CSCM@intern

Interne Kommunikation in der Forschungsgruppe Kooperationssysteme (CSCM)

Florian Ott
um 18:39 am 7. August 2014
Permalink (http://sns.cscwlab.de/cscm/2014/08/07/mirror-entwicklung-css/)

Begriffe: CMF (44), CommunityMirrors (118), CSS (6), FXML, muc (22)

Mirror-Entwicklung CSS

Bei meinen kurzen Tests mit dem heute habe ich auch einen Blick in die aktuelle FXML-Umsetzung geworfen. Dazu sollten wir uns auf jeden Fall nochmal (spätestens nach der Konferenz – vielleicht aber schon für die noch ausstehenden Partikel-Designs) Gedanken machen. Aktuell sind die Partikel-Layouts noch stark vergleichbar mit frühen HTML-Seiten, wo Formate direkt in HTML-Tags als beispielsweise font-type, font-size, etc. gesetzt wurden. Mit CSS bietet ja hier alle Vorteile des modernen Layoutings, d.h. Schriftgrößen, Farben, Fonts, Positionen (jenseits von Layout-Managern), Hintergründe und ähnliches sollten nicht in den FXMLs der Partikel (so aktuell eigentlich alles vermischt drin steht) sondern in entsprechend eingebundenen Theme-CSS-Files auftauchen. Auch sollten wir hier die Kaskaden von CSS nutzen und beispielsweise eine font.css anlegen, die dann theme-spezifisch in ALLEN Elementen eingebunden ist und in der beispielsweise Elemente wie Basisfont, H1 / Überschrift 1, Aufzählungen, etc. inkl. Standard-Farben, Größen, Zeilenhöhen und Margins definiert werden. Zusätzlich sollte jeder Partikel eine partikeltyp.css – also beispielsweise eine person. haben, in der zugehörige „Sonder“-Bestandteile definiert werden. Wiederkehrende Formatierungen sollten wir unbedingt über entsprechende Klassen abbilden, also z.B. class=“small“ für kleinen Text, etc.

Für die Zukunft (v2/v3) sollte das Styling dann über weitere SASS / LESS Variablen und ggf. FXML-Templates anpassbar sein, so dass sich das gesamte Layout über einfache Theme-Parameter anpassen und ggf. mit eigenen Themes erweitern lässt. Wie ist eigentlich das Theming aktuell gelöst? Daraus bin ich auf die Schnelle nicht so richtig schlau geworden und dazu hatten wir doch vor ca. einem Jahr schon mal einen ausführlicheren Workshop (damals noch zusammen mit Johannes), in dem wir auch die kaskadierenden properties sowie die Auslagerung aller Styles analog zu meinem einleitenden Absatz über besprochen hatten. Haben wir das schon in irgendeiner „benutzbaren“ Form und wenn ja, wie könnte ich mein eigenes Theme anlegen / überschreiben, um mal etwas zu experimentieren (für die MuC natürlich relativ egal, wenn es das noch nicht gibt, da wesentlich wichtiger, dass die Anwendung erst mal überhaupt läuft und brauchbar aussieht – also rein interessehalber)?

Michael Koch
um 20:11 am 7. August 2014

Schreiben Sie die Anforderung bitte als Task in Jira (falls nicht eh schon geschehen)? (für v2 …) – ich finde nämlich, dass wir sowas alles dort sammeln sollten

Florian Ott
um 12:36 am 8. August 2014

Guter Punkt und … done:
Concolidate cascading theming mechanism
Create base files
Integrate LESS or SASS compiler for / FXML
Externalize
Neben den Tickets ist das grundsätzliche Vorgehen hier aber auch etwas, das sich nicht

sinnvoll in ein Ticket packen lässt, deshalb der Hinweis aus langjähriger Webdev-Erfahrung und auch aus den verschiedenen Mirror-Erfahrungen @AN und @EL: Essentiell ist aus meiner Sicht, nicht ausschließlich auf den SceneBuilder Output zu vertrauen (das haben in den Anfängen des Webdev leider auch zu viele Entwickler bei HTML und beispielsweise FrontPage gemacht), sondern wo immer es geht, direkt ein eigenes -File mit sinnvollen / wiederverwendbaren Klassen und ID-spezifischen Styles anzulegen, einzubinden und dann vom GUI-Editor generierte Styles dort hin auszulagern.

