Web 3

Lesson 12: Front controller continued

Op het examen…

- ✓ Leg het Front Controller patroon uit.
- ✓ Verschillende stukjes code kunnen uitleggen van de verschillende opties die er hier in de slides worden getoond.
- ✓ Voor- en nadelen van de verschillende opties kunnen uitleggen.
- ✓ Option 4 moet je NIET kunnen implementeren!
- ✓ …

FRONT CONTROLLER

‘Single Point of Entry’:

- handles requests
- delegates…
    - business processing
    - choice proper view
    - …

Single point of Entry

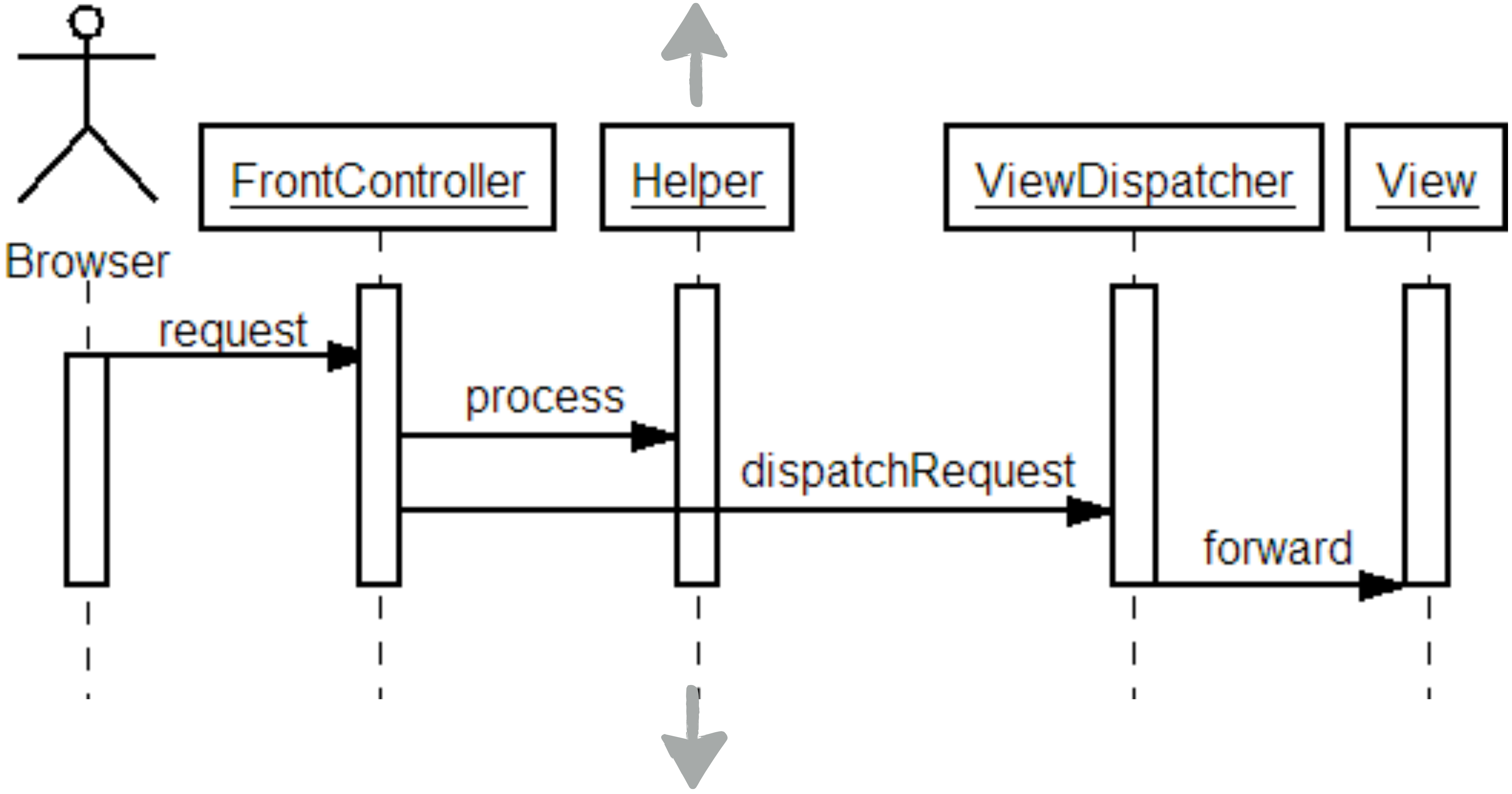
```
public class Controller extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse
         response)
           throws ServletException, IOException {
      String destination = "index.jsp";
      String action = request.getParameter("action");

      if (action == null) {
         destination = "index.jsp";
      }
      else if (action.equals("overview")) {
         destination = getOverview(request, response);
      }
      else if (action.equals("signUp")) {
         destination = "signUp.jsp";
      }
      else if (action.equals("confirmSignup")) {
         destination=confirmSignup(request,response);
      }
      else if (…) {
        …
      }
      request.setAttribute("action", action);
      RequestDispatcher view = request.getRequestDispatcher(destination);
      view.forward(request, response);
}
```

