MVC (Model-View-Controller)

Veerle Ongenae

Servlets

- HTTP
- Servlets
- JSP
- MVC

MVC in GUI's

- Voorbeelden in Java Swing
  - Tabellen
  - Tekstcomponenten
    - M: document
    - V: tonen
    - C: editeermogelijkheden
  - …
- Klok
  - Mathematisch model
  - Tonen
  - Aanpassen

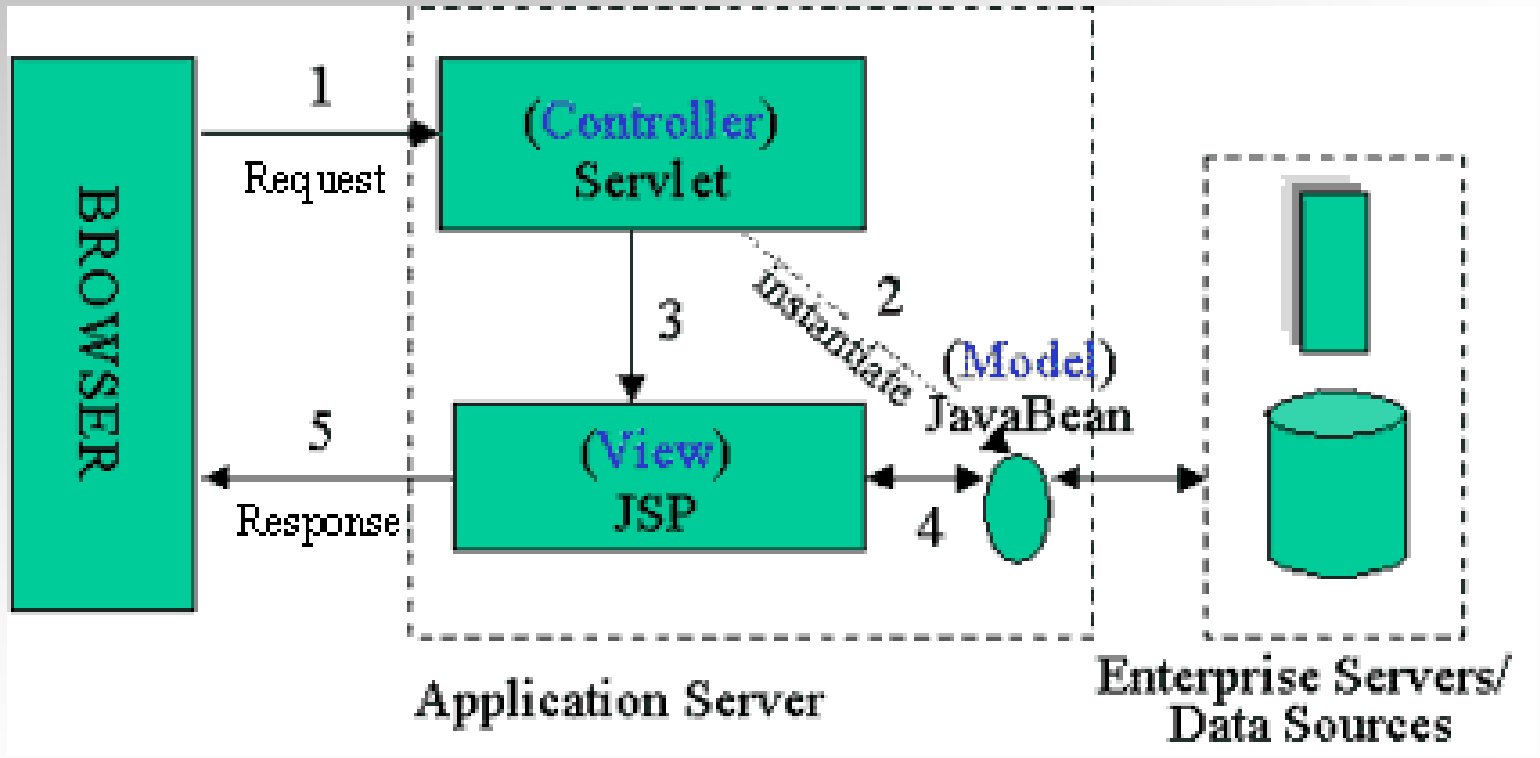
MVC in webapplicaties

- Wordt vaak Model 2 genoemd
- Principe
  - Scheiding van belangen
    - Opsplitsen programma
    - Gescheiden functionaliteit
    - Modules zoveel mogelijk “op zich”

