Wicket – Structures

TUTORIAL 4

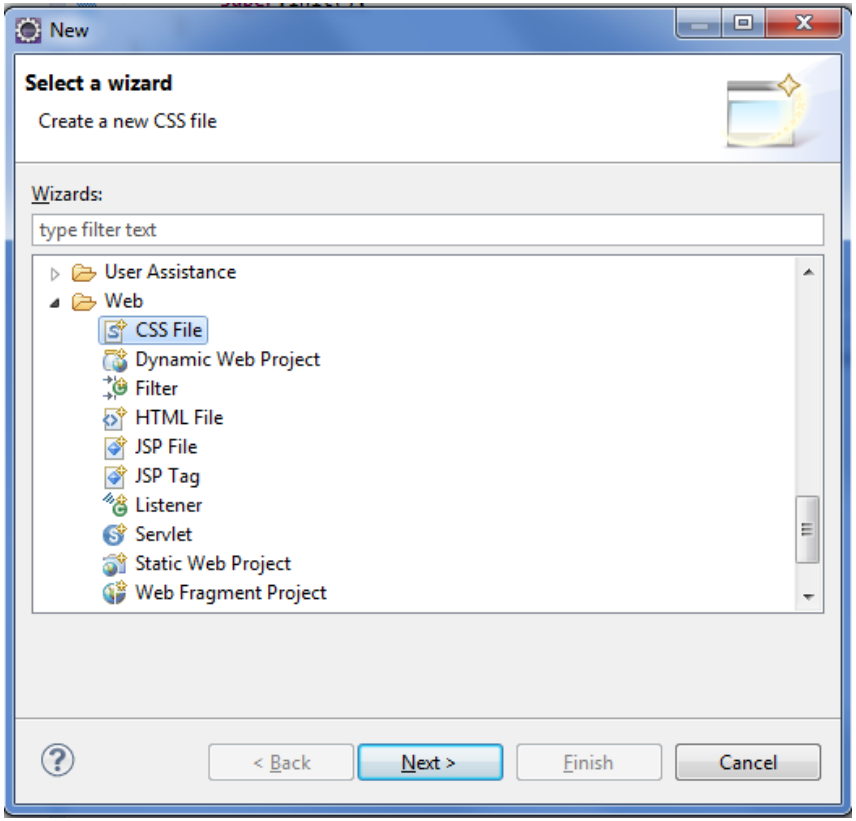
DBMS 2012-2013 Fall

TAs: Nagehan Ilhan

Mahiye Uluyağmur

Adding Stylesheet

- Select File->New->Other->Web.



Adding Stylesheet

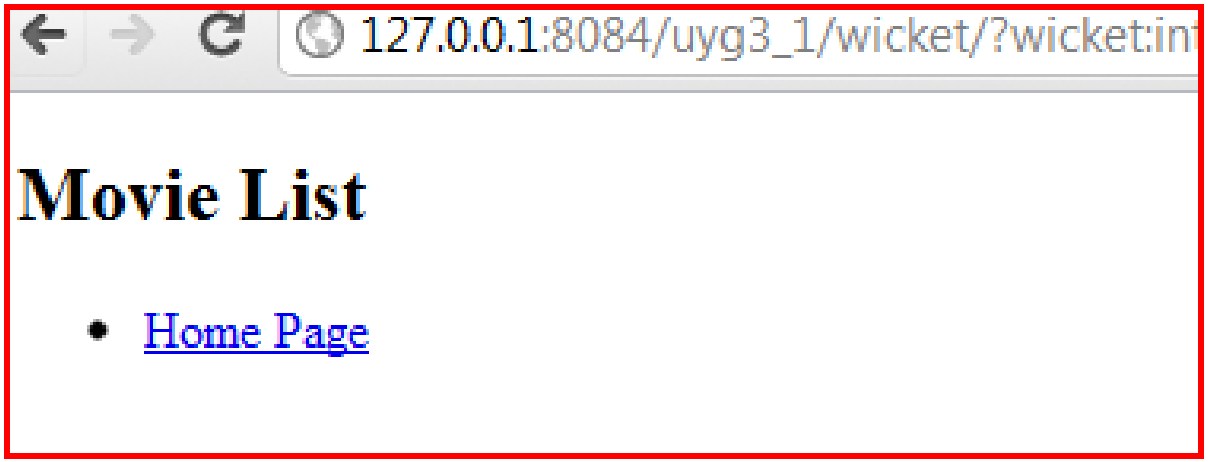
- style.css file will be created.
- Add the html codes which is below in HomePage.html file.

