jPolls Entwurfsbeschreibung

Inhaltsverzeichnis

1 Einleitung

Das vorliegende Dokument beschreibt die grundsätzliche Struktur und Entwurfsprinzipien von jPolls. jPolls ist ein auf J2EE-Architektur aufbauendes Web-Framework, welches die Erstellung von Umfragesystemen vereinfachen soll. Dabei ermöglichen spezielle web-basierte Umfragesysteme eine Akquirierung und Verwaltung von sehr großen Informationsmengen und können für statistische oder Marketingzwecke eingesetzt werden. Das jPolls-Framework unterstützt den Softwareentwickler beim Erstellen, Modifizieren und Löschen von Umfragen, sowie bei der Stimmenabgabe. Es realisiert die Nutzerverwaltungs-, Authentifikations- und Autorisierungsfunktionalitäten, unterstützt verschiedene Persistenzframeworks, stellt die Möglichkeit zur Verfügung, mehrsprachige Anwendungen entwickeln zu können und bietet eine leicht an eigene Bedürfnisse anpassbare Architektur. Der Schwerpunkt der Framework-Untersuchungen liegt einerseits auf dem Aufbau des Systems als Ganzes, andererseits aber auch auf detaillierten Klassenstrukturen sowie deren Beziehungen untereinander.

2 JPolls-Architektur

Das jPolls ist ein leichtgewichtiges J2EE-Framework, welches für eine vertikale Aufgabe der webunterstützten Umfrageverwaltung konzipiert worden ist. Die folgenden Unterabschnitte beschreiben das Framework auf verschiedenen Abstraktionsniveaus. Zunächst wird im Abschnitt 2.1 der prinzipielle Aufbau sowie die Integrationsmöglichkeiten von jPolls aufgezeigt. Im Abschnitt 2.2 werden schließlich die feineren Paketstrukturen sowie deren Klassen dargestellt.

2.1 Systemaufbau

Das jPolls-Framework folgt dem Model-View-Controller-Konzept (MVC) und ist sehr modular aufgebaut. Die Hauptschichten dabei sind: die Präsentationsschicht (View), die Geschäftslogikschicht (Controller) sowie die für die Datenkapselung und –persistenz verantwortliche Persistenzschicht (Model). Einzelne Schichten sind über klare Schnittstellen (Persistence Services Interface, Konfigurationsdateien) voneinander getrennt, sodass sie leicht ausgetauscht werden können. Die Abbildung 2-1 verdeutlicht diesen Zusammenhang.

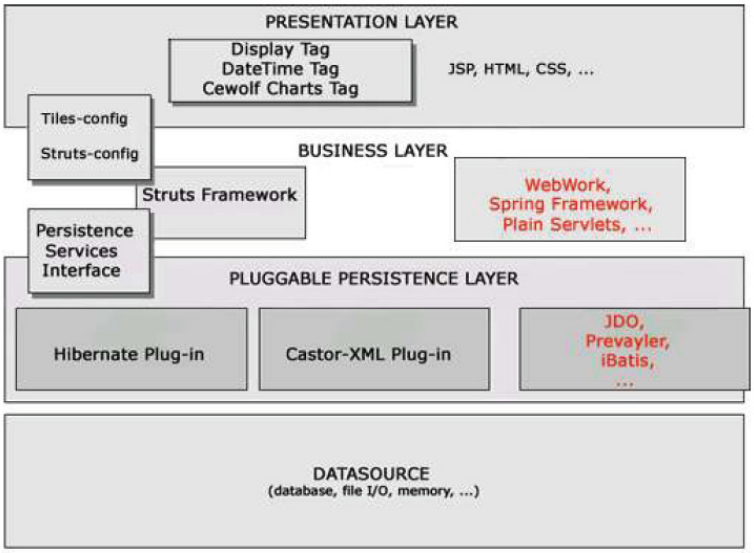

Abbildung 2-1. jPolls-Architektur. Quelle: jPolls Users Guide Revision 0.3

2.1.1. Präsentationsschicht

Die Präsentationsschicht basiert auf der JSP-Technologie sowie HTML, es können jedoch auch andere Frameworks wie z.B. Velocity für die Ausgabe eingesetzt werden.

2.1.2. Geschäftslogik-Schicht

Die Geschäftslogik-Schicht stützt sich auf das Struts-Framework, welches jedoch auch durch ein anderes Plugin ersetzt werden kann. Dabei ist zu beachten das die Logik-Schicht durch das Persistent-Services-Interface an die Persistenzschicht angebunden sein muss und (bei Nutzung von Struts) durch die XML-Konfigurationsdateien mit der Präsentationsschicht kommuniziert.

2.1.3. Persistenzschicht

Die Persistenzschicht von jPolls realisiert eine persistente Speicherung sowie Laden von Daten auf ein beliebiges Medium. Die Persistierung von Daten kann dabei durch ein beliebiges Framework-Plugin erfolgen, welches die „Persistent-Services-Interface“-Schnittstelle implementiert. jPolls bietet selbst eine RDBS-Anbindung durch das Hibernate-Plugin sowie Speicherung von XML-Dateien mittels des Castor-XML-Plugins.

2.2. Paketstrukturen

Das gesamte jPolls-Framework gliedert sich in 6 Pakete, welche im Folgenden beschrieben werden.

2.2.1. Paket org.sf.polls

Die Struktur des Hauptpakets org.sf.polls ist in Abbildung 2-2 dargestellt. Das Paket besteht aus mehreren Klassen, sowie Unterpaketen und ist hauptsächlich für die Übertragung von Umfragen bzw. Umfrageergebnissen zwischen Logik- und Persistenzschicht zuständig. Eine weitere Funktionalität des Pakets ist die Bereitstellung der anwendungsweiten Konfigurations-eigenschaften, welche standardmäßig aus der Datei /polls-config.xml ausgelesen werden.

