

Spring framework

Motto: Musíte rozbít vejce když chcete udělat omeletu



Spring framework training materials by [Roman Pichlík](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Spring MVC

Jak jsme se tam dostali

Špagetový kód

- JSP 1.0, skripty

```
<%  
    Connection con = null;  
    String sql = "select name from users";  
    PreparedStatement ps = con.prepareStatement(sql);  
    ResultSet rs = ps.executeQuery();  
    out.println("<table>");  
    while(rs.next()) {  
        out.println("<tr>");  
        out.println("<td>");  
        out.println(rs.getString(1));  
        out.println("</td>");  
        out.println("</tr>");  
    }  
    out.println("</table>");  
%>
```

Nevýhody

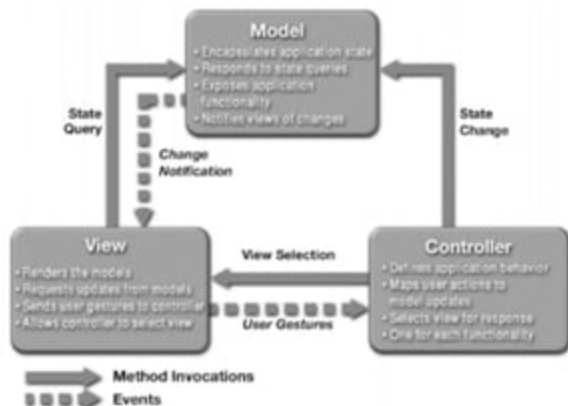
- Znovupoužitelnost
- Míchání odpovědností
 - prezentační a aplikační logika dohromady
 - nemožnost alternativní reprezentace
- Špatná testovatelnost

MVC

- Oddělení odpovědností
 - Aplikační chování
 - Data a jejich reprezentaci
- Obecný návrhový vzor
 - Desktop, klientské frameworky
 - JavaScript - SproutCore
 - Swing

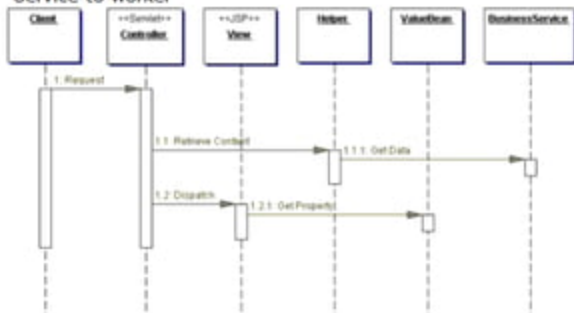
MVC

- Model
- Date
- Controller
- Aplikacní chování
- View
- Prezence Modelu

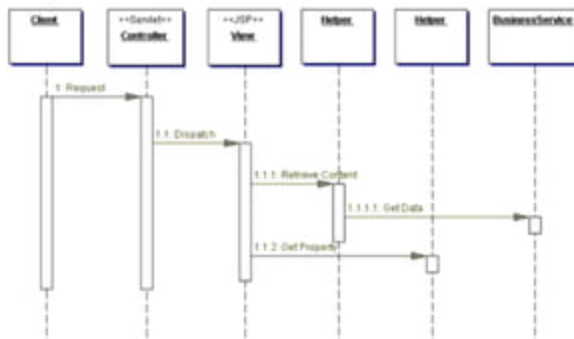


MVC v prostředí web framew.