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
  <title>MovieDB</title>
  <link wicket:id="stylesheet" rel="stylesheet" type="text/css" href="#"/>
 </head
```

Adding Stylesheet

- Stylesheet will be called from HomePage.java file.
- **this.add(new StyleSheetReference("stylesheet", HomePage.class,"style.css"));**
- We just added stylesheet to the HomePage.
- Our aim is to use these stylesheet in our all pages, not only in HomePage.

Adding Stylesheet



Shared Components

- Adding shared components to multiple pages in an application is a tedious and error-prone approach.
- Specify these shared components at one point and let pages get them from that single source.
- Add these shared components to a base page and extend all application pages from this base page.

Shared Components

```
public class BasePage extends WebPage {
public BasePage() {  this(null);    }
public BasePage(IModel model) {
        super(model);
        this.add(new StyleSheetReference("stylesheet",
    BasePage.class,"style.css"));
        this.add(new HeaderPanel("mainNavigation"));
    }
}
```

Shared Components

- Note that shared components in HeaderPanel.html
  - <wicket:panel>
  -
  - **
Home
**
  - **
List movies
**
  -
  - </wicket:panel>

Shared Components

Changes In HeaderPanel.java.

```
public class HeaderPanel extends Panel {
 public HeaderPanel(String id) {
  super(id);
 Link homeLink = new Link("home_page") {
        public void onClick() {
         this.setResponsePage(new HomePage());   }
   };
 this.add(homeLink);
 Link movieListLink = new Link("list_movies") {
        public void onClick() {
         this.setResponsePage(new MovieListPage());     }
};
 this.add(movieListLink);}}
```

Shared Components

- Usage of the Panel in other pages.
- HomePage.html
  - links to common pages
- Add also to the BasePage.java.
  - this.add(new HeaderPanel("mainNavigation"));

Shared Components

- The template which is created in BasePage.java is inherited from HomePage.java
  - public class **HomePage extends BasePage** {
  - public HomePage() {
  - Date now = new Date();
  - this.add(new Label("datetime", now.toString())); }}

Shared Components

Shared Components

- The template is used only in HomePage.

Using Template in All Pages

- MovieList page must be created from the same BasePage for using same template.
- Changes in MovieListPage.html.
- ** links to common pages **

Using Template in All Pages

- MovieListPage.java must be inherited from BasePage.java as it is seen below.

```
public final class MovieListPage extends BasePage {
public MovieListPage() {
 Link movieListLink = new Link("home_page") {
        @Override
  public void onClick() {
  this.setResponsePage(new HomePage());
        }      };
  this.add(movieListLink);}}
```

Using Template in All Pages

Using Template in All Pages

Data Model

- We will implement the «movie class» and fill the «movie list page» with some in-place generated test data.

Data Model

Constructors

- First creates a movie with title and year as NULL
- Second creates it with a title.

Getter-Setter Methods

```
public class Movie {
        private String title = null;
        private Integer year = null;
        public Movie() {
                }
        public Movie(String aTitle) {
                this.setTitle(aTitle);
                }
        public void setTitle(String aTitle) {
                this.title = aTitle;
                }
        public String getTitle() {
                return this.title;
                }
        public void setYear(Integer aYear) {
                this.year = aYear;

}
        public Integer getYear() {
                return this.year;
                }
}
```

Data Model

To display a list of movies,
We have to add these tags to the template.

