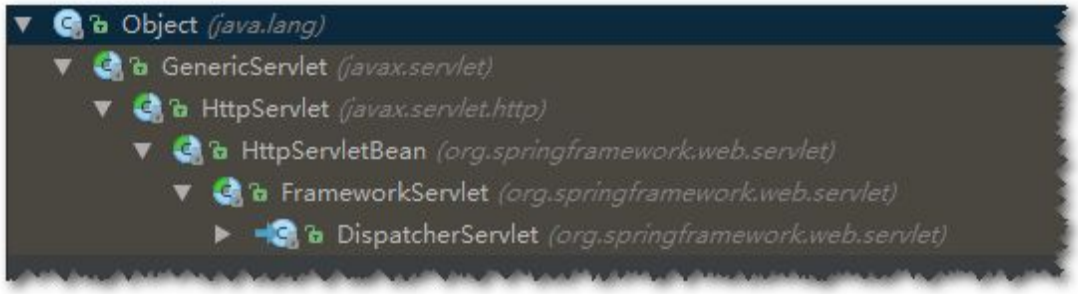
三

3 Spring MVC核心组件

Spring MVC 请求流程



DispatcherServlet-初始化




```
437     protected void initStrategies(ApplicationContext context) {
438         initMultipartResolver(context);
439         initLocaleResolver(context);
440         initThemeResolver(context);
441         initHandlerMappings(context);
442         initHandlerAdapters(context);
443         initHandlerExceptionResolvers(context);
444         initRequestToViewNameTranslator(context);
445         initViewResolvers(context);
446         initFlashMapManager(context);
447     }
448
```

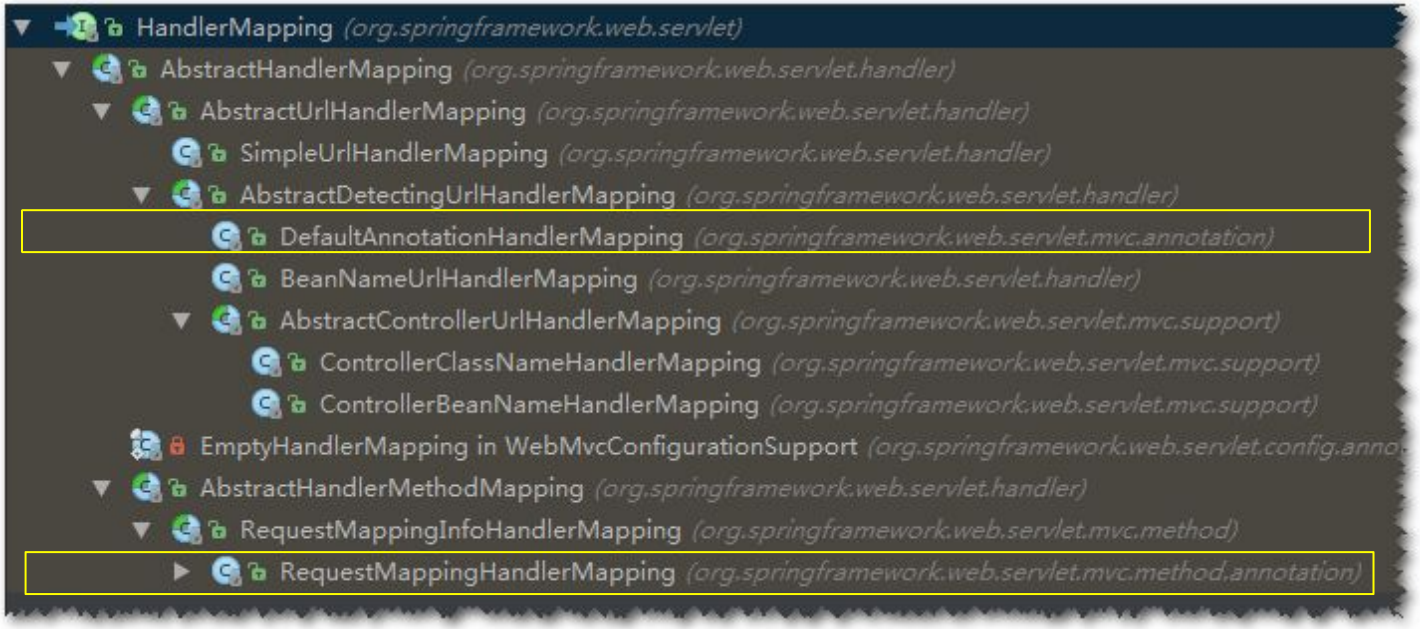
MVC

HandlerMapping　HandlerAdapter　ViewResolver

DispatcherServlet-请求处理

doService	快照request.attribute数据,加入mvc相关对象引用 执行doDispatch()调度 restoreAttributes恢复request数据
doDispatch	输入request获取HandlerExecutionChain及HandlerAdapter 拦截器preHandler() 执行handler获取ModelAndView 执行拦截器postHandler 执行processDispatchResult()处理结果
processDispatchResult	校验是执行否存在异常，修正ModelAndView render渲染ModelAndView