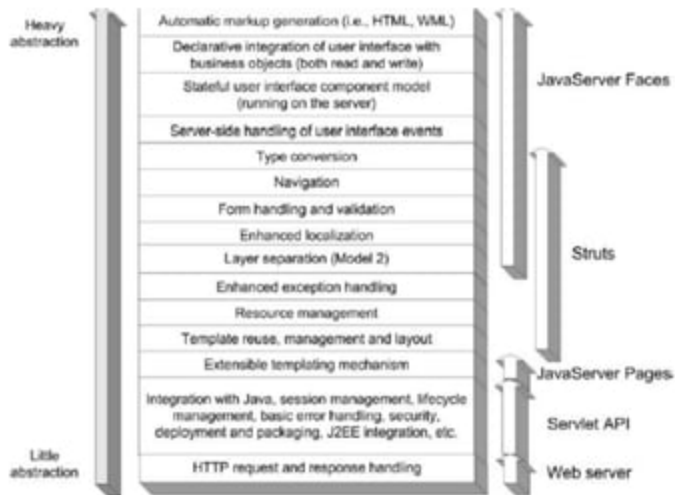
Service to worker



Dispatcher view



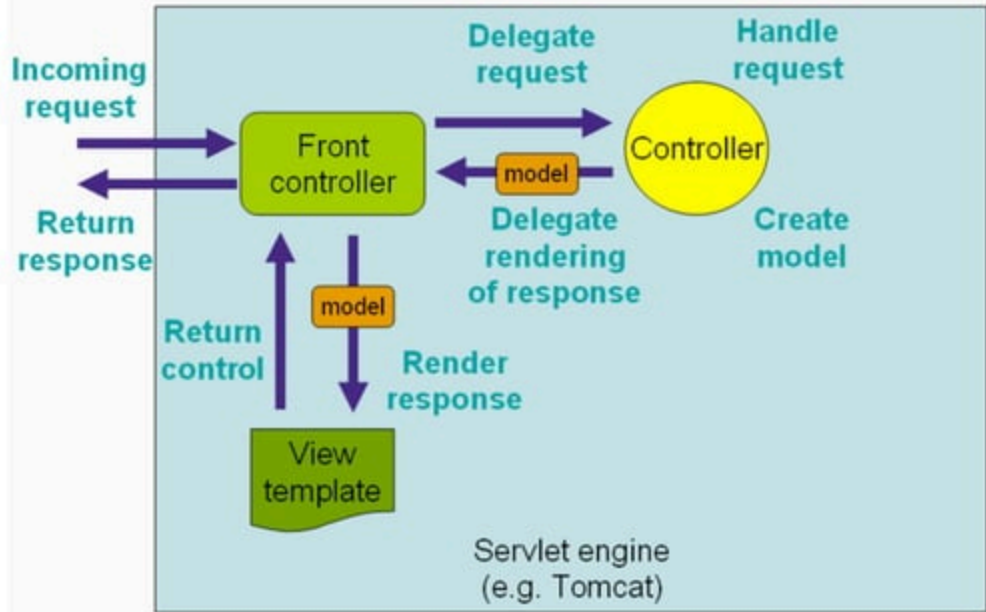
Web frameworky



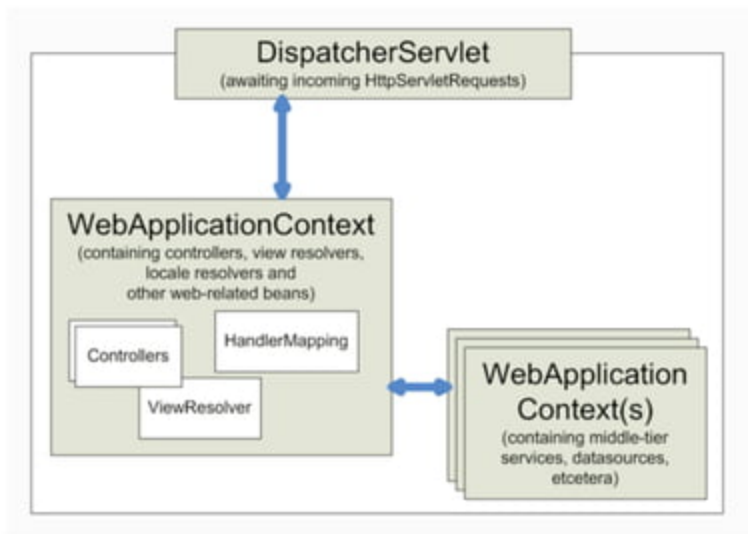
Spring MVC

- XML, Anotace
- Dependency Injection
- SPEL, Validace
- Koncepty zůstávají zachované

Odbavení požadavku



Aplikační context



Základní anotace

- @Controller
- @RequestMapping
- @PathVariable
- @RequestParam

Controller

- @Controller
- Vstupní bod
- Mapování na konkrétní URL path
- Bridge HTTP / Aplikační logika
- Měl by být jednotkově testovatelný

Controller ukázka

```
@Controller
public class UserController {

    @Autowired
    private UserStorageDao userStorageDao;

    @RequestMapping("/users.htm")
    public ModelMap usersHandler() {
        return new ModelMap("users", userStorageDao.getAll());
    }
}
```