```
<body>
        <div wicket:id="mainNavigation">links to common pages
        </div>
        <h2>Movie List
        </h2>
        <table>
        <tr wicket:id="movie_list">
                <td>
                        <span wicket:id="title">The Matrix
                </span> (
                        <span wicket:id="year">1999
                        </span>)
                </td>
        </tr>
        </table>
        </body>
```

MovieListPage.html

Wicket elements (title,year,movie_list)

Data Model

We are adding the data of movies (title and year)

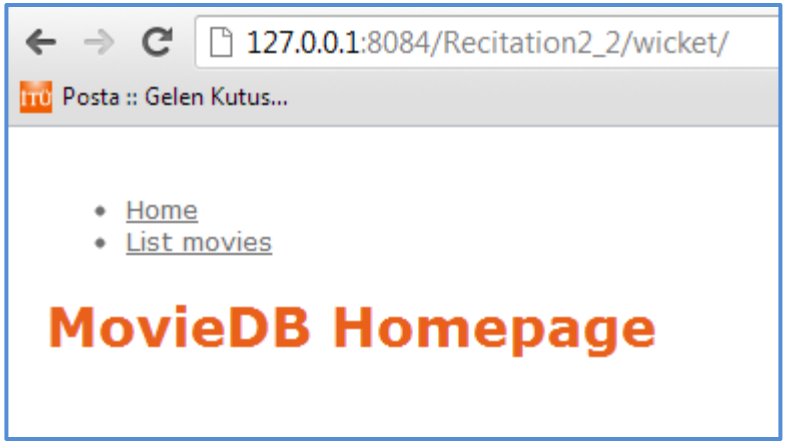
- The Shining-1980
- Barton Fink -1991

```
public MovieListPage() {
    List<Movie> movies = new LinkedList<Movie>();
    Movie movie1 = new Movie("The Shining");
    movie1.setYear(1980);
    movies.add(movie1);
    Movie movie2 = new Movie("Barton Fink");
    movie2.setYear(1991);
    movies.add(movie2);

    ListView movieListView = new ListView("movie_list", movies) {

        @Override
        protected void populateItem(ListItem item) {
            Movie movie = (Movie) item.getModelObject();
            item.add(new Label("title", movie.getTitle()));
            item.add(new Label("year", movie.getYear().toString()));
        }
    };
    this.add(movieListView);
}
```

MovieListPage.java

Wicket elements (title,year,movie_list)

Data Model


127.0.0.1:8084/Recitation2_2/wicket/
Posta :: Gelen Kutus...
Home
List movies
MovieDB Homepage


127.0.0.1:8084/Rec
Posta :: Gelen Kutus...
Home
List movies
Movie List
The Shining (1980)
Barton Fink (1991)

Creating Collection

We will implement a class that will represent a movie collection

```
public class MovieCollection {
        private List<Movie> movies;
        public MovieCollection() {
                this.movies = new LinkedList<Movie>();
                }
        public List<Movie> getMovies() {
                return this.movies;
                        }
        public void addMovie(Movie aMovie) {
                this.movies.add(aMovie);
                        }
        public void deleteMovie(Movie aMovie)           {
                this.movies.remove(aMovie); } }
```

Creating Collection

Movie collection is created in Application.java

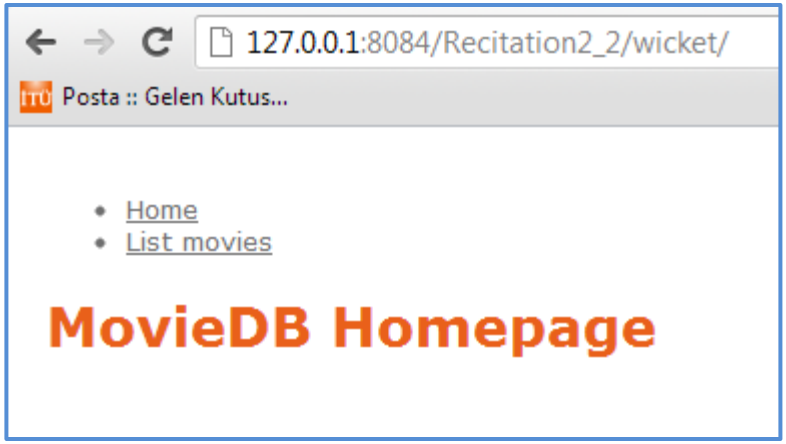
```
public class Application extends WebApplication {
        private MovieCollection collection;
        public Application() {
                this.collection = new MovieCollection();
                Movie movie1 = new Movie("The Shining");
                movie1.setYear(1980);
                this.collection.addMovie(movie1);
                Movie movie2 = new Movie("Barton Fink");
                movie2.setYear(1991);
                this.collection.addMovie(movie2);
                        }
        public Class getHomePage() {
                return HomePage.class; }
        public MovieCollection getCollection() {
                return this.collection; } }
```

Creating Collection