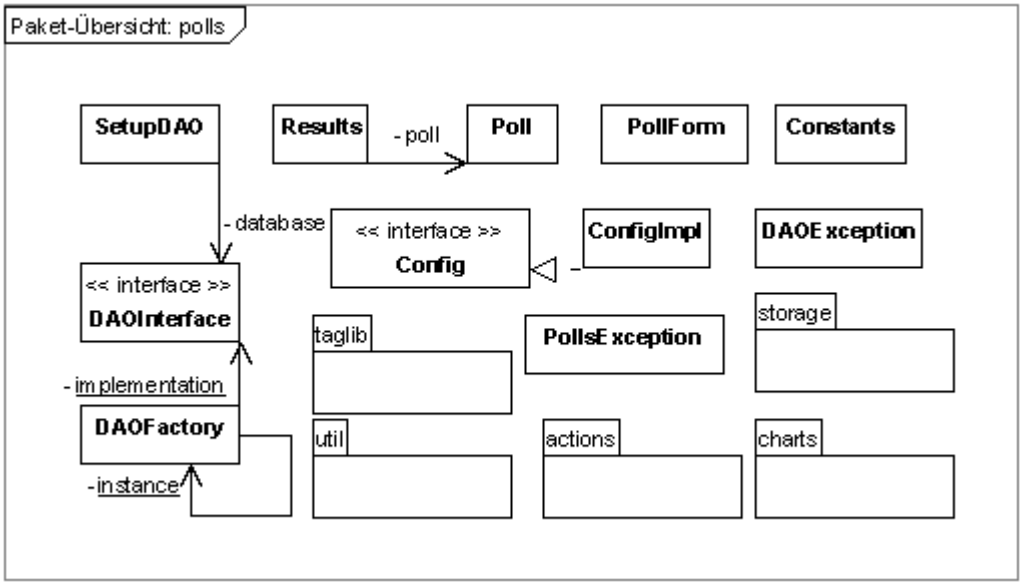

Abbildung 2-2. Package org.sf.polls

Für die Kopplung zwischen Logik- und Persistenzschicht sind die Klassen SetupDAO, DAOFactory sowie das Interface DAOInterface (DAO steht für Data Access Object, also Datenzugriffsobjekt) zuständig. Jede Persistenzschicht muss dieses Interface implementieren, was den Vorteil bringt, dass die Logikschicht von der konkret realisierten Implementierung des Persistenzmechanismus abstrahiert wird.
Das Interface DAOInterface gibt somit die Mächtigkeit der Persistenzmechanismen an. Seine Beschreibung ist in der Abbildung 2-3 dargestellt. DAOInterface bietet die Methoden zum Starten/Stoppen der Persistenzschicht (Methoden startService()/endService()), zum Operieren mit Umfragen und deren Ergebnissen (Methoden …Poll()) sowie zu einer statistischen Umfrageauswertung (getTotal…()).

Der Zugriff auf eine konkrete DAOInterface-implementierende Persistenzschicht geschieht dabei über eine im gesamten Programm eindeutige (weil als Singleton-Pattern realisierte) DAOFactory-Instanz. Die Persistenzschicht wird in dem private-Attribute implementation dieser Instanz gespeichert (siehe Abbildung 2-2). Durch die Eindeutigkeit der DAOFactory-Instanz ist automatisch auch die Eindeutigkeit dieser Persistenzimplementierung gewährleistet.
Für die Initialisierung/Freigabe der Persistenzschicht sowie für das von der Art der Zugriffsschicht unabhängige Laden bzw. Speichern der Daten ist die Klasse SetupDAO zuständig.
Die eigentlichen Umfrage-Werte, die von der Persistenzschicht geladen bzw. gespeichert werden, sind in der Logik-Schicht durch die Klassen Poll und PollForm repräsentiert. Diese Klassen können als Beans betrachtet werden, weil sie allgemeine Attribute (einer Umfrage) mittels Getter-/Setter-Methoden kapseln. Diese Klassen beinhalten z.B. solche Attribute wie Umfragename und -beschreibung, mehrere String-Attribute für verschiedene (untypisierte) Eingaben, sowie Multimedia-Inhalte. Die PollForm-Klasse unterscheidet sich von der Klasse Poll hauptsächlich dadurch, dass sie zusätzliche formularbasierte Attribute sowie die Request-Properties verändern kann. Eine ähnliche Funktionalität weist die Klasse Result auf, welche die Ergebnisse einer Umfrage (z.B. ausgewählte Einträge) kapselt.