Spring MVC 核心组件-HandlerMapping



HandlerMapping

HandlerExecutionChain getHandler(HttpServletRequest request) throws Exception;

HandlerInterceptior[] interceptors
//ServletInvocableHandlerMethod
Object handler

HandlerInterceptor

```
25      <mvc:interceptors>
26          <mvc:interceptor >
27              <mvc:mapping path="/demo/*"/>
28              <bean class="org.jimlgx.showcase.spring.support.DurationHandlerInterceptor"></bean>
29          </mvc:interceptor>
30      </mvc:interceptors>
```

DurationHandlerInterceptor

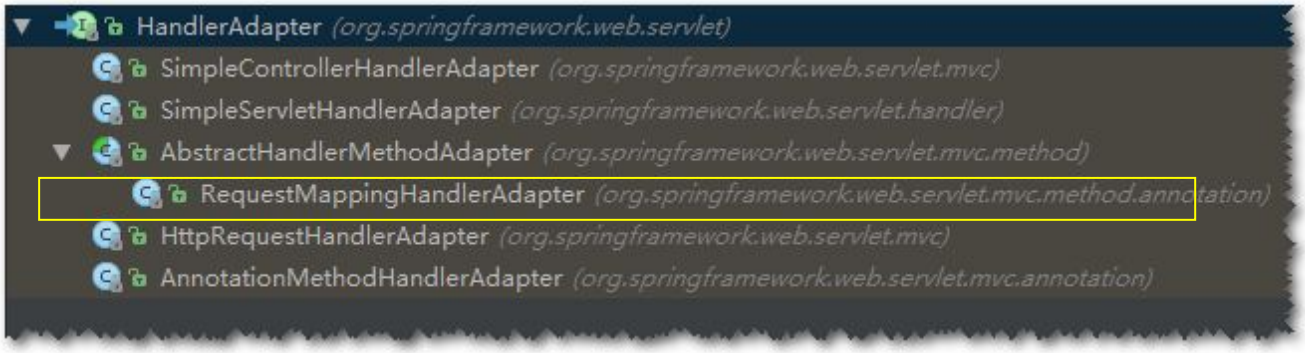
preHandle　postHandle　afterCompletion

执行时间统计的拦截器

该类实现HandlerInterceptor接口或继承HandlerInterceptorAdapter适配类

稍后看java code

Spring MVC 核心组件-HandlerAdapter



HandlerAdapter

```
boolean supports(Object handler);
ModelAndView handle(
            HttpServletRequest request,
            HttpServletResponse response, Object handler) throws Exception;
long getLastModified(HttpServletRequest request, Object handler);
```

HandlerExceptionResolver

HandlerExceptionResolver

```
ModelAndView resolveException(
        HttpServletRequest request, HttpServletResponse response,
        Object handler, Exception ex);
```