```
public MovieListPage() {
        Application app = (Application) this.getApplication();
        MovieCollection collection = app.getCollection();
        List<Movie> movies = collection.getMovies();
        PropertyListView movieListView = new PropertyListView("movie_list", movies) {
        @Override protected void populateItem(ListItem item) {
                item.add(new Label("title"));
                item.add(new Label("year")); }
                 };
        this.add(movieListView);
        }
```

We create an object from Application class without writing data of movies in MovieListPage.java.

Creating Collection



Adding Link for Movie Details

We will create a wicket page «MovieDisplayPage» for displaying details of movies.

```
public final class MovieDisplayPage extends BasePage {
        public MovieDisplayPage(Movie aMovie) {
                this.add(new Label("title", aMovie.getTitle()));
                this.add(new Label("year", Integer.toString(aMovie.getYear())));
                }
        }
```

Adding Link for Movie Details

We will create a java class
for controlling movie links
«MovieDisplayPageLink».

```
public class MovieDisplayPageLink extends Link {
        private Movie movie;
        public MovieDisplayPageLink(String id, Movie aMovie) {
                super(id);
                this.movie = aMovie; }
        @Override
        public void onClick() {
                this.setResponsePage(new MovieDisplayPage(this.movie)); } }
```

Adding Link for Movie Details

MovieListPage.html

```
<body>
    <div wicket:id="mainNavigation">links to common pages</div>

    <h2>Movie List</h2>

    <table>
      <tr wicket:id="movie_list">
        <td>
          <a href="#" wicket:id="movie_link">
            <span wicket:id="title">The Matrix</span>
            (<span wicket:id="year">1999</span>)
          </a>
        </td>
      </tr>
    </table>
  </body>
```

Adding Link for Movie Details

MovieListPage.java

```
public final class MovieListPage extends BasePage {
  public MovieListPage() {
      Application app = (Application) this.getApplication();
      MovieCollection collection = app.getCollection();
      List<Movie> movies = collection.getMovies();

      PropertyListView movieListView =
            new PropertyListView("movie_list", movies) {

                @Override
                protected void populateItem(ListItem item) {
                    Movie movie = (Movie) item.getModelObject();
                    MovieDisplayPageLink movieLink = new MovieDisplayPageLink("movie_link", movie);
                    movieLink.add(new Label("title"));
                    movieLink.add(new Label("year"));
                    item.add(movieLink);
                }
            };
      this.add(movieListView);
   }
}
```

Adding Link for Movie Details

127.0.0.1:8084/Recitation_3_3/wicket/

Posta :: Gelen Kutus...

- Home
- List movies

MovieDB Homepage

127.0.0.1:8084/Recitation_3

Posta :: Gelen Kutus...

- Home
- List movies

Movie List

The Shining (1980)

Barton Fink (1991)

127.0.0.1:8084/Recitati

Posta :: Gelen Kutus...

- Home
- List movies

The Shining

Year: 1980