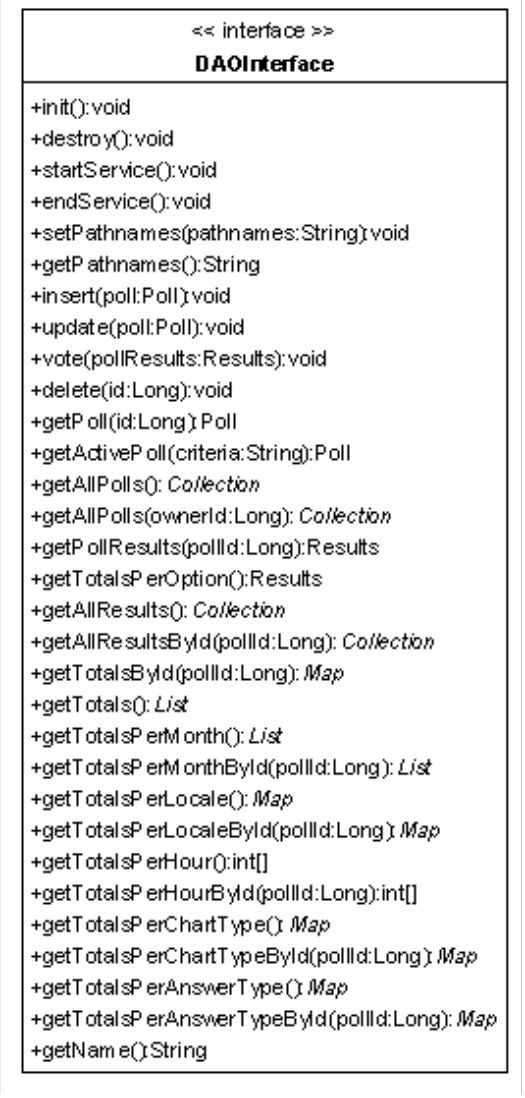

Abbildung 2-4. Kapselung der Persistenzschicht mittels der Schnittstelle DAOInterface

2.2.2. Paket org.sf.polls.actions

Die Struktur des Pakets org.sf.polls.actions ist in Abbildung 2-4 wiedergegeben. Das Paket bildet die Logikschicht und realisiert die Use-Cases aus der Abbildung 2-11. Die Implementierung basiert auf dem Struts-Framework und ist nach dem Command-Pattern (auch als Action-Pattern bekannt) realisiert worden. Dabei werden von Struts die Servlet-Klasse sowie die (abstrakten) Action-Klassen (Kommandos) zur Verfügung gestellt. Das jPolls-Framework leitet die konkreten Action-Nachfahren von der LookupDispatchAction-Klasse ab und realisiert in jeder dieser Klasse ein Use-Case. Dabei werden die von jedem Use-Case benötigten Funktionalitäten in der Basisklasse BaseAction zusammengefasst.
Die wichtigsten von der BaseAction-Klasse realisierten Methoden umfassen eine auf dem Struts-Framework basierende Kontrollflusssteuerung und Navigationsverwaltung zwischen den Controllern / URIs. Weiterhin ermöglicht die Klasse eine Benutzerautorisierung, -authentisierung und Cookies-Verwaltung, sowie Dateimanagement (Upload von Files und Anlegen von Verzeichnissen). Außerdem bestimmt jede BaseAction-Instanz (bzw. deren Nachfahre) die verwendete Persistenz- und Konfigurationsschnittstelle aus dem Anwendungskontext.

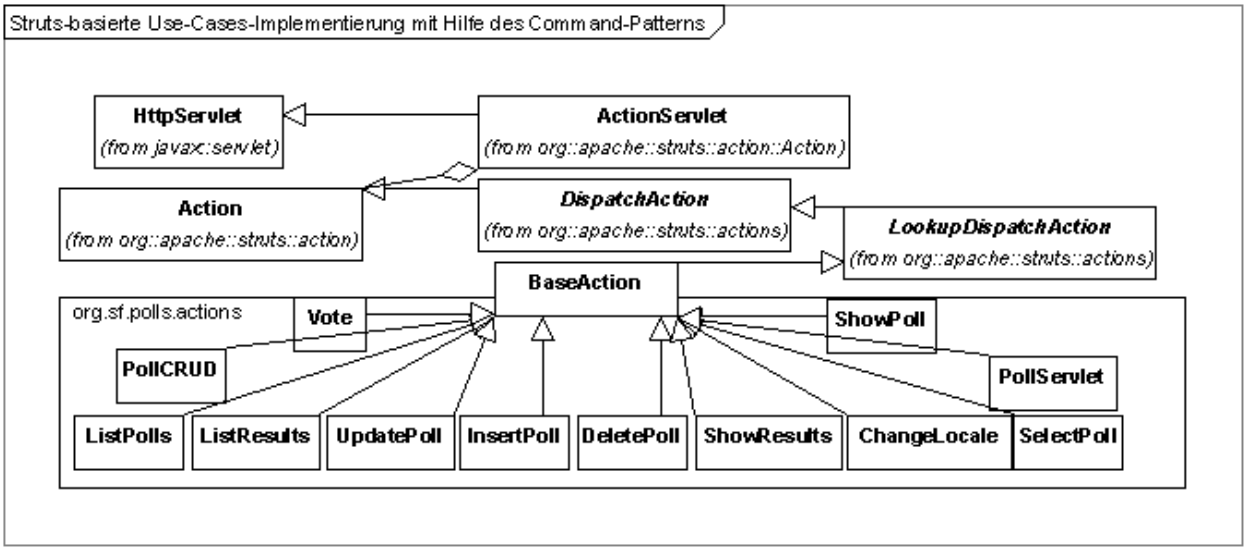

Abbildung 2-4. Struts-basierende Controller-Implementierung

Alle spezialisierten Controller des Pakets erben die Methode *execute()* von der Struts-Klasse *LookupDispatchAction* (indirekt über die *BaseAction*-Klasse). Innerhalb dieser Methode wird die logische Use-Case-Abarbeitung, sowie eine optionale Weiterleitung an den nächsten Controller durchgeführt. Die Controller überschreiben deshalb diese Methode mit ihrer eigenen Funktionalität. Beispielsweise extrahiert der Controller *ListPolls* über seine *execute()*-Methode alle persistenten Umfragen. Die generische Abarbeitung einer http-Anfrage (eines Use-Cases) wird in Abbildung 2-5 dargestellt.