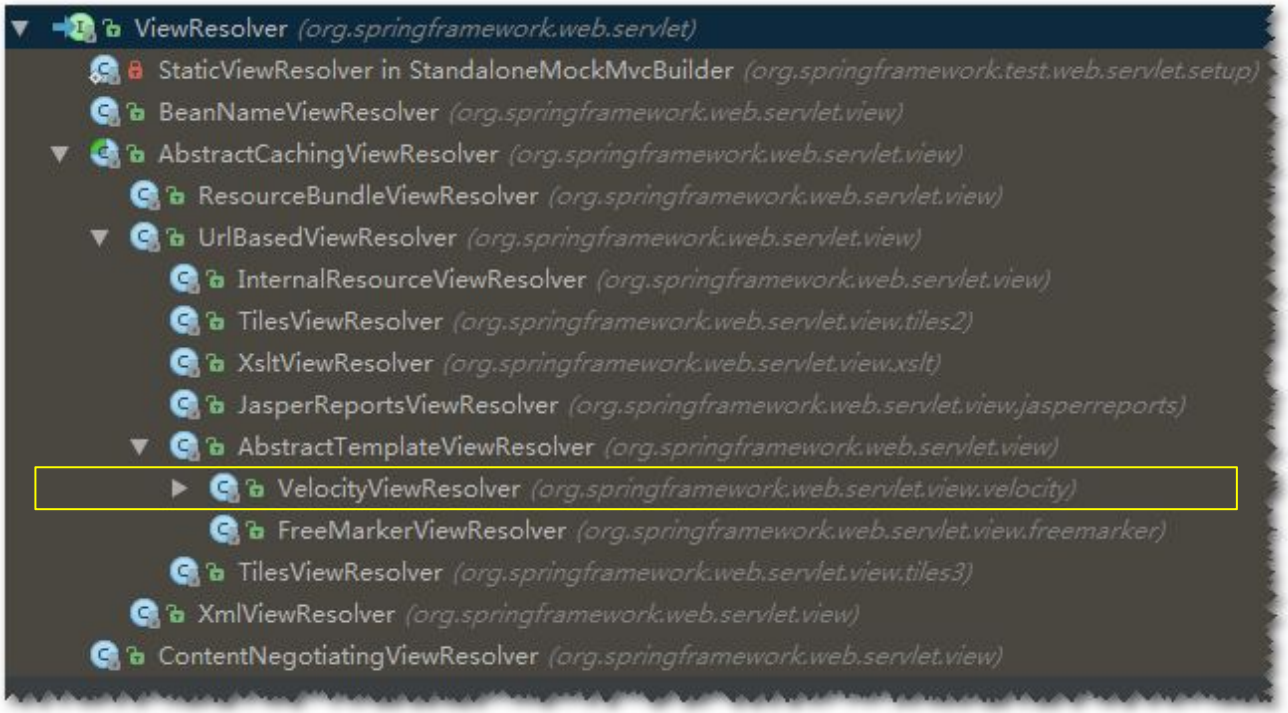
@ExceptionHandler

```
264     @ExceptionHandler({ Exception.class, RuntimeException.class })
265     @ResponseBody
266     public ResponseEntity<String> exceptionHandler(Exception e) {
267         LOG.warn(e.getMessage(), e);
268         ResponseEntity<String> entity = new ResponseEntity<String>("some error",HttpStatus.OK);
269         return entity;
270     }
271
```

@ExceptionHandler

当Controller中发生注解指定的异常时执行该方法渲染数据到Response相应

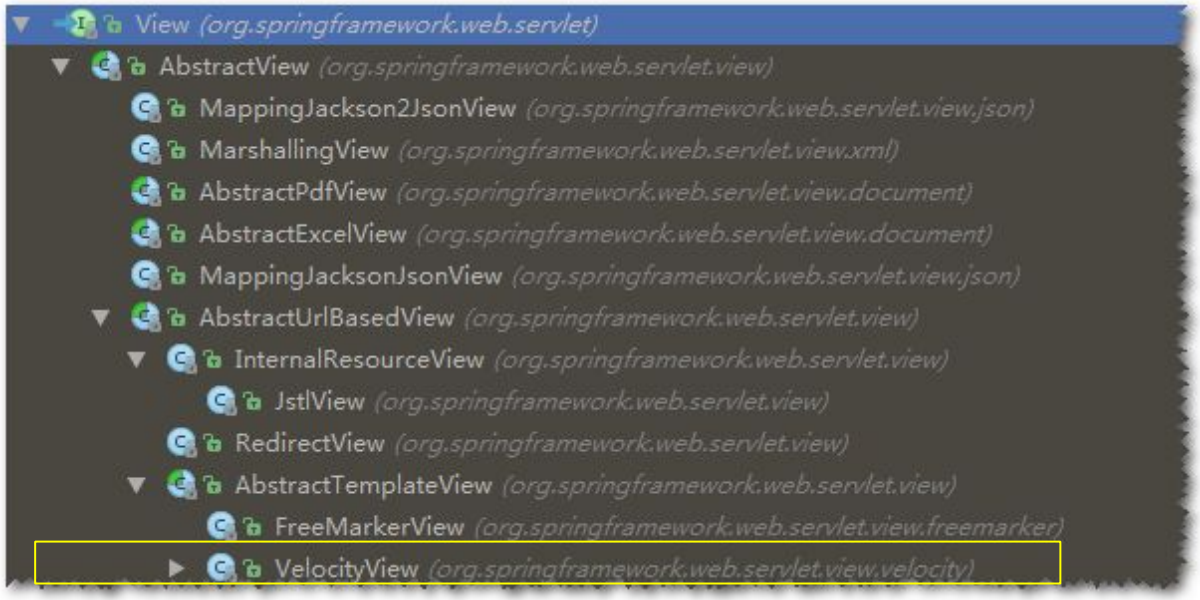
Spring MVC 核心组件-ViewResolver



ViewResolver

```
View resolveViewName(
        String viewName, Locale locale) throws Exception;
```