Call model to perform business logic

use dispatcher to navigate to proper view

FRONT CONTROLLER

Is now a helper **method**



Should be a helper **class**

Single point of Entry

```
public class Controller extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse
            response)
               throws ServletException, IOException {

               String action = request.getParameter("action");
               RequestHandler handler = null;

               if (action.equals("overview")) {
                   handler = new PersonOverviewHandler(service);
               } else if (action.equals("signUp")) {
                   handler = new SignUpHandler();
               } else if (action.equals("confirmSignup")) {
                    handler = new ConfirmSignupHandler(service);
                } else if (action.equals("…")) {
                   handler = new … Handler();
               }
               String destination = handler.handleRequest(request, response);
               RequestDispatcher view = request.getRequestDispatcher(destination);
               view.forward(request, response);
           }


    }
```

Use helpers to delegate to model

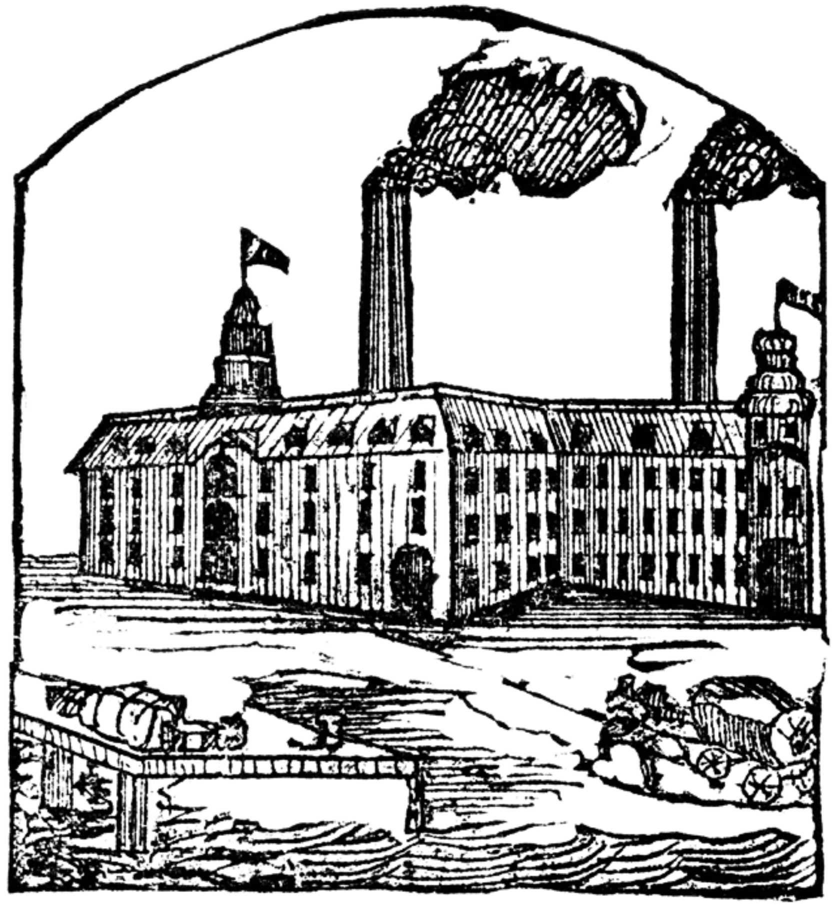
use dispatcher to navigate to proper view

```
public class Controller extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse
         response)
           throws ServletException, IOException {

           String action = request.getParameter("action");
           RequestHandler handler = null;

           if (action.equals("overview")) {
               handler = new PersonOverviewHandler(service);
           } else if (action.equals("signUp")) {
               handler = new SignUpHandler();
           } else if (action.equals("confirmSignup")) {
                handler = new ConfirmSignupHandler(service);
            } else if (action.equals("…")) {
               handler = new … Handler();
           }
           String destination = handler.handleRequest(request, response);
           RequestDispatcher view = request.getRequestDispatcher(destination);
           view.forward(request, response);
       }


   }
```

~~OCP~~ ?

→ How to avoid?

Simple Factory !



Controller

```
protected void processRequest(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
    try {
        String action = request.getParameter("action");
        RequestHandler handler =
                                handlerFactory.getHandler(action, service);
        String destination = handler.handleRequest(request, response);
        RequestDispatcher view = request.getRequestDispatcher(destination);
        view.forward(request, response);
    } catch (Exception e) {
       throw new ServletException(e.getMessage(), e);
    }
}
```

Option 1

HandlerFactory

```
public class HandlerFactory {

    public RequestHandler getHandler(String action,
                                        ShopService service) {
        RequestHandler handler = null;

        if (action.equals("overview")) {
            handler = new PersonOverviewHandler(service);
        } else if (action.equals("signUp")) {
            handler = new SignUpHandler();
        } else if (action.equals("confirmSignup")) {
            handler = new ConfirmSignupHandler(service);
        } else if (action.equals("…")) {
            handler = new … Handler();
        }
    }
}
```

