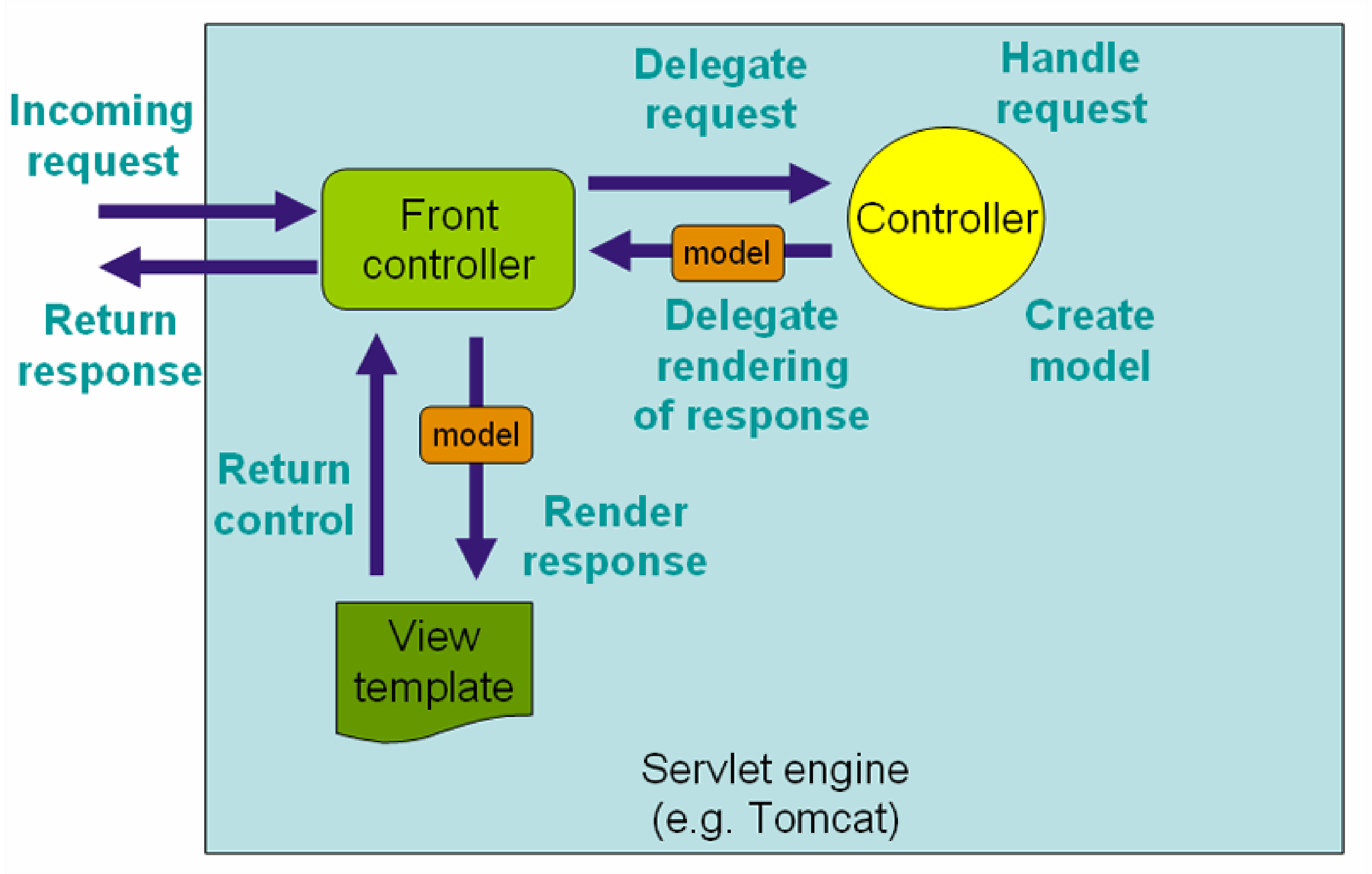
SpringMVC 4.x

凯盛软件

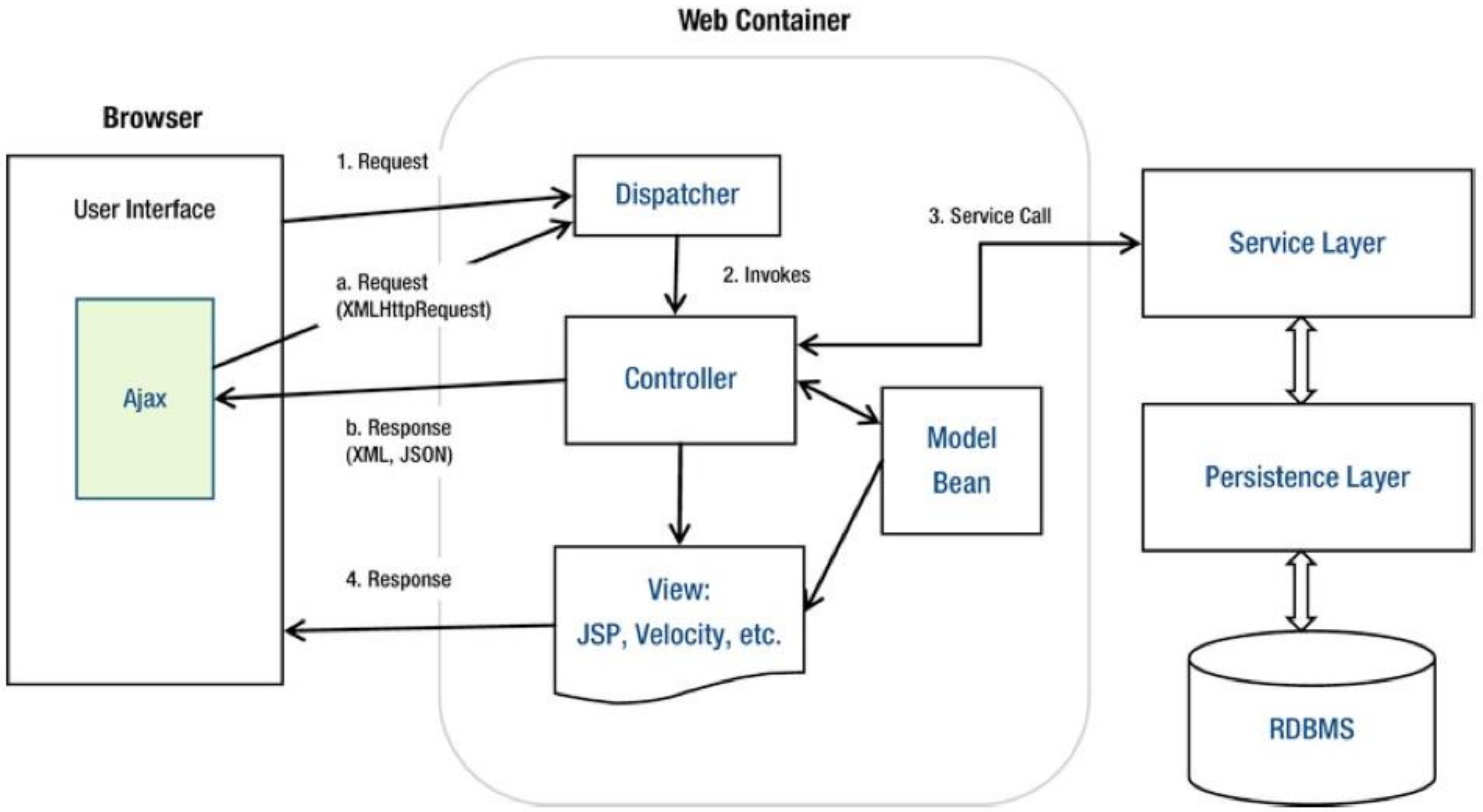
SpringMVC

凯盛软件

Spring的web框架围绕**DispatcherServlet**设计。 DispatcherServlet的作用是将请求分发到不同的处理器。



凯盛软件



Web Container
Browser
User Interface
Ajax
1. Request
a. Request (XMLHttpRequest)
b. Response (XML, JSON)
4. Response
Dispatcher
2. Invokes
Controller
Model Bean
View: JSP, Velocity, etc.
3. Service Call
Service Layer
Persistence Layer
RDBMS

配置中央(前端)控制器

凯盛软件

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

<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

/WEB-INF/**springmvc**-servlet.xml

1.添加maven依赖

2.在web.xml中添加中央控制器的配置；
在WEB-INF文件中添加SpringMVC的配置文件配置文件名为xxx-servlet.xml

3.springMVC配置的内容：配置自动扫描、视图解析器

SpringMVC的运行流程：

1.在容器启动时会创建DispatcherServlet的实例，该实例会去读取WEB-INF中的SpringMVC配置文件
2.将@Controller注解的类放入到spring容器中
3.将@RequestMapping注解的请求路径和类以及方法对应
4.根据客户端请求的路径，找到对应的类和方法并执行
5.根据方法的执行结果跳转到对应的视图，给客户端作出响应

springmvc-servlet.xml

凯盛软件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

  <context:component-scan base-package="com.kaishengit.controller"/>

  <mvc:annotation-driven/>

  <bean id="viewResolver"
   class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/WEB-INF/views/"/>
        <property name="suffix" value=".jsp"/>
  </bean>

</beans>
```

Controller

凯盛软件

```
@Controller
public class HomeController {

        @RequestMapping("/home")
        public String home() {
                System.out.println("Hello,SpringMVC");
                return "home";
        }
}
```

限定url的请求方式：

@RequestMapper注解的method属性

更多RequestMapping

凯盛软件

```
@Controller

@RequestMapping("/home")

public class HomeController {


		@RequestMapping(method=RequestMethod.GET)

		public String home() {

				System.out.println("Hello,SpringMVC");

				return "home";

