ADAP CW#05

Projektdaten

Projekt name:	3D-Printing
Projekt repository:	https://github.com/**ModischFabrications**/wahlzeit
This week's tag:	adap-hw04 on master
Homework diff:	https://github.com/ModischFabrications/wahlzeit/compare/adap-hw03...adap-hw04
CI:	https://travis-ci.org/**ModischFabrications**/wahlzeit
Docker Hub Repo:	https://hub.docker.com/r/**modischfabrications**/wahlzeit

Hausaufgaben

Implementation

Neue Klassen (falls vorhanden)

PrintPhoto, PrintPhotoManager, PrintPhotoFactory

PrintPhotoTest, PrintPhotoManagerTest, PrintPhotoFactoryTest, PrintPhotoTestSuite

Veränderte Klassen (falls vorhanden)

Jede Referenz auf Photo*, die im Programmablauf relevant ist:

NotifyUsersAboutPraiseAgent, PersistPhotoAgent, AbstractWebPartHandler, AdminUserPhotoFormHandler, EditUserPhotoFormHandler, FlagPhotoFormHandler, PraisePhotoFormHandler, SendEmailFormHandler, ShowAdminPageHandler, ShowPhotoPageHandler, ShowUserHomePageHandler, ShowUserPhotoFormHandler, TellFriendFormHandler, UploadPhotoFormHandler, ModelMain, Client, PhotoFilter, PhotoManager, PhotoUtil, UserSession, StaticDataServlet

Erklärung

Ich habe diese Woche mit der Korrektur der Pfade begonnen, dabei habe ich das Projekt auf die geltende Konvention der Package/Suite Struktur für Tests angepasst.

Dann habe ich mich in das bestehende System einarbeitet und mich für "PrintPhoto" entschieden, also einer Submenge von Fotos, die 3D-gedruckte Objekte enthält.

Natürlich habe ich vor der Implementierung damit angefangen Tests zu schreiben, hierbei habe zuerst Stubs der zu implementierenden Klassen erzeugt. Danach habe ich Test sowohl für das PrintPhoto selbst als auch für die dazugehörigen Manager und Factory implementiert. Die größte Schwierigkeit war dabei, dass es keine Tests der Basisklassen gab und daher keine einfach ersichtliche Verwendung der Klassen aufgezeigt wurde. Weitere Details zu den Tests sind [Which

tests did you add and why?] zu entnehmen.

Dann habe ich die Klasse “Photo” mit “PrintPhoto” ergänzt und dort das Druckmaterial als Attribut hinterlegt. Zulässige Typen sind in einem enum definiert, das Material kann über einen Getter von Außen angefragt werden, so können Werte weder unbeabsichtigt verändert noch falsche Werte überlegen werden. Da das Material nicht immer ermittelt werden kann ist ein Fallback-Feld “UNKOWN” möglich, bei einer anderen Anwendung könnte ein korrekter Wert zugunsten der Konsistenz erzwungen werden.

Dann habe ich den zugehörigen Manager und die Factory implementiert. Hier habe ich mich dafür entschieden alle Methoden zu wrappen, die Photos bekommen oder zurückgeben. So kann die neue Klasse das interne Verhalten weiter nutzen, bietet aber dennoch eine saubere Schnittstelle an. Zusätzlich musste die Instanzverwaltung überarbeitet werden, um immer die Instanzen der Kindklasse zu nutzen und inkonsistente Präsentationen durch verschiedene Instanzen auszuschließen. Die gleiche Instanz konnte nicht wiederverwendet werden, da die Typdefinition der Kindklasse notwendig war.

Dann habe ich alle Erwähnungen der Basisklasse durch die neue Implementierung ersetzt und die Software erneut getestet.

Zuletzt habe ich alle Änderungen gepusht und den letzten Commit mit dem oben genannten Tag für die Abgabe markiert.

Fragen

Why did you extend the Photo class/Why did you not just replace the Photo class?

Ich habe die Fotoklasse ergänzt, um im Sinne einer modularen Architektur die neuen Features vom „Core“ zu trennen. So können innere Klassen auch für andere Anwendungen wiederverwendet werden, ohne die Implementierung oder die zugehörigen Tests anpassen zu müssen. Dies erhöht zugleich auch die Verlässlichkeit, da diese Kernklassen bereits „Production-ready“ sind und viele Fehler bereits aufgedeckt wurden.

Which tests did you add and why?

Ich habe vorallem Tests zu den Konstruktoren und den Erzeugern geschrieben, da diese den kritischen Teil der Anwendung (die Interaktion mit AppEngine) absichern und vor Regressionen schützen. Weitere waren stichpunktartig und vor allem zum Testen der überschriebenen Methoden gedacht. Besondere Aufmerksamkeit war dabei gefordert die Photoklasse zu testen, da das AppEngine-Framework keinen trivialen Junit-Test erlaubt hat und daher der komplexere Weg mit den Providern genutzt werden musste.