Spring instaliavimas, MVC projektas

Paruoškime IDE darbui su spring

Iš eclipse marketplace instaliuokime STS - Spring Tools (Spring IDE and Spring Tool Suite)

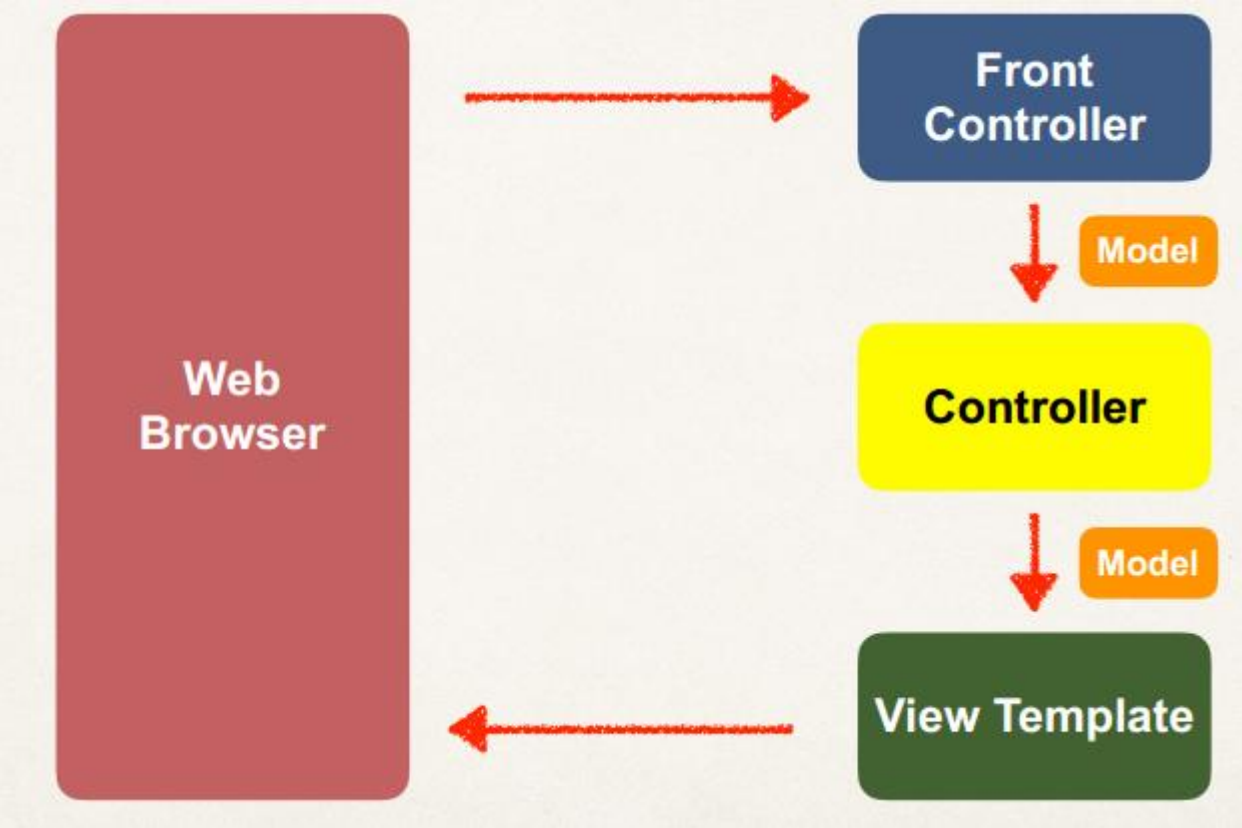
Tai galime padaryti nuėję:

http://marketplace.eclipse.org/content/spring-tools-aka-spring-ide-and-spring-tool-suite

Užduotis

Susikurkime naują Spring MVC projektą

Spring MVC architektūra



Spring MVC architektūra

Front controller – DispatcherServlet

- Dalis Spring framework
- Priima duomenis iš vartotojo, nukreipia į reikiamą kontrolerį

Controller classes

- Valdymo struktūra, Duomenų patalpinimas ir paėmimas, Patalpinimas duomenų į modelius kurie perduodami View templates
- Dažniausiai realizuojama servletais

Model Objects

- Duomenų patalpinimas ir perdavimas iš
- Duomenų bazė, Spring beans, Hibernate

View templates

- Visas tinklapio atvaizdavimas
- Dažniausiai JSP+JSTL, tačiau gali būti ir Thymeleaf, Groovy, Velocity ...

Spring servletų sukūrimas ir priskyrimas

Faile WEB-INF/web.xml yra patalpinta visas URL maping failas, kuriame yra nurodyti servletai, jų konfigūracija ir mappinimas:

```
<servlet>
<servlet-name>appServlet</servlet-name>     //Servleto pavadinimas
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
.....
</servlet>


<servlet-mapping>
<servlet-name>appServlet</servlet-name>     //Servleto pavadinimas
<url-pattern>/</url-pattern>                //Paternas URL kurie bus persiųsti į servletą
</servlet-mapping>
```

Spring servletų sukūrimas ir priskyrimas

Faile servlet-context.xml rasime visą informaciją apie pačio servleto konfiguravimą.

```
<!– Nuoroda jog naudosime anotacijų aprašą (su @ pradžioje) -->
<annotation-driven />

<!—Resursų mappingas -->
<resources mapping="/resources/**" location="/resources/" />

<!– Templait‘ų atvaizdavimui skirta informacia-->
<beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="lt.poligonas.SpringTest" />
```

Užduotis

Pasinaudodami Spring MVC frameworku sukurkime formą, kurioje įvestumėme du skaičius, kitame dokumente išvestų tų skaičių sumą.

