7 Seaside: Composition

In this weeks exercises we will compose and extend the different components we have written during the last weeks exercises. We will now extend the existing code with an appealing user interface and with a snappy navigation for our imaginary theater company.

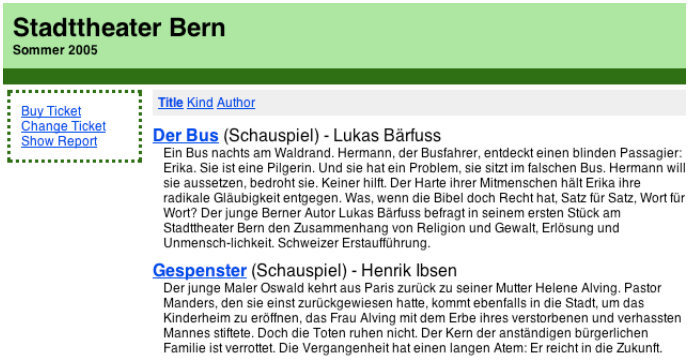

Figure 1: View of `STMainFrame`

7.1 Frame, Subcomponent and Backtracking

Exercise 1 Create a new subclass of `WAComponent` and register it as a new entry point to your application. Render into different div-tags the name of the theater and the current season; you can find this information in the model. Also create a simple menu that is empty for now. Create a style-sheet to make the application look nicer.

Exercise 2 Add an instance variable to your main-frame to hold a child component. Create a method `#buyTicket` that initializes the variable with a new instance of `STBuyTicketTask` and send `#buyTicket` in the initialization method of the component. Place the child beside the menu you have created before. Don't forget to implement the message `#children`, else you will sooner or later run into troubles. Create a menu item called *Buy Ticket* that sends the message `#buyTicket` when clicked. Enjoy the application with the halos turned on.

Exercise 3 Test the new functionality you implemented. Especially try out the behavior of the application when using the back-button. Try clicking on *Buy Ticket*, hit the back-button of your web-browser and then click on any link or control within the child-component. Why do you get an error? Fix the problem and make sure everything works as expected.

7.2 Reuse of Components

In this part of the exercises you are basically free about the implementation details of a new requirement of the application: The theater company wants to

be able to let the customers return tickets and exchange them with another one from the same play but a different show.

Exercise 4 Use the id of the ticket to identify the one to be replaced. Probably you need to improve the model to make the necessary mutations possible. Also write tests to ensure it works as expected. For the web interface try to write as few lines of code as possible. Reuse the existing components that you have written in the previous steps. You might also want to use components provided by the framework. The example solution requires 7 lines of code, including the validation of the ticket id. Can you do it with less lines of code?

7.3 Reporting

Exercise 5★ Create a new component called `STShowReport` showing a report of all the shows from the model as seen in Figure 2. Use `WABatchedList` to enable the batching of the huge list and only display 10 items at once. For the reporting you might want to use `WATableReport` or write your own component. By default the list should be sorted according to the timestamp. Add the new component to the menu in the main-frame.

Figure 2: View of `STShowReport` with halos toggled on

Please save the Monticello package `Tutorial` and send it by mail to st-staff@iam.unibe.ch. Attach your written solutions that are not part of the source-code to the mail or hand them in as hardcopy at the beginning of the next exercise session. Your mail and solutions should be clearly marked with names and matrikel numbers of the solution authors.