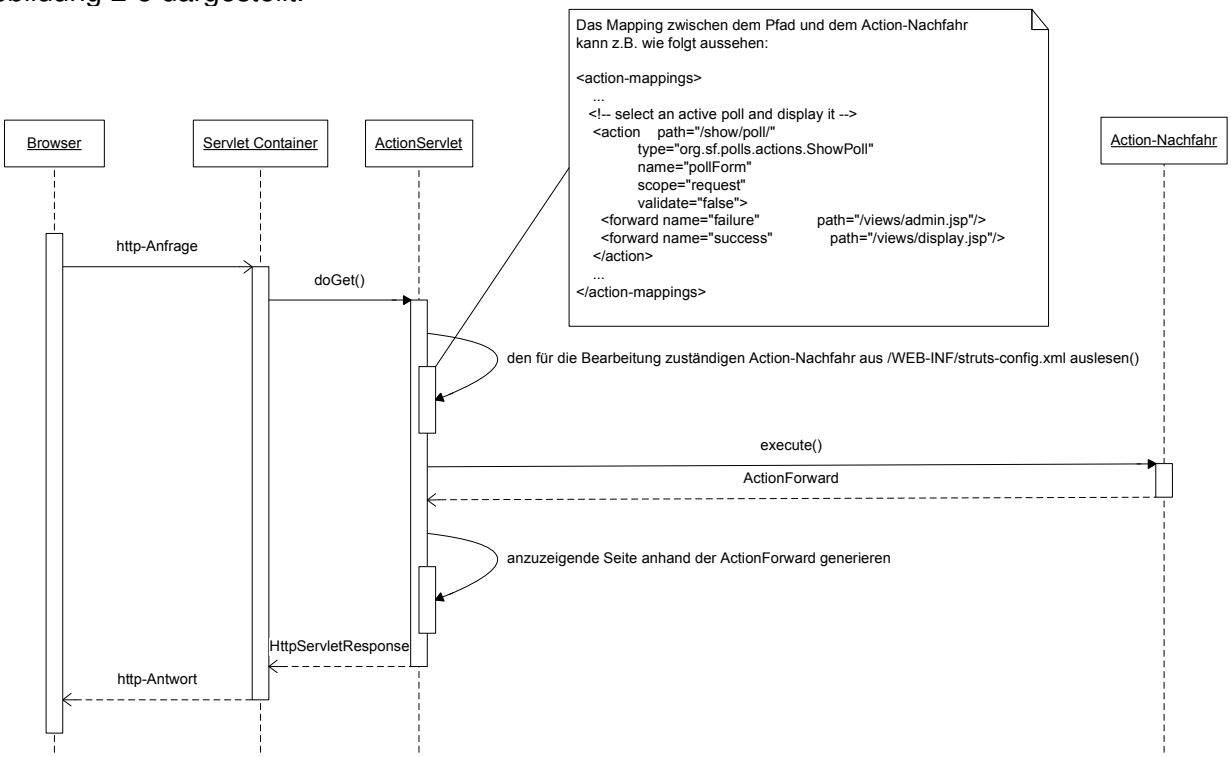

Abbildung 2-5. Generische Abarbeitung eines HTTP-Requests in Struts/jPolls

2.2.3. Paket org.sf.polls.charts

Das Paket org.sf.polls.charts bietet Klassen, welche statistische Auswertungen für die Umfragen implementieren und für eine Diagrammgenerierung verwendet werden können (Abbildung 2-6). Für das Erzeugen von Diagrammen wird die Library jfreechart-0.9.8.jar (siehe Abschnitt 3.2.2) verwendet. Die Klassen ResultsData, PollsData, MonthlyData sowie HourlyData nutzen die Methoden der Persistenzschnittstelle DAOInterface (siehe Abschnitt 2.2.1. bzw. die Abbildung 2-3) zur Extrahierung der statistischen Werte und füllen dann damit die jeweiligen DataSets aus der jFreeChart-Libriary. Die Abbildung 2-7 verdeutlicht diesen Ablauf anhand der Klasse ResultsData.

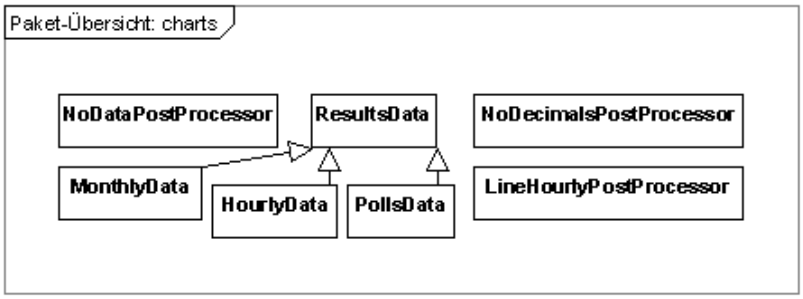

Abbildung 2-6. Aufbau des Packages org.sf.polls.charts

Die Klassen LineHourlyPostProcessor, NoDecimalsPostProcessor sowie NoDataPostProcessor werden für eine Nachbearbeitung der in generierten Diagrammen vorkommenden Werte eingesetzt. So generiert z.B. eine NoDecimalsPostProcessor-Instanz eine angepasste Fehlermeldung bei leeren Charts.