Florian Ott
um 13:36 am 10. August 2014

Wie besprochen habe ich mich mir gerade mal eine Std. Zeit genommen und mich selbst etwas mit JavaFX-CSS beschäftigt, um das Keyboard zu stylen; Zwischenstand im SVN. Hier lässt sich wirklich mit Minimalaufwand fast ALLES wunderbar gestalten, aber wir sind leider – wie schon oben und auch nochmal in meiner Dokumentation bzw. dem Refactoring-Task dazu unter https://sociotech.atlassian.net/browse/MUC-184 beschrieben – extrem weit von sauberem Code entfernt, der konsistent den von angedachten (und von Microsoft mit WPF schon lange propagierten) ViewModel-Ansatz verwendet. Entsprechend habe ich jetzt auch nur ein paar -Klassen im Code „injected“ und einen Großteil der dort bisher enthaltenen Pseudo-Formatierung auskommentiert, Rest dann für v2.

Hier nochmal in Kürze, worauf wir für alle VISUELLEN Elemente (also alles, was irgendwo sichtbar auf dem Screen auftaucht) aus meiner Sicht achten sollten; vielleicht sollten wir dazu wirklich nochmal einen kurzen „Workshop“ nächste Woche ansetzen, denn bisher findet das fast noch nirgendwo statt:

1. Strikte Trennung von Layout-Struktur (Container, Verschachtelung, Klassen- und ID-Zuweisung), Styling (Farben, Formen, Effekte, Schrift, Größen, Abstände) und Verhalten (Aktionen, Auslösen von Animationen, Datenbindung, etc.).

2. Layout-Struktur ausschließlich in FXML – hier aber bitte nicht stumpf per Copy & Paste aus einem GUI-Editor inkl. der sort ggf. enthaltenen Formate, sondern sinnvoll strukturiert; im Optimalfall selbst „geschrieben“. Variablen später über SASS / LESS, vgl. http://sns.informatik.unibw-muenchen.de/browse/MUC-180.

3. Jegliches Styling (auch Größen!) ausschließlich in externen -Files; hier sollvollerweise Kaskadierung entsprechend http://sns.informatik.unibw-muenchen.de/browse/MUC-182 und Auslagerung der wichtigesten konfigurierbaren Elemente (Schrift, Farben, etc.) in Variablen entsprechend http://sns.informatik.unibw-muenchen.de/browse/MUC-180.

Beim Keyboard hatten wir bisher weder FXML noch CSS. Die Zwischenversion verwendet jetzt wenigstens , die visuelle Struktur ist aber immer noch Code-basiert anstatt FXML-basiert. Wie schon im Ticket beschrieben würde ich auch stark dafür plädieren, hier das Rad nicht neu zu erfinden, sondern ein gut gewartetes, offen verwendbares existierendes Projekt einzubinden (das gleichzeitig auch den ViewModel-Ansatz verwendet), z.B. https://github.com/comtel2000/fx-experience/tree/master/fx-onscreen-keyboard.

Bei den Partikel-Styles haben wir aktuell zwar bereits ein FXML (Struktur) und eine zugehörige Controller-Klasse (Verhalten), aber die Styles sind wildest im FXML eingemischt. In der „Person_inDetail.fxml“ könnte man beispielsweise ca. 70% des Codes (Positionierung, Größen, Farben, Rundungen, Schrift, Icons) einsparen und über abbilden (gleiches gilt natürlich für alle anderen bereits existierenden Partikel). Auch die Icons ließen sich einfach über „Minimal-Container-Klassen“ einbinden, so wie von Bootstrap vorgesehen und dann auch unabhängig von der Text-Größe stylen. Für die v1 würde ich den bestehenden Entwicklungsansatz aus Zeitgründen noch beibehalten, aber für die v2 sollten wir uns dringend nochmal in größerer Runde zusammensetzen, um ein sinnvolles und v.a. nachhaltiges Prozedere für die Weiterentwicklung zu besprechen.

