- Entwurfsbeschreibung - Vorprojekt -

Kita Tauschbörse

Version: 1.0

Projektbezeichnung	Kita Tauschbörse	
Projektleiter	F. Teichmann	
Verantwortlich	B. Kalloch, R. Rößling	
Erstellt am		
Zuletzt geändert	15.04.12 14:39	
Bearbeitungszustand		in Bearbeitung
		vorgelegt
	X	fertig gestellt
Dokumentablage		
V-Modell-XT Version	1.3	

INHALTSVERZEICHNIS

1 ALLGEMEINES

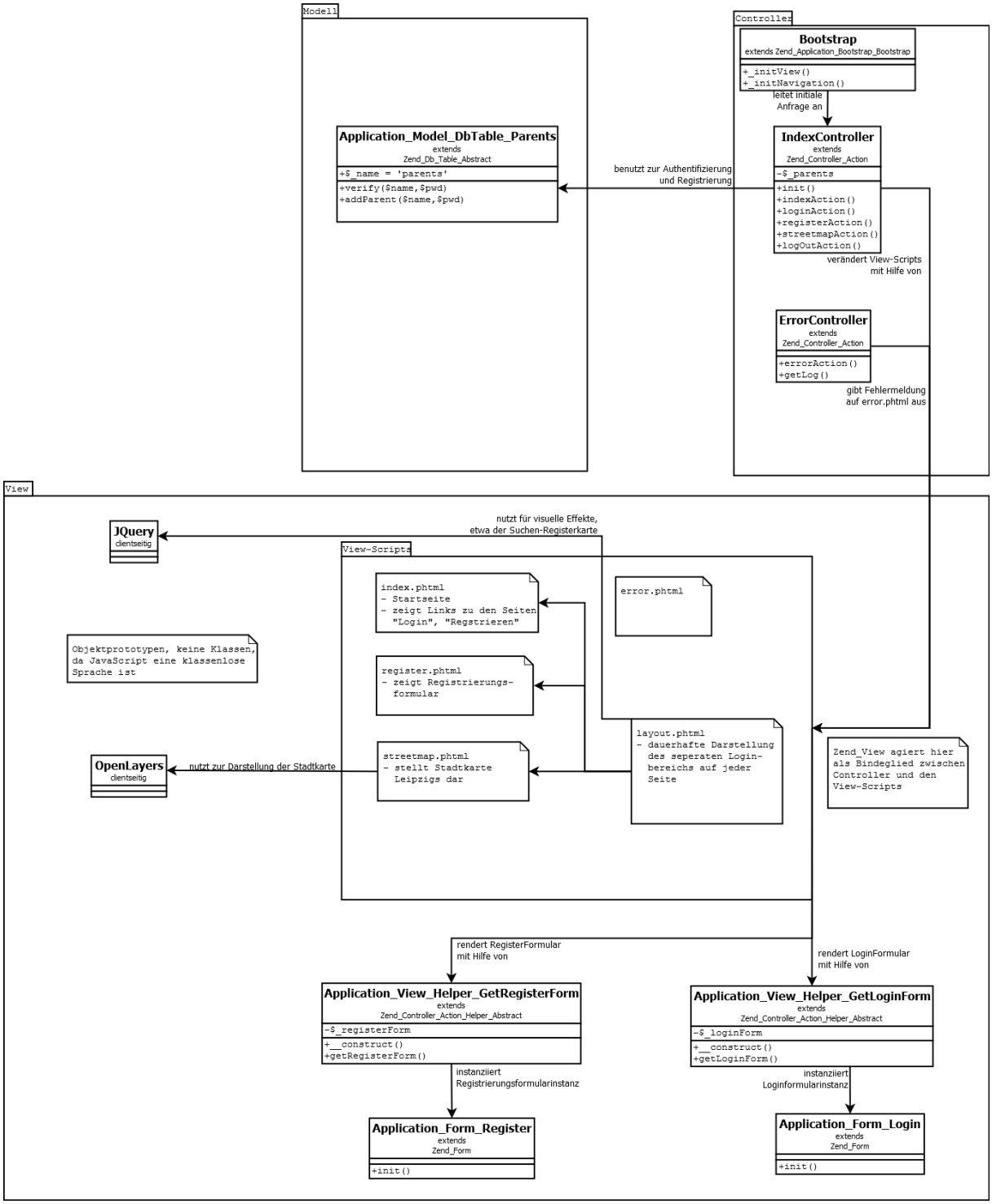
Diese Softwarestudie soll dazu dienen, die Durchführbarkeit unseres Projekts, welches ein Tauschbörsensystem für Kita-Plätze der Stadt Leipzig anbietet, abzuschätzen. Hierbei soll als Prototyp eine rudimentäre Karte mit allen Standorten der Kita-Plätze von Stadt Leipzig visualisiert werden. Zudem erstellen wir eine Anmeldeapplikation für die Website mit einer funktionierenden Benutzeroberfläche.
Das Projekt ist in unserem Mercurial Repository gespeichert, auch alle späteren Versionen des Projektes werden in diesem Repository erstellt und gespeichert werden.