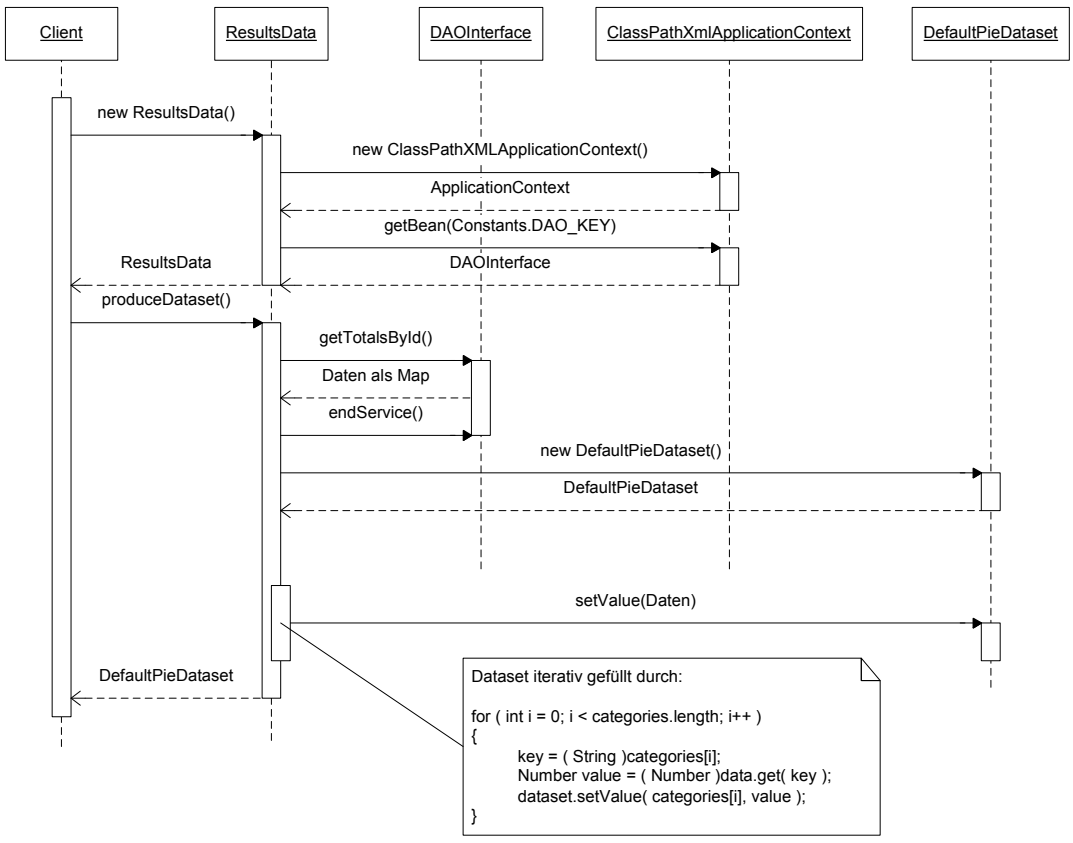

Abbildung 2-7. Erstellen eines Diagramms mit ResultsData

2.2.4. Paket org.sf.polls.storage

Das Package org.sf.polls.storage stellt die Methoden zur Nutzung von verschiedenen Persistenzschicht-Plugins (wie z.B. das Hibernate, siehe Abschnitt 3.2.3) und ermöglicht somit das Speichern und Laden von Daten aus der Logik-Schicht. Die Struktur des Pakets ist in Abbildung 2-8 dargestellt.
Die zentralen Klassen dabei sind das *HibernatePlugin* und das *CastorPlugin*. Beide erben von *SetupDAO*, die Klasse *HibernatePlugin* überschreibt jedoch die *init*-Methode und instantiiert darin die Klasse *SessionFactory* von Hibernate. Weiterhin bietet die *HibernatePlugin*-Klasse Methoden für das Öffnen und Schließen von Sessions, sowie für die Freigabe von Hibernate-Ressourcen. Sowohl *HibernatePlugin* als auch *CastorPlugin* sind jedoch als deprecated markiert, stattdessen sollen die Klassen *HibernateDAOImpl* bzw. *CastorDAOImpl* eingesetzt werden, welche das *DAOInterface* (siehe Abschnitt 2.2.1. und die Abbildung 2-3) direkt implementieren und dabei auf das Spring-Framework (siehe Abschnitt 3.2.5) zurückgreifen.

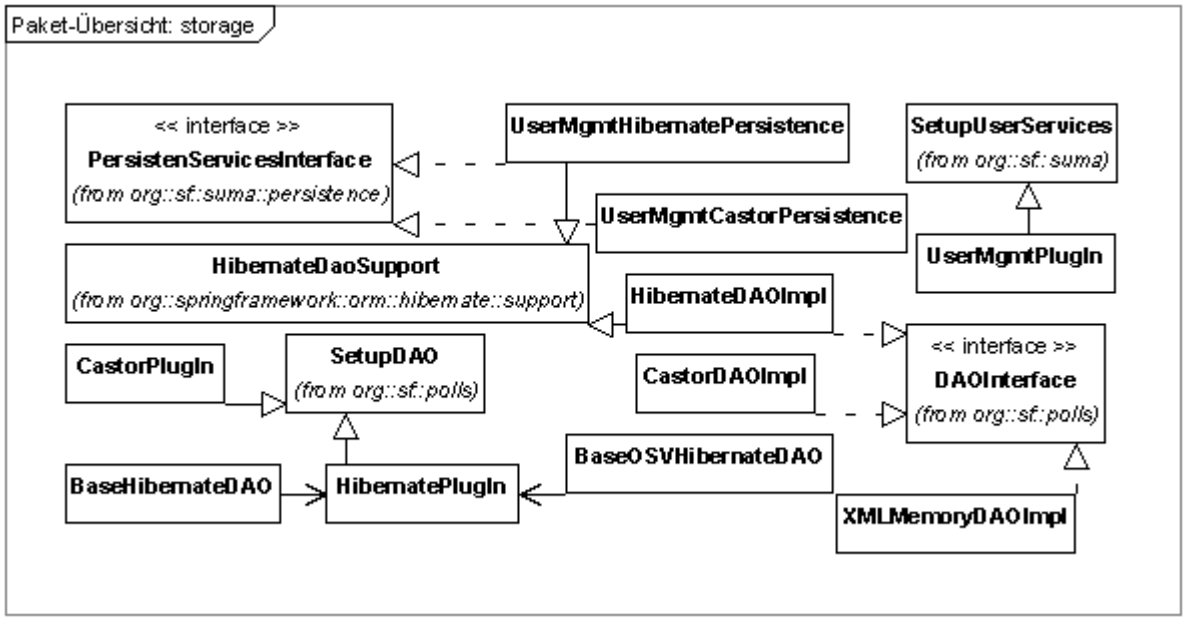

Abbildung 2-8. Package org.sf.polls.storage

2.2.5. Paket org.sf.polls.taglib

Das Package org.sf.polls.taglib bietet eine Sammlung von Klassen, welche drei jPolls-Tags für ein einfacheres Design der Präsentationsschicht definieren (Abbildung 2-9).