MVC in webapplicaties

- Drie modules
  - Applicatiemodel: gegevensvoorstelling en applicatielogica (Model)
  - Presentatie gegevens en gebruikersinput (View)
  - Gedrag applicatie (flow) – doorsturen aanvragen (Controller)

MVC in webapplicaties



Controller

- Structuur:
  - Aanvragen interpreteren
  - Doorspelen naar applicatielogica
  - View selecteren

Controller

- Realisatie
  - ControllerServlet
  - Controleklassen
    - Bv. actieklassen

Model

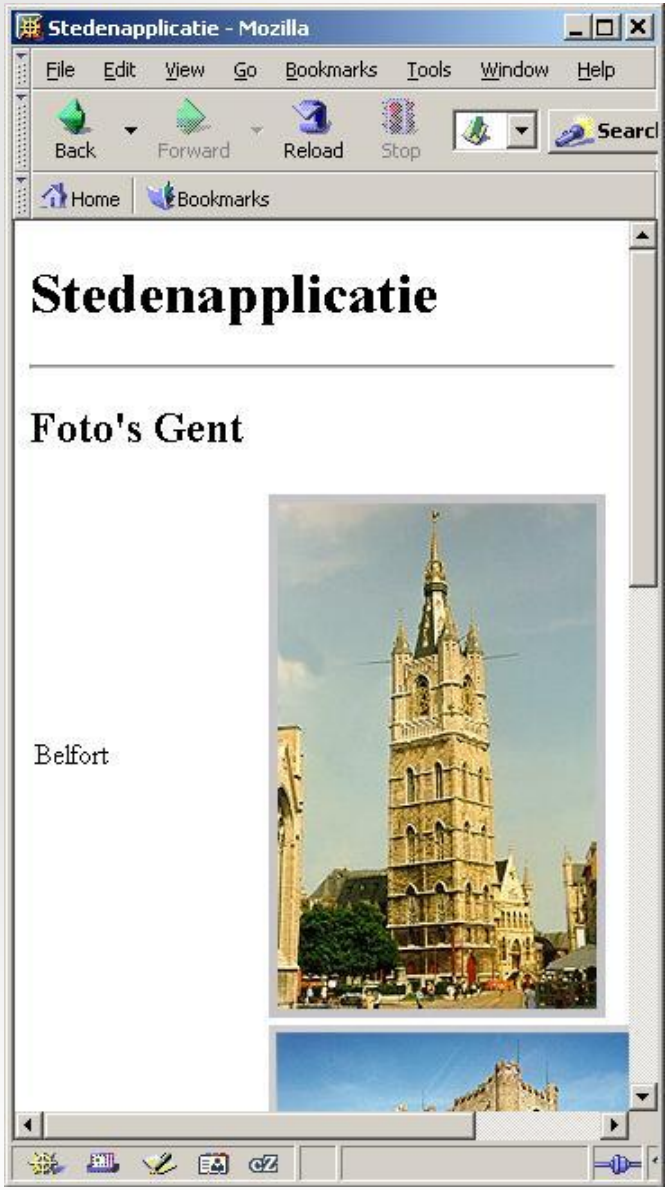
- Status van de applicatie
- Toegang tot data
- Structuur data
- Bewerkingen op data
- Functionaliteit applicatie
- Realisatie: Java-objecten en Java Beans

View

- Inhoud van Java Beans (of andere Java objecten) tonen
- JSP’s, HTML, …

Voorbeeld


Stedenapplicatie - Mozilla
Stedenapplicatie
Steden
Brussel
Gent


Stedenapplicatie - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
Home Bookmarks
Stedenapplicatie
Foto's Gent
Belfort

Voorbeeld

- Model
  - Klassen die steden en foto's voorstellen
  - Foto's opvragen van steden
  - Lijst van steden opvragen
- Controller
  - HTTP-aanvraag verwerken
  - Attributen beschikbaar stellen
- View
  - Lijstje van steden tonen
  - Foto's van één stad tonen

Model

<<interface>> OverzichtSteden	*<<interface>> Stad*	Foto
•*Stad[] getSteden()* •*Stad getStad(String)*	•*Foto[] getFotos()* •*String getNaam()*	•String getOmschrijving() •String getUrl()

- Pakket
  - be.hogent.iii.steden.bo
- Implementatie interfaces
  - be.hogent.iii.steden.bo.memImpl
    - FotoFabriek
    - StadImpl (implementatie Stad)
    - StedenFabriek (implementatie OverzichtSteden)
    - fotos.properties