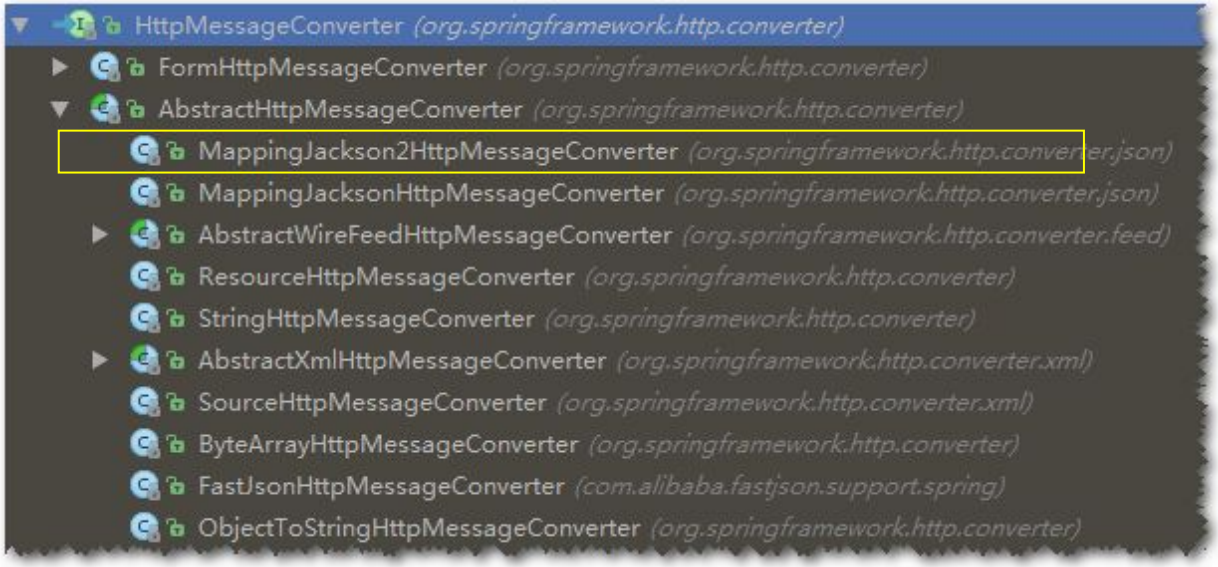
Spring MVC 核心组件-View



View

```
void render(
        Map<String, ?> model,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception;
```

HttpMessageConverter



HttpMessageConverter<T>

```
boolean canRead(Class<?> clazz, MediaType mediaType);
boolean canWrite(Class<?> clazz, MediaType mediaType);
List<MediaType> getSupportedMediaTypes();
T read(Class<? extends T> clazz, HttpInputMessage inputMessage)
        throws IOException, HttpMessageNotReadableException;
void write(T t, MediaType contentType, HttpOutputMessage
        outputMessage)throws IOException, HttpMessageNotWritableException;
```

HandlerMethodArgumentResolver

参数描述	支持的实现类
@ModelAttribute	ModelAttributeMethodProcessor
@RequestParam	RequestParamMethodArgumentResolver
@PathVariable Map	PathVariableMapMethodArgumentResolver
@RequestHeader	RequestHeaderMethodArgumentResolver
……的集合	HandlerMethodArgumentResolverComposite

HandlerMethodArgumentResolver

```
boolean supportsParameter(MethodParameter parameter);
void resolveArgument(
        MethodParameter parameter,
        ModelAndViewContainer mavContainer,
        NativeWebRequest webRequest,
        WebDataBinderFactory binderFactory) throws Exception;
```

HandlerMethodReturnValueHandler

ReturnValue描述	支持的实现类
ModelAndView	ModelAndViewMethodReturnValueHandler
@ResponseBody	RequestResponseBodyMethodProcessor
HttpEntity	HttpEntityMethodProcessor
void or String	ViewNameMethodReturnValueHandler
View	ViewMethodReturnValueHandler

HandlerMethodReturnValueHandler

```
boolean supportsReturnType(MethodParameter returnType);
Object handleReturnValue(
        Object returnValue,
        MethodParameter returnType,
        ModelAndViewContainer mavContainer,
        NativeWebRequest webRequest) throws Exception;
```

组件及Demo演示

DispatcherServlet

HandlerMapping

HandlerAdapter

HandlerExceptionResolver

ViewResolver

View

HandlerInterceptior

.....何其多

QA

Q

&

A

谢谢！