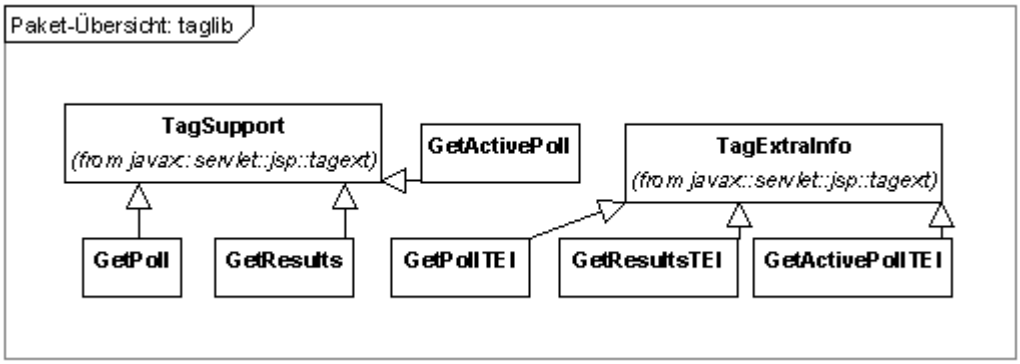

Abbildung 2-9. Package org.sf.polls.taglib

Die GetPoll- und GetResults-Klassen liefern dabei anhand der im Tag angegebenen Umfrage-Id eine Poll- bzw. eine Results-Instanz (siehe Abschnitt 2.3.1). Die Klasse GetActivePoll sucht

dagegen nach Poll-Instanzen anhand angegebener Kriterien. Alle drei Tag-Klassen benutzen jeweils das aktive DAOInterface für den Zugriff auf persistenten Daten.

2.2.6. Paket org.sf.polls.util

Das Package org.sf.polls.util beinhaltet drei Klassen (Abbildung 2-10), welche für zusätzliche Funktionalität sorgen. Die Klasse Utils beinhaltet Methoden zum Anlegen und Löschen von Verzeichnissen, der DateDecorator wandelt ein Datum in das gewünschte Format um und das KeyValuePair realisiert eine einfache Zuordnung zwischen zwei Objekten.

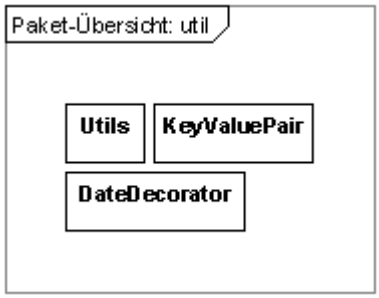

Abbildung 2-10. Package org.sf.polls.util

2.3. Systemfunktionalität

jPolls bietet Unterstützung bei der Verwaltung von Umfragen und deren Ergebnissen. Als wichtigste fachliche (Logik-Schicht, Abschnitt 2.1.2) Aktivitäten können dabei z.B. das Anzeigen von existierenden Umfragen, Anzeigen einer Umfrage/deren Ergebnisse, Erstellen/Ändern/Löschen einer Umfrage, sowie Stimmabgabe genannt werden (für eine Beschreibung der Implementierung der Aktivitäten siehe auch Abschnitt 2.2.2.).
Die Abbildung 2-11 verdeutlicht diese Use Cases.

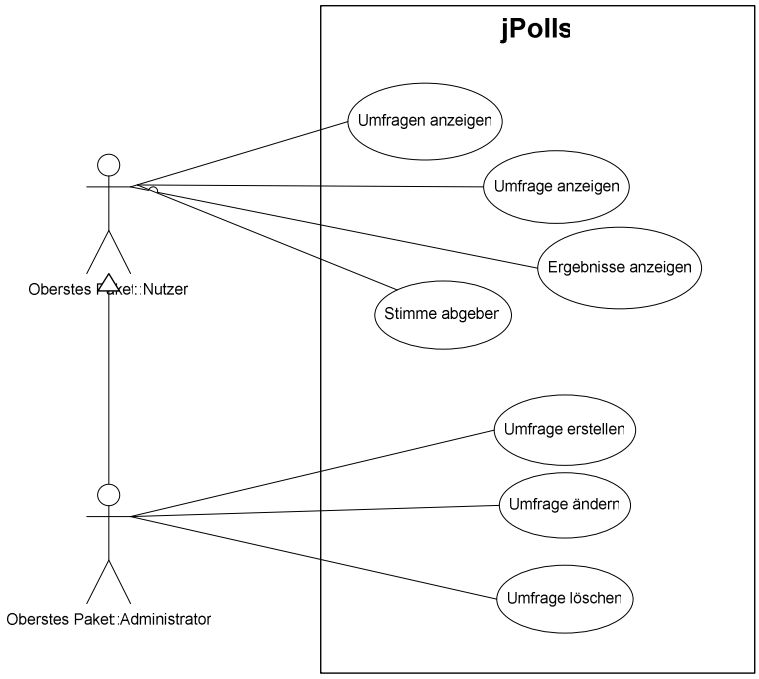

Abbildung 2-11. Von jPolls unterstützte Use Cases

3. Installation und Deployment

Das jPolls wird als ein war-Archiv ausgeliefert und kann entweder standalone oder im Rahmen eigener Anwendung deployed werden. Die Standalone-Installation geschieht durch einfaches kopieren des Archivs in das WebApp-Verzeichnis eines Webservers. Beim Nutzen des jPolls als

Framework für eigene Anwendung muss die Datei polls2.X.X.jar ins /WEB-INF/lib-Folder des WebServers kopiert werden.
Die wichtigsten Verzeichnisse des Archivs sind: /css (enthält Style-Sheets), /databases (SQL-Skripte), /skripts (JavaScript-Dateien), /src (Quellcode) sowie /view (JSP-Seiten) und /WEB-INF (Konfigurationsdateien und eingebundene Bibliotheken).