Just move the if-statement to a separate class

HandlerFactory

```java
public class HandlerFactory {
    private Map<String, RequestHandler> handlers = new HashMap<>();

    public HandlerFactory(ShopService service) {
        handlers.put("overview", new PersonOverviewHandler(service));
        handlers.put("signUp", new SignUpHandler());
        handlers.put("confirmSignup", new ConfirmSignupHandler(service));
        …;
   }

    public RequestHandler getHandler(String key) {
        return handlers.get(key);
    }
}
```

Same idea, but a bit more elegant …

OPTION 1

- Factory class hardcoded:
  - code not 'Closed for modification'
  + isolated

Option 2

HandlerFactory

```
public class HandlerFactory {

   private RequestHandler getHandler(String handlerName, ShopService model)
         throws ServiceException {
      RequestHandler handler = null;
      try {
         Class handlerClass = Class.forName("controller."+ handlerName);
         Object handlerObject = handlerClass.newInstance();
         handler = (RequestHandler) handlerObject;
         handler.setModel(model);
      } catch (ClassNotFoundException e) {
         throw new ServiceException(e);
      }

      return handler;
   }
}
```

Use reflection

OPTION 2

- Factory using reflection:
  - + code **is** ‘Closed for modification’
  - - name action should match name handler

$\searrow$

tight coupling view - controller !

Option 3

HANDLER.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="userOverview">controller.handler.UserOverviewHandler</entry>
    <entry key="confirmSignup">controller.handler.ConfirmSignupHandler</entry>
    <entry key="signUp">controller.handler.SignUpHandler</entry>
     …
</properties>
```

Use a property file

CONTROLLER

```
private HandlerFactory handlerFactory;

public void init() throws ServletException {

     super.init();
     …
     try {
         ShopService service = new ShopService(…);

         InputStream input = context.getResourceAsStream("/WEB-INF/handler.xml");
         Properties properties = new Properties();
         properties.loadFromXML(input);

         handlerFactory = new HandlerFactory(properties, service);
      } catch (Exception ex) {
           …
      }
}
```

HandlerFactory

```java
public class HandlerFactory {
    private Map<String, RequestHandler> handlers = new HashMap<>();

    public HandlerFactory(Properties handlerNames, ShopService model) {
         for(Object key : handlerNames.keySet()) {
               RequestHandler handler = null;
               String handlerName = handlerNames.getProperty((String) key);
               try {
                  Class<?> handlerClass = Class.forName(handlerName);
                  Object handlerObject = handlerClass.newInstance();
                  handler = (RequestHandler) handlerObject;
               } catch (ClassNotFoundException e) {
                  …
               }
               handler.setModel(model);
               handlers.put((String)key, handler);
         }
    }

    public RequestHandler getRequestHandler(String key) {
        return handlers.get(key);
    }
}
```

OPTION 3

- Factory with configuration file:
  - + code **is** ‘Closed for modification’
  - + name action should **not** match name handler
  - - a lot of XML

Option 4

Annotations

```
@RequestMapping(action="login")
public class LoginHandler extends RequestHandler {
   …
}
```

Challenge: write your own annotations

ANNOTATIONS

- No Factory , but…
- Only implement this option if you need a challenge or if you have some time left
  - Implementation is no exam material
- Create your own annotations
  - more information on Toledo …

OPTION 4

- Annotations:
  - + code i**s** ‘Closed for modification’
  - + name action should **not** match name servlet
  - + no configuration file

Warning
Post Redirect Get!

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    try {
        String action = request.getParameter("action");
        RequestHandler handler = handlerFactory.getHandler(action, service);
        String destination = handler.handleRequest(request, response);

        RequestDispatcher view = request.getRequestDispatcher(destination);
        view.forward(request, response);
    } catch (Exception e) {
       throw new ServletException(e.getMessage(), e);
    }
}
```

Move to helper classes !

CONTROLLER

```
protected void processRequest(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
    try {
        String action = request.getParameter("action");
        RequestHandler handler = handlerFactory.getHandler(action, service);
        handler.handleRequest(request, response);
    } catch (Exception e) {
       throw new ServletException(e.getMessage(), e);
    }
}
```

Handler

```java
public class LoginHandler extends RequestHandler {
   private ShopService service;

   public LoginHandler(ShopService service, boolean redirect) {
      this.service = service;
   }

   @Override
   public void handle(HttpServletRequest request, HttpServletResponse response) {
      String userId = request.getParameter("userId");
      String password = request.getParameter("password");

      Person person = service.getUserIfAuthenticated(userId, password);

      if (person != null) {
         HttpSession session = request.getSession();
         session.setAttribute("user", person);

         response.sendRedirect("Controller?action=home");
      } else {
         RequestDispatcher view = request.getRequestDispatcher("login.jsp");
         view.forward(request, response);
      }
   }
}
```

?