Tam sukursime atskirą kontrollerį (@Controller)

Kontroleris turės turėti du metodus:

showForm() – atvaizduos įvedimo formą

processForm() – apdoros įvestą infromaciją ir išves atsakymą

Padarykime mappinimą tarp URL ir metodų:

/showFrom -> showForm()

/processForm -> processFrom()

URL mapping

Norėdami nurodyti kuris kontrolerio metodas apdoros kurį kelią, mes galime nurodyti anotaciją virš metodo: RequestMapping

```
@RequestMapping(value = "/")
```

Jei norime taip pat galime nurodyti ir metodą kuriuo atsiųstus duomenis apdoros mūsų metodas:

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

Request parametras

Jei mums reiktų parametrų atsiųstų per GET ar post metodą, mes galime į parametrą paprašyti jog paduotų HttpServletRequest objektą:

```
public String home(HttpServletRequest request) {

}
```

Tuomet duomenys būtų request.getParameter("Lauko pavadinimas");

Model parametras

Jei mums reikės perduoti duomenis į View, mes galime pasinaudoti modeliu ir tuos duomenis perduoti per jį. Tuomet modelio paprašytumėme kartu su kitais parametrais:

```
public String home(HttpServletRequest request, Model model) {

}
```

Tuomet duomenų pridėjimui galėtumėme naudoti

```
model.addAttribute("pavadinimas", "reikšmė");
```

Duomenys JSP faile būtų pasiekiami ${pavadinimas}

Spring MVC - statinio turinio pridėjimas

Norėdami pridėti statinį turinį (CSS/JS ar paveikslus) prie JSP failų, konfigūracijoje turėtumėme nurodyti:

```
<resources mapping="/resources/**" location="/resources/" />
```

Toumet visus resursus galėtumėme talpinti į katalogą webapp/resources

Failai JSP failuose bus pasiekiami per komandas:

```
<img src="${pageContext.request.contextPath}/resources/logo.png">
```

`${pageContext.request.contextPath}` – gražins URL iki projekto

Užduotis

Perkelkime CSS iš mūsų skaičiavimo formos į atskirą failą: style.css ir padarykime kad jis būtų užkrautas iš ten.

Parametrų paėmimas

Spring karkasas leidžia parametrus paiminėti ne per serveltus, bet naudojant anotacijas metodo parametrų apraše. Tam naudojamas užrašas:

```
@RequestParam("http_kintamasis") String kintamasis
```

Pavyzdžiui:

```
@RequestMapping(value = "/labas", method = RequestMethod.GET)
public String labas(
  @RequestParam("vardas") String vardas,
  Model model) {

  model.addAttribute("vardas", vardas);
  return "home";
}
```

Užduotis

Pakoreguokime skaičiuotuvo formą taip, kad visi duomenys būtų paimami iš vartotojo per RequestParameter

Kontrolerio RequestMapping

Mes galime nurodyti RequestMapping ne būtinai atskiriems metodams, tačiau ir pačiam kontroleriui, tuomet bus gaunamas hierarchinis kelias: kontrolerio_map/metodo_map

Pavyzdžiui:

```
@Controller
@RequestMapping ("/home")
public class HomeController {
  @RequestMapping("/show")
  public String home() {
    ...
  }
  @RequestMapping("/delete")
  public String delete() {
    ...
  }
}
```

Tuomet turėsime kelius: /home/show ir /home/delete

Užduotis

Perdarykime skaičiuotuvo metodus taip, kad jie būtų sudaryti:

- /calculator/showFrom
- /calculator/processForm

Form Spring elementų naudojimas

Sukurkime formą panaudojant Spring Form elementus, tam mums reiktų į JSP prisidėti biblioteką:

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
```

Tuomet forma galėtų atrodyti taip:

```
<form:form action="processForm" modelAttribute="duomenys">
    <form:input path="y" />
    <br>
    <form:input path="x" />

    <input type="submit" value="Submit" />
</form:form>
```

RequestMapping duomenų formai

Metode kuris atvaizduos formą galėtumėme nurodyti jog duomenys bus kraunami į Beans klasę duomenys:

```
@RequestMapping("/skaiciuoti")
public String showForm(Model model) {
model.addAttribute("duomenys", new Duomenys());
return "forma";
}
```

Spring vykdymas

Kai duomenys bus užkrauti, jis įvykdys getterius nurodyto modelio,

Kai duomenys bus įrašomi, bus vykdomi setteriai

Duomenų paėmimas ir atvaizdavimas

Formos išsaugojimui ir duomenų paėmimui galėtume naudoti tokį kodą:

```
@RequestMapping("/suskaiciuoti")
public String processForm(
  @ModelAttribute("duomenys") Duomenys duomenys)
  {
     System.out.println("duomenys: " + duomenys.getFirst());
     return "confirmation";

  }
```

Užduotis

Visą prieš tai buvusią užduotį perdarykime su form elementais ir Beans klase.