3.1. SQL-Dateien

Ordner /databases enthält SQL Tabelle Kreation Indexe für die verschiedenen geprüften Datenbanken.
Zu einer MaSQL-Distribution gehören die folgenden Tools:

- **SQL-Server**. Dies ist die Einzige, die MySQL antreibt und den Zugriff auf Datenbanken ermöglicht.
- **Clientprogramme für den Serverzugriff**. Ein interaktives Programm ermöglicht es uns, Anfragen direkt einzugeben und die Ergebnisse anzeigen zu lassen, und mehrere Administrations- und Dienstprogramme helfen uns, die Webseite zu betreiben. Mit einem der Dienstprogramme kann man den Server steuern, mit anderen kann man Daten im- und exportieren, Zugriffsrechte prüfen und anderes mehr.
- **Eine Clientbibliothek, damit wir eigene Programme schreiben können**. Wir können Clients in C schreiben, da auch die Bibliothek in C geschrieben ist, aber sie bietet zudem eine Grundlage zur Einbindung in andere Programmiersprachen

3.2. Verwendete Libriaries

3.2.1 Batik-Toolkit

Batik (ein Projekt der Apache Software Foundation) ist ein auf Java basierendes Toolkit, das Applikationen die Darstellung, das Erzeugen und die Manipulation von Scalable Vector Graphics (SVG) ermöglicht.
Um die Visualisierung des Projekts zu ermöglichen, wurde die folgenden Bibliotheken in das Verzeichnis polls/WEB-INF/lib integriert:
batik-awt-util.jar – AWT und Swing, enthält z.B. die Klasse mit GridBagConstraints-Konstanten, Hilfsklasse, Implementierung von JPanel mit der Verwendung von GridBagLayout, die Klasse für Managing von Message für Swing-Erweiterungen, die Klasse, die Erweiterungen zum java.awt-Package beinhaltet, usw
batik-dom.jar – stellt die Implementierung vom DOM Niveau 2 Hauptmodul bereit
batik-svggen.jar – stellt API auf dem höheren Niveau von AbstractGraphics2D für das Übersetzen von Java 2D-Primitivitäten in das SVG-Format bereit
batik-util.jar – stellt einige nützliche Klassen bereit, wie z.B. Klassen für die Realisierung der Arbeit mit XML, URI, CSS und DOM SVG1.2 usw
batik-xml.jar – enthält die Klassen und einen Interface für die Realisierung der Arbeit mit XML-Dokumenten

3.2.2. JFreeChart

JFreeChart – eine freie Klassen-Bibliothek für das Generieren von Diagrammen für Java, zur Verwendung in Applikationen, Applets und Servlets.
Die zugehörigen Bibliothek-Dateien aus polls/WEB-INF/lib:
jfreechart-0.9.8.jar – JfreeChart Runtime-Jar-Datei
jcommon-0.8.0.jar – The JCommon Runtime-Jar-Datei

3.2.3. Cewolf

Cewolf wird für die Integration von verschiedenen Diagrammen in Servlet/JSP-basierenden Web-Applikationen auf der Web-Seite verwendet. Alles ist mit Hilfe von XML beschrieben. Cewolf basiert auf JfreeChart.
Die zugehörige Bibliothek:
cewolf.jar – auf dem Servlet basierendes Framework für die Visualisierung von Diagrammen in den Rückmeldungs-Stream des Klienten

3.2.4. Castor

Castor ist ein Open Source Data-PersistenzFramework für Java. Es ist eine einfache Bindung zwischen Java-Objekten, XML-Dokumenten und relationalen Datenbanken. Castor bietet Java-to-XML-Bindung, Java-to-SQL-Stabilität, und viel mehr.
Die Zugehörige Bibliotheken:
castor-0.9.5.2.jar - enthält Source Code-Generator, DSML für Import/Export von LDAP-Verzeichnissen als XML, Java Data Objects, Class Mapping, Implementierung von Castor der spezifischen XML-Schema-Typen
castor-0.9.5.2-xml.jar – XML-Marshaller