FotoFabriek

- Singleton Pattern
  - Maar één object van de klasse
  - Afgeschermde constructor
  - Gesynchroniseerde methode “getInstance()”
    - Resultaat: referentie naar een “private static” veld
- Gebruik ResourceBundle
  - Naam/waarde-paren lezen uit een .properties-bestand
  - Kan ook als .properties-bestand deel uit maakt van een .jar-bestand

fotos.properties

- Properties-bestand
  - Tekstbestand
  - Per lijn een sleutel-waarde-paar

  sleutel=waarde
  - Commentaar na ‘#’

Inladen properties-bestand

```
ResourceBundle fotoData = ResourceBundle.getBundle("
    be.hogent.iii.steden.bo.memImpl.fotos");
```

- Methode getBundle
  - Zoekt eerst klasse met de naam be.hogent.iii.steden.bo.memImpl.fotos
  - Indien niet gevonden
    - Zoekt naar bestand

be/hogent/iii/steden/bo/memImpl/fotos.properties

    - Maakt hiermee een object van het type PropertyResourceBundle

Klasse ResourceBundle

- Laden ResourceBundle-objecten
  - Methode getBundle
- Ophalen waarden horende bij een sleutel
  - Methode

    public final String **getString**(String key)
  - Voorbeeld

```
String aantal = fotoData.getString("aantalGent");
```

Voorbeeld

- Model
  - Klassen die steden en foto's voorstellen
  - Foto's opvragen van steden
  - Lijst van steden opvragen
- Controller
  - HTTP-aanvraag verwerken
  - Attributen beschikbaar stellen
- View
  - Lijstje van steden tonen
  - Foto's van één stad tonen

Controller

- Pakket
  - be.hogent.iii.steden.web
- AanmaakOverzichtSteden
  - ServletContextListener
  - Initialisatie ServletContext
    - Object type OverzichtSteden op ServletContext
      - Attribuut scope application
- ControleServlet
  - Servlet
  - Afhandelen HTTP-aanvragen
  - Hulpklassen: verschillende acties

ControleServlet

- Bij initialisatie
  - Aanmaak lijst van mogelijke acties
    - Bewaard op ServletContext
- Afhandelen HTTP-aanvraag
  - Op basis van pad actie bepalen
  - Actie uitvoeren
  - Aanvraag doorsturen naar JSP (template.jsp)

Acties

- Implementeren de interface ServletActie
  - Methode execute(HttpServletRequest, HttpServletResponse)
- Elke actie plaatst de naam van JSP die ingesloten moet worden in template.jsp op de “request”
  - Attribuut scope request
- Twee acties
  - OverzichtStedenActie (~ overzicht.do)
  - ToonStadActie (~ toonstad.do)

Acties

- OverzichtStedenActie
  - Plaatst het attribuut (“view”,”steden.jsp”) op de request scope
- ToonStadActie
  - Leest parameter stadsnaam
  - Vraagt referentie naar OverzichtSteden
  - Vraagt stad aan OverzichtSteden
  - Plaatst Stad-object op de request scope
  - Plaatst het attribuut (“view”,”stad.jsp”) op de request scope

Voorbeeld

- Model
  - Klassen die steden en foto's voorstellen
  - Foto's opvragen van steden
  - Lijst van steden opvragen
- Controller
  - HTTP-aanvraag verwerken
  - Attributen beschikbaar stellen
- View
  - Lijstje van steden tonen
  - Foto's van één stad tonen

View

- template.jsp
  - Elke pagina dezelfde structuur
    - Hoofding → hoofding.jspf
    - Inhoud → stad.jsp of steden.jsp
- hoofding.jspf
  - JSP-fragment → expliciet aangeven dat dit een stukje van een JSP is
- stad.jsp
  - Toont foto's Stad-object
- steden.jsp
  - Toont lijst met steden

web.xml

- Naam klasse implementatie OverzichtSteden
- Registeren luisteraar AanmaakOverzichtSteden
- ControleServlet
- Alle *.do URL's doorsturen naar de ControleServlet
- Timeout sessie
- Startpagina

Drie types url-pattern

- De container zoekt naar (in volgorde!)
  - Exacte naam
  - Directory
    - Begint met /
    - Eindigt met /*
    - Vb. /sub/*
  - Extensie
    - Begint met *.
    - Vb. *.do

Informatie

- Server-side Java: Understanding JavaServer Pages Model 2 architecture, Exploring the MVC design pattern (http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html)
- Java BluePrints: Model-View-Controller (http://java.sun.com/blueprints/patterns/MVC-detailed.html)