@RequestMapping

- Používá se pro třídy i metody
- Lze vhodně kombinovat
- Může definovat path, HTTP metodu případně další podmínky, za kterých dojde k match


```

@Controller
@RequestMapping("/appointments")
public class AppointmentsController {

    private final AppointmentBook appointmentBook;

    @Autowired
    public AppointmentsController(AppointmentBook appointmentBook) {
        this.appointmentBook = appointmentBook;
    }

    @RequestMapping(method = RequestMethod.GET)
    public Map<String, Appointment> get() {
        return appointmentBook.getAppointmentsForToday();
    }

    @RequestMapping(value="/{day}", method = RequestMethod.GET)
    public Map<String, Appointment> getForDay(@PathVariable Date day, Model model) {
        return appointmentBook.getAppointmentsForDay(day);
    }

    @RequestMapping(value="/new", method = RequestMethod.GET)
    public AppointmentForm getNewForm() {
        return new AppointmentForm();
    }

    @RequestMapping(method = RequestMethod.POST)
    public String add(@Valid AppointmentForm appointment, BindingResult result) {
        if (result.hasErrors()) {
            return "appointments/new";
        }
        appointmentBook.addAppointment(appointment);
        return "redirect:/appointments";
    }
}

```

Path mapping

```
@Controller
@RequestMapping("/appointments")
public class AppointmentsController {
```

← třída definuje první část cesty

```
    private final AppointmentBook appointmentBook;
```

```
    @Autowired
```

```
    public AppointmentsController(AppointmentBook appointmentBook) {
        this.appointmentBook = appointmentBook;
    }
```

```
    @RequestMapping(method = RequestMethod.GET)
```

```
    public Map<String, Appointment> get() {
        return appointmentBook.getAppointmentsForToday();
    }
```

```
    @RequestMapping(value="/{day}", method = RequestMethod.GET)
```

```
    public Map<String, Appointment> getForDay(@PathVariable Date day, Model model) {
        return appointmentBook.getAppointmentsForDay(day);
    }
```

```
    @RequestMapping(value="/new", method = RequestMethod.GET)
```

```
    public AppointmentForm getNewForm() {
        return new AppointmentForm();
    }
```

```
    ...
```

metody definují zbytek

Path mapping

http://myapp/appointments/2011-01-01

http://myapp/appointments/new

```
@Controller
@RequestMapping("/appointments")
public class AppointmentsController {

    private final AppointmentBook appointmentBook;

    ...

    @RequestMapping(value="/{day}", method = RequestMethod.GET)
    public Map<String, Appointment> getForDay(@PathVariable Date day, Model model) {
        return appointmentBook.getAppointmentsForDay(day);
    }

    @RequestMapping(value="/new", method = RequestMethod.GET)
    public AppointmentForm getNewForm() {
        return new AppointmentForm();
    }

    ...
}
```



HTTP method mapping

mapping na konkrétní HTTP metodu

```
@Controller
@RequestMapping("/appointments")
public class AppointmentsController {

    @RequestMapping(method = RequestMethod.GET)
    public Map<String, Appointment> get() {
        return appointmentBook.getAppointmentsForToday();
    }

    @RequestMapping(value="/{day}", method = RequestMethod.GET)
    public Map<String, Appointment> getForDay(@PathVariable Date day, Model model) {
        return appointmentBook.getAppointmentsForDay(day);
    }

    @RequestMapping(method = RequestMethod.POST)
    public String add(@Valid AppointmentForm appointment, BindingResult result) {
        if (result.hasErrors()) {
            return "appointments/new";
        }
        appointmentBook.addAppointment(appointment);
        return "redirect:/appointments";
    }
}
```

Podmínečný mapping

HTTP hlavičkou

```
@Controller
@RequestMapping("/{ownerId}")
public class RelativePathUriTemplateController {

    @RequestMapping(value = "/pets", method = RequestMethod.POST, headers="content-type=text/*")
    public void addPet(Pet pet, @PathVariable String ownerId) {
        // implementation omitted
    }
}
```

Query parametrem

```
@Controller
@RequestMapping("/{ownerId}")
public class RelativePathUriTemplateController {

    @RequestMapping(value = "/pets/{petId}", params="myParam=myValue")
    public void findPet(@PathVariable String ownerId, @PathVariable String petId, Model model) {
        // implementation omitted
    }
}
```

Handler method argumenty

- `ServletRequest` or `HttpServletRequest`, `HttpSession`.
- `org.springframework.web.context.request.WebRequest`,
`org.springframework.web.context.request.NativeWebRequest`.
- `java.util.Locale`
- `java.io.InputStream` / `java.io.Reader`
- `java.io.OutputStream` / `java.io.Writer`
- `java.security.Principal`
- `@PathVariable`
- `@RequestParam`
- `@RequestHeader`
- `@RequestBody`
- `HttpEntity<?>`
- `java.util.Map` / `org.springframework.ui.Model` / `org.springframework.ui.ModelMap`
- `org.springframework.validation.Errors` / `org.springframework.validation.BindingResult`
- `org.springframework.web.bind.support.SessionStatus`