3.2.5. Spring

Spring ist ein vielstufiges Java/J2EE-Applikation Framework, basierend auf dem Code, der in Expert One-on-One J2EE Design and Development von Rod Johnson veröffentlicht wurde.
cglib-2.0-rc2.jar - CGLIB Code Generation Library, Bibliothek der Generierung des Codes, wird bei der Hibernate-Bibliothek für die dynamische Generierung des Codes verwendet.
dom4j-full.jar - DOM-Bibliothek für Java, wird bei der Hibernate-Bibliothek für das Einlesen von XML-Configuration-Dateien verwendet.
jta.jar - Java Transaktion API
hibernate2.jar - Hibernate Objekt-relationale Speicherung, wird für Java-Objekte in den Datenbanken verwendet
odmg-3.0.jar – ODMG (Object Database Management Group) API Specifikation wird bei der Hibernate-Bibliothek für die Unterstützung von Standards für gespeicherte Objekte verwendet
spring-aop.jar - Core Spring AOP Schnittstelle, built on AOP Alliance AOP interoperability interfaces, basiert auf dem AOP Alliance, Schnittstelle AOP, der funktionalen Vereinbarkeit AOP.
spring-context.jar – Dieses Paket basiert auf dem Beans-Paket für das Hinzufügen von Message-Quellen und für Observer design pattern, und stellt für die Objekte der Applikationen die Möglichkeit bereit, Ressourcen mit der Verwendung von konsequenten API zu bekommen
spring-core.jar – stellt grundlegende Klassen für die Bearbeitung von Exception und das Auffinden von Version und andere Hilfsklassen bereit, die nicht für das bestimmte Framework spezifiziert sind
spring-dao.jar – Exception- Hierarchie, die die Behandlung von schwierigen Fehlern ermöglicht.
spring-orm.jar – Root-Package für das Integrieren von Spring's O/R Mapping-Klassen.
spring-web.jar – bereitstellt Web-spezifische Funktionalität von Daten-Bindung, inklusive Utility-Klasse für den einfachen Aufruf von Bindung und Validation.
springwebmvc.jar – standardmäßige Ausführung von Controller für das MVC-Framework, das Spring enthält.
spring.jar - enthält AOP Alliance-Schnittstellen.

3.2.6. Jakarta-Projekt

Jakarta Project bietet vielfaltigen Auswahl von open source Java-Losungen und ist ein Teil von Apache Software Foundation.
commons-collection.jar – Jakarta Commons Collections, wird bei der Hibernate-Bibliothek für die effiziente Behandlung von Datenstrukturen verwendet.
commons-logging.jar – diese Bibliothek ist Teil des Apache Axis Projektes, wird bei der Hibernate-Bibliothek für die Loggung verwendet.
commons-beanutils.jar - Bean Introspection Utilities-Komponent von Jakarta Commons. Das Unterprojekt bietet low-level-Service-Klassen, die das Getting und Setting in Java-Klassen ermöglichen, die design patterns outlined in JavaBeans Specification nennen.
commons-digester.jar - Digester-Packet ist für regel-basierte Processing von freien XML-Documenten vorausgesehen.
commons-fileupload.jar – ein Komponent für die Behandlung von HTML-Dateien.
commons-lang-2.0.jar – bereitstellt statische Utility-Methoden, die die Funktionalitäten von java.lang und anderen Standart-Klassen verstärken.

commons-validator.jar – Validator-Packet ß beretstellt die Validation für JavaBeans basierend auf den XML-Dateien.
jakarta-oro-2.0.8.jar - Jakarta-ORO Bibliothek enthält Pakete für die gesamte Bearbeitung von Java-Texten mit der Unterstützung von Servlets
taglibs-datetime.jar – Dies Projekt ist open-source repository für JSP custom tag libraries.
Andere benutzte Bibliotheken:
displaytag-1.0-2b.jar – Display-tag-Bibliothek ist ein open source suite für custom tags, die high-level Web-Presentation patterns bereitstellt, die mit MVC-Model arbeiten. Die Bibliothek bietet viele Funktionalitäten und ist leicht zu benutzen.
ehcache-0.6.jar - Easy Hibernate Cache, wird bei der Hibernate-Bibliothek für die Caching verwendet.
mysql-connector-java-3-0-10-stable-bin.jar - MySQL-Datenbanktreiber.
log4j-1.2.8.jar - Apache Logging Services, Log4j ist für die Bildung von log-Information des Workspace verwendbar.
xercesImpl.jar - Xerces XML-Parser
xmlParserAPIs.jar – XML-Parser APIs

4. Fazit

Das jPolls-Framework unterstützt den Softwareentwickler beim Erstellen, Modifizieren, Löschen von Umfragen und Erstellen von Diagrammen. Es bietet eine leicht an eigene Bedürfnisse anpassbare Architektur, die in unserem Projekt benutzt werden kann, weil der Kunde mit dem Editor die komplette Dienstleistung entwerfen und deren einzelnen Komponenten erstellen und bearbeiten können soll.

Das jPolls-Framework folgt dem Model-View-Controller-Konzept (MVC), die Struktur, d.h. Präsentationsschicht (View) und Geschäftslogik-Schicht (Controller) kann für die Struktur der Präsentationsschicht und der Geschäftslogik-Schicht des Editors hilfreich sein, in der Umsetzung unserer Struts-Applikation, auf die sich auch die Geschäftslogikschicht des Editors stützt. Die Persistenzschicht unseres MVC muss von uns jedoch anders implementiert werden, da die zu verarbeitende Datenmenge bei uns eher geringer ausfällt als in einem kompletten Umfragesystem.

5. Glossar

SVG - Scalable Vector Graphics (deutsch Skalierbare Vektorgrafiken) ist ein Standard zur Beschreibung zweidimensionaler Vektorgrafiken in der XML-Syntax.
DOM - Document Object Model, Programmierschnittstelle für den Zugriff auf HTML- und XML Dokumente.
URI - Uniform Resource Identifier (engl. „einheitlicher Bezeichner für Ressourcen“) ist eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physikalischen Ressource dient.
CSS Cascading Style Sheets ist eine deklarative Stylesheet-Sprache für strukturierte Dokumente.
AbstractGraphics2D - die Klasse des Package *org.apache.batik.ext.awt.g2d*
Batik ist ein toolkit für Anwendungen, die Bilder in skallierbare Vektor-Graphik (SVG) Formate für verschiedenen Zwecken, wie die Betrachtung, Generierung oder Manipulation verwenden wollen.
Marshaller - die Umwandlung von Anfragen, einschließlich Parameter, in ein standardisiertes Format, das für die Übergabe im Internet verwendbar ist.