2 PRODUKTÜBERSICHT

Der Nutzer kann mittels eines funktionierenden Webbrowsers einer neueren Generation (JavaScript zwingend benötigt) die Softwarestudie betrachten. Wenn gewünscht, kann der Nutzer sich auf der Website anmelden und die durch OpenLayers visualisierte Karte betrachten und nähere Informationen, hier die genaue Adresse der einzelnen Kita-Plätze, durch Anklicken der einzelnen Marker erhalten. Damit setzt die Softwarestudie die fundamentalen Funktionen unseres Projektes rudimentär um. Im Einzelnen sind hier **/LF 10/** - **/LF 30/** sowie **/LF 101/** gemeint, welche näher in dem Projektvertrag bzw. dem Projektvorschlag beschrieben werden.

3 GRUNDSÄTZLICHE STRUKTUR- UND ENTWURFSPRINZIPIEN FÜR DAS GESAMTSYSTEM

Unsere Softwarestudie ist nach einem bekannten und etablierten Designmuster, Model - View - Controller (MVC), aufgebaut. Das Model versorgt die Webanwendung mit Daten aus unserer Datenbank, in der die wichtigsten Informationen zu den Kitas und den Nutzern gespeichert sind. Das View generiert die HTML-Ausgabe, d.h. er setzt die vom Model bereitgestellten Daten im Browser um. Der Controller entscheidet, was mit den übergebenen Parametern geschieht und informiert das Model und/oder das View über Veränderungen. Er steuert damit die Applikation.
Das folgende UML-Diagramm beschreibt den Aufbau der Softwarestudie, aufbauend auf den MVC Prinzip:



3.1 Model: Application_Model_DbTable_Parents

Das Model wird von *Zend_Db_Table* abgeleitet und stellt Methoden für den Zugriff auf die Eltern-Tabelle in der Datenbank zur Verfügung. Diese Methoden sind zum einen das Authentifizieren des Nutzers. Dabei werden dessen Anmeldedaten mit den in der Datenbank abgelegten Daten abgeglichen. Zum anderen wird mit Hilfe der Klasse eine Registrierung ermöglicht. Es lassen sich neue Tupel in die Tabelle eintragen.

3.2 Controller: IndexController

Der Controller wird von *Zend_Controller_Action* abgeleitet. Er besteht aus einzelnen Actions, mit denen sich der View beeinflussen lässt. Eine bestimmte Controller-Action wird beim Aufruf des zugehörigen View-Scripts aktiviert.
Zusätzlich enthält Zend noch einen Error-Controller, der auf einem gesonderten View-Script jegliche Fehlermeldungen ausgibt, die während des Betriebes des Scripts auftreten.

3.3 View: Zend_View

Der View setzt sich aus zwei Bestandteilen zusammen. Das sind die View-Scripts, auf welchen die eigentlichen Ausgaben generiert werden. Der Zugriff auf diese View-Scripts wird mit Hilfe des Views ermöglicht. Das ist in unserem Fall die Standardklasse *Zend_View*. Mit ihr ist es möglich, Elemente direkt auf den View-Scripts zu generieren. Hierzu können dem View sogenannte View-Helpers zur Seite gestellt werden, die etwa bestimmt Objekte bereits erzeugen. Wir nutzen diese Eigenschaft, um die beiden Formulare für den Login und die Registrierung bereits zu erzeugen, sodass diese nicht mehr vom View instanziiert werden müssen, sondern sofort dargestellt werden können. Das hat den Vorteil, dass die Formulare so nicht bei jeder Darstellung neu instanziiert werden müssen. Die Klassen *Application_View_Helper_GetLoginForm* und *Application_View_Helper_GetRegisterForm* sind derartige View-Helpers.

3.4 Application-Model von Zend

Da das Projekt mit dem Zend Framework erstellt wird, orientiert sich bereits die Implementierung des Vorprojektes am Application-Modell von Zend. Die *Zend_Application* Klasse arbeitet im Hintergrund. Sie beinhaltet beispielsweise einen Autoloader, der automatisch benötigte Klassen in Skripte einbindet, ohne, dass sie vorher von Programmierer manuell eingebunden werden. Es existiert einen separate Konfigurationsdatei, in welcher man häufig benutzte und veränderliche Einstellungen und Ressourcen, wie etwa die Zugriffsdaten für die Datenbank, zentral vermerken kann. *Zend_Application* wird zuerst, vor allen anderen Klassen, beim Aufruf der Seite gestartet. Über eine gesonderte Klasse, Bootstrap, lassen sich bestimmte Einstellungen beim Programmstart aufrufen. Im Vorprojekt nutzen wir *Bootstrap* um einen einheitlichen View und eine zentrale Navigation einzuführen. Da *Zend_Application* eine Menge Automatismen mit sich bringt, ist die strenge Einhaltung der Ordnerhierarchie und Namenskonventionen unbedingt erforderlich.

4 GRUNDSÄTZLICHE STRUKTUR- UND ENTWURFSPRINZIPIEN DER EINZELNEN PAKETE

Unsere Softwarestudie besitzt selbst keine weiteren Pakete, deren Struktur- und Entwurfsprinzipien beschrieben werden könnten. Das gesamte System selbst wird von einem Client mittels einem Webbrowser aufgerufen, wobei der *IndexControllor* dafür zuständig ist, alle vom Client erwünschten Funktionen zu navigieren. In unserer Softwarestudie sind das die Grundfunktionen *register, login, streetmap* und *logout.*