Florian Ott
um 13:48 am 10. August 2014

Noch ein Nachtrag: ich hatte das zwar im SVN-Kommentar auch geschrieben, aber sicherheitshalber: Ich habe nirgendwo eine Einbindung der Open Sans Font in CSS gefunden und deshalb mal ein entsprechendes CSS-File im keyboard-Ressources-Ordner als Übergangslösung angelegt. Das File sollte da natürlich wieder raus und in den entsprechenden Theme-Ressourcen-Ordner. Generell sollten wir uns Gedanken machen, wir wir externe Fonts durchgängig und für Entwickler mit möglichst geringem Aufwand in CodeBehind und CSS einbinden können. Mein Vorschlag wäre eine theme.proerty (z.B. theme.text.font), in der nur ein Schriftname angegeben wird, z.B. „Open Sans“, darauf basiert könnten wir dann ein entsprechendes CSS (ggf. mit Fallback zu Google Fonts) laden und auch versuchen die Schriften dynamisch zu laden, sofern nicht lokal vorhanden. Siehe dazu auch:
http://fxexperience.com/2012/12/use-webgoogle-fonts-in- ____/
https://github.com/fxexperience/code/blob/master/FXExperienceControls/src/com/fxexperience/ ____

Michael Koch
um 08:49 am 11. August 2014

Im Sinne der Weiterentwickelbarkeit ist es sicher sinnvoll das, was wir in Web Technologies predigen, auch bei der -Entwicklung zu beherzigen. Und was man dazu jetzt für Version 1 noch berücksichtigen kann ist es auf jeden Fall wert. Ich bin aber auch gerne bei einem Workshop zur zukünftigen, noch nachhaltigeren Strukturierung dabei.

Michael Koch
um 20:12 am 13. August 2014

Ich habe mir mal in die Todos geschrieben, dass ich im Oktober mal eine neue View für einen anderen Anwendungsbereich generiere (mit leicht verändertem Aussehen) – dann werden wir sehen, wie die Anpassbarkeit von Hintergrund, Partikellayout, Flow-Verhalten, Partikelauswahl etc. funktioniert und können das bis Jahresende glatt ziehen.

Michael Koch
um 20:14 am 13. August 2014

Ab Oktober habe ich ja endlich wieder Entwicklungskapazitäten frei 🙂 und freue mich schon darauf endlich mitmischen zu dürfen … Irgendwie sind aktuell die „Randaufgaben“ der MuC zu „deckend“ – Bis heute Mittag haben wir noch am Programmheft gebastelt – dann ist das endlich zum Drucker – und eine Stunde später meldet sich der erste „freie Tester“ mit einem gefundenen Fehler …

Florian Ott
um 19:14 am 23. Juli 2014
Permalink (http://sns.cscwlab.de/cscm/2014/07/23/ein-paar-javafx-links/)

Begriffe: AeroFX, CMF (44), FontAwesome, JavaFX (4), OSGi (7), pdf (7)

Ein paar JavaFX-Links …

… die ggf. für die weitere -Entwicklung hilfreich sein könnten:

FontAwesomeFX (nachdem Eva hieraus die überarbeiteten Icons bereitstellen wird):
https://bitbucket.org/Jerady/fontawesomefx

Anzeige von PDF-Dokumenten (haben wir die überhaupt?)
http://stackoverflow.com/questions/18207116/displaying-pdf-in-

AeroFX (vielleicht kann man sich da das Blurring für den Hintergrund abschauen):
http://www.guigarage.com/2014/06/sneak-peek-aerofx/

Für die Zukunft ggf. mal wieder interessant: OSGi und gehen wohl inzwischen doch: http://puces-blog.blogspot.co.nz/2014/05/drombler-fx- -8-support.html

Florian Ott
um 13:31 am 29. Juli 2014

Vielleicht ebenfalls noch interessant für die AwesomeFont-Icon-Integration ins CMF: http://dlemmermann.wordpress.com/2014/07/22/ -tip-12-define-icons-in- /

Florian Ott
um 13:36 am 30. Juli 2014

Und nochwas zum Thema Graph: http://blog.jeffreyguenther.com/post/61616273121/exploring-the-possibility-of-graph-layouts-in-