		}

}
```

```
@Controller
@RequestMapping("/user")
public class UserController {

        @Inject
        private UserServcie userService;

        @RequestMapping(method=RequestMethod.GET)
        public String list(Model model) {
                model.addAttribute("userList", userService.findAll());
                return "user/list";
        }
}
```

```
@RequestMapping(value="/{id:\\d+}",method=RequestMethod.GET)
public ModelAndView get(@PathVariable int id) {
		ModelAndView mav = new ModelAndView("user/user-info");
		mav.addObject("user", userService.findById(id));
		return mav;
}

@RequestMapping("/{nickname}")
public String getByName(@PathVariable("nickname") String name) {}

@RequestMapping("/{classId:\\d+}/{nickname}")
public String getByClassIdAndName(@PathVariable int classId,@PathVariable String nickname) {}
```

```
@RequestMapping(value="/{id:\\d+}",method=RequestMethod.GET,produces="application/json;charset=UTF-8")

@ResponseBody

public User get(@PathVariable int id) {

        return userService.findById(id);

}


@RequestMapping(value="/{id:\\d+}",method=RequestMethod.GET)

@ResponseBody

public String get(@PathVariable int id) {

        return "ok";

}
```

```
@RequestMapping(value="/del/{id:\\d+}",method=RequestMethod.GET)
public String del(@PathVariable int id) {
		userService.del(id);
		return "redirect:/user";
}

@RequestMapping(value="/del/{id:\\d+}",method=RequestMethod.GET)
public String del(@PathVariable int id,RedirectAttributes redirectAttributes) {
		userService.del(id);
		redirectAttributes.addFlashAttribute("message", "删除成功");
		return "redirect:/user";
}
```

http://127.0.0.1:8080/springmvc/user/page?p=2

```
@RequestMapping("/page")
public String page(@RequestParam(value="p")int pageNum) {
		System.out.println(pageNum);
		return "";
}

@RequestMapping("/page")
public String page(@RequestParam(value="p",defaultValue="1",required=false) int pageNum) {
		System.out.println(pageNum);
		return "";
}
```

```
@RequestMapping("/method")

public String method(HttpServletResponse response,HttpServletRequest request,HttpSession session) {



}
```

接受表单值

凯盛软件

```
<form action="user/new" method="post">
        UserName:<input type="text" name="name"/></br>
        Password:<input type="password" name="password"/><br/>
        Zip:<input type="text" name="zip"/><br/>
        <input type="submit"/>
</form>
```

```
@RequestMapping(value="/new",method=RequestMethod.POST)
public String save(User user,String zip) {
        System.out.println(user.getName());
        System.out.println(user.getPassword());
        System.out.println(zip);
        return "suc";
}
```

文件上传

凯盛软件

```
<bean id="multipartResolver"
  class="org.springframework.web.multipart.commons.CommonsMultipartResolver”>
          <property name="maxUploadSize" value="100000"/>
</bean>

<form method="post" action="user/upload" enctype="multipart/form-data">
    <input type="text" name="name"/>
    <input type="file" name="file"/>
    <input type="submit"/>
</form>
```

```
@RequestMapping("/upload")
public String fileupload(String name,MultipartFile file) {
        System.out.println("name:" + name);
        System.out.println("OriginalFileName:" + file.getOriginalFilename());
        System.out.println("size" + file.getSize());
        if(!file.isEmpty()) {
                InputStream inputStream = file.getInputStream();
                //后面的该会了吧
        }
        return "";
}
```

静态资源访问

凯盛软件

```
<mvc:resources location="/static/" mapping="/static/**"/>
```

其他映射

凯盛软件

```
<mvc:view-controller path="/" view-name="index"/>
```

拦截器

凯盛软件

```
public class MyInterceptor extends HandlerInterceptorAdapter{
		private List<String> excluedUrls;
		@Override
		public boolean preHandle(HttpServletRequest request,
						HttpServletResponse response, Object handler) throws Exception {
			for(String url : excluedUrls) {
				if(request.getRequestURI().endsWith(url)) {
					return true;
				}
			}
			HttpSession session = request.getSession();
			if(session.getAttribute("curr_user") == null) {
				throw new AuthorizationException();
			}
			return true;
		}
		public void setExcluedUrls(List<String> excluedUrls) {
			this.excluedUrls = excluedUrls;
		}

}
```

拦截器：类似于过滤器

创建一个拦截器可以通过实现HandlerInterceptor接口或者继承 HandlerInterceptorAdapter这个类。

preHandler方法用于请求拦截，返回值为Boolean类型，true放行，false进行拦截，还需要在springMVC配置文件进行配置。

静态资源也会通过拦截器。

```
<mvc:interceptors>
        <mvc:interceptor>
                <mvc:mapping path="/**"/>
                <bean class="com.kaishengit.controller.MyInterceptor">
                        <property name="excluedUrls">
                                <list>
                                        <value>/home</value>
                                </list>
                        </property>
                </bean>
        </mvc:interceptor>
</mvc:interceptors>
```