@PathVariable

```
@RequestMapping(value="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {
    Owner owner = ownerService.findOwner(ownerId);
    model.addAttribute("owner", owner);
    return "displayOwner";
}
```

@PathVariable

http://myapp/owners/1

A blue circle highlights the '1' in the URL 'http://myapp/owners/1'. Two arrows originate from this circle: one points to the '@PathVariable String ownerId' parameter in the code below, and the other points to the '{ownerId}' placeholder in the @RequestMapping value.

```
@RequestMapping(value="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {
    Owner owner = ownerService.findOwner(ownerId);
    model.addAttribute("owner", owner);
    return "displayOwner";
}
```


@PathVariable

- @PathVariable typ může být pouze základní typ (long, String)
- Pro další typy =>PropertyEditor

```
@Controller
public class MyFormController {

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
        dateFormat.setLenient(false);
        binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, false));
    }
}
```

@ModelAttribute

- Přednahrání dat do formuláře
- Propojení modelu do konverzačního schématu
- obsluha formuláře

Přednahrání modelu

- metoda se volá ještě před vlastní handler metodou
- model atributy lze namapovat do handler metody

```
public class EditPetForm {  
  
    // ...  
  
    @ModelAttribute("types")  
    public Collection<PetType> populatePetTypes() {  
        return this.clinic.getPetTypes();  
    }  
  
    @RequestMapping  
    public void handle(@ModelAttribute Collection<PetType> types)
```

Konverzační použití

```
@Controller
@RequestMapping("/editUser.htm")
@SessionAttributes("user")
public class UserEditFormController {


    @Autowired
    private UserStorageDao userStorageDao;

    @RequestMapping(method = RequestMethod.GET)
    public String setupForm(ModelMap model) {
        model.addAttribute("user", new User());
        return "editUser"; //viewname
    }

    @RequestMapping(method = RequestMethod.POST)
    public String processSubmit( @ModelAttribute("user") User user, BindingResult result, SessionStatus status) {
        userStorageDao.save(user);
        status.setComplete();
        return "redirect:users.htm";
    }
}
```

Návratové typy

návratový typ



```
@RequestMapping(method = RequestMethod.POST)
public XXX processSubmit(@ModelAttribute("pet") Pet pet,
    BindingResult result, Model model) {
    ...
}
```

Možné návratové typy

- `org.springframework.web.servlet.ModelAndView`
- `org.springframework.ui.Model`
- `java.util.Map`
- `org.springframework.ui.View`
- `java.lang.String`
- `void`
- `@ResponseBody`
- A `HttpEntity<?>` or `ResponseEntity<?>`
- Jiný typ

Automatická validace

- integrace JSR-303
- BindingResult objekt s výsledkem validace

```
@RequestMapping(method = RequestMethod.POST)
public String processSubmit( @ModelAttribute("user") @Valid User user, BindingResult result, SessionStatus status) {
    if (result.hasErrors()) {
        return "editUser";
    }

    userStorageDao.save(user);
    status.setComplete();
    return "redirect:users.htm";
}
```

Exception handling

- Vytvoření beany implementující
 - `HandlerExceptionResolver`
 - Vrací `ModelAndView`
- Ošetření všech výjímek
 - Zobrazení error stránky
 - Zalogování kontextu
 - REST (správný HTTP status)

Interceptors

- Možnost reagovat před a po zpracování HTTP požadavku
 - pre, post, after
- Vytvoření beany implementující
 - `HandlerInterceptor`
 - `HandlerInterceptorAdapter`

Spring MVC zavedení

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

Spring MVC zavedení s view r.

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

    <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/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

Praktické cvičení

- Vytvořte beany pro
 - ReservationServiceImp
 - InMemoryBookStoreDao
 - inicializujte s několika knihami viz setter pro bookHolder
- Vytvořte controller
 - vrátí seznam rezervací
 - použijte @RequestParam pro její vyfiltrování
 - vrátí konkrétní rezervaci
 - umožní vytvořit novou rezervaci (inspiraci hledejte v již naimplementovaných uživatelích)
 - použijte @PathVariable
 - validace vstupu
 - přidejte interceptor, který vypíše čas obsluhy
 - přidejte ErrorHandler, který bude logovat výjimky