Eine eGovernment-Architektur mit GIS-Unterstützung als Basis nachhaltiger Anwendungsentwicklung

Michael Gengenbach

Referat P5
Bayerisches Staatsministerium für Ernährung, Landwirtschaft und Forsten
Ludwigstr. 2
80539 München
Michael.Gengenbach@StMELF.Bayern.DE

Abstract: Überblick einer Architektur für Java-Webanwendungen mit Fokus auf der Präsentation, Erfassung und Verarbeitung von Daten einschließlich Geodaten. Die objektorientierte 3-Schichten-Architektur wurde vom Bayerischen Landwirtschaftsministerium nach der Evaluierung verschiedener Frameworks definiert und wird derzeit auch für große Projekte erfolgreich eingesetzt. Sie basiert auf der *Java Persistence API*, *Spring*, *Apache Wicket* und *OpenLayers*.

1 Problemstellung und Zielsetzung

Für die Abwicklung der Agrarförderung werden Anwendungen zur Präsentation, Erfassung und Verarbeitung von Antragsdaten benötigt (z.B. Betriebs- und Feldstücksdaten), wobei seit einigen Jahren auch Geodaten aller beantragten Flächen erfasst werden müssen. Diese Geodaten sind zwar einerseits von großer Bedeutung, stellen aber andererseits nur eine Facette der Anwendungen dar, denn letztlich ist ein Polygon nur ein zusätzliches Attribut eines Feldstücks neben vielen anderen.

Damit sich die Verarbeitung von Geodaten nahtlos in alle Schichten der Anwendungsentwicklung integriert, hat sich das Staatsministerium entschlossen, für die zukünftige Entwicklung von Anwendungen selbst eine möglichst umfassende und erweiterbare Architektur zu erarbeiten, die auf der Basis existierender offener Standards alle bekannten Anforderungen an die Anwendungsentwicklung abdecken kann, einschließlich der Verarbeitung von Geodaten. Weitere Anforderungen an die Architektur waren

- eine gute Performance auch bei mehr als 1000 gleichzeitigen Nutzern
- revisionssichere Speicherung aller Änderungen an Daten
- Kapselung der hohen technischen Komplexität von Java, damit sich Anwendungsentwickler stärker auf fachliche Anforderungen konzentrieren können.

2 Die eGovernment-Architektur

Als Programmiersprache kommt Java zum Einsatz, das sich als *General Purpose Language* mit sehr guter Unterstützung für nahezu alle Spezialgebiete und Endgeräte ganz besonders für eine möglichst flexible und zukunftsträchtige Architektur eignet. Gewählt wurde eine klassische 3-Schichten-Architektur für Webanwendungen, allerdings nicht auf Basis der *Java Enterprise Edition* (JEE) mit einem Applikationsserver, sondern aufgrund der hohen Dynamik und Komplexität dieser Technologie auf Basis des Servlet-Containers *Tomcat*.

Frühere Erfahrungen haben gezeigt, dass allein die Festlegung auf eine Programmiersprache bzw. sogar bestimmte Bibliotheken nicht ausreicht, um mit einer großen und heterogenen Entwicklungsmannschaft einheitliche Anwendungen zu erhalten. Deshalb umfasst die entwickelte Architektur darüber hinaus:

- Vorgaben, wie die Bibliotheken einzusetzen sind,
- selbst entwickelte Klassen, welche den Anwendungsentwicklern noch fehlende, aber anwendungsübergreifend benötigte Funktionalität zur Verfügung stellt,
- eine Umgebung mit Basiskomponenten für die einheitliche Bereitstellung administrativer Funktionen, z.B. *Single Sign On* oder eine Kompetenzverwaltung.

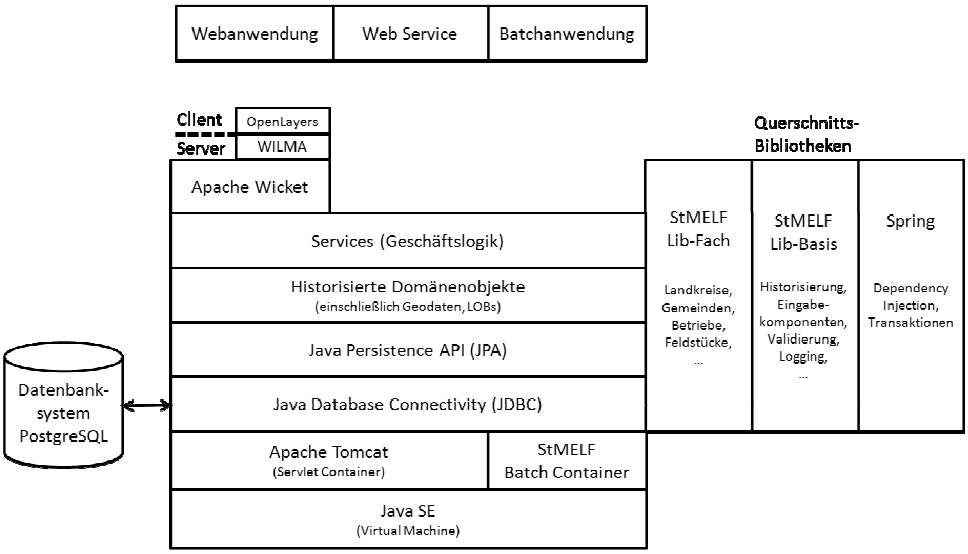

Abbildung 1: Schematische Darstellung der eGovernment-Architektur

2.1 Die Persistenzschicht

Die Architektur ist objektorientiert und verwendet fachliche Domänenobjekte (z.B. Betrieb, Feldstück, Nutzung). Seit vielen Jahren ist es üblich, die Persistenz fachlicher Objekte mittels *objektrelationalem Mapping* (ORM) umzusetzen. Im Java-Bereich hat sich hierfür inzwischen die *Java Persistence API* gemäß des *Java Specification Request* JSR 220 durchgesetzt, die als Interface von verschiedenen Frameworks implementiert wird. Nach einer Evaluierung verschiedener Frameworks hat sich das Staatsministerium hier für *Hibernate* [BK06] entschieden.

Im Bereich der Agrarförderung gibt es allerdings zwei Anforderungen, die sich mit JPA alleine nicht umsetzen lassen:

- die Verwendung von Geometrien gemäß dem *OGC Simple Feature Standard* als Datentyp für Attribute
- eine Historisierung aller Datenänderungen zum Zwecke der Revisionssicherheit

Die erste Anforderung konnte über die Hibernate-Erweiterung „*Hibernate Spatial*“ so umgesetzt werden, dass sich Geodaten in Form von Geometrieobjekten der *Java Topology Suite* [JTS] als @Embedded-Attribute nahtlos in JPA-Entities einfügen. Es ist somit möglich, in *Entities* auch Geometrien als Datentyp für Attribute zu verwenden.

Bei der Anforderung der (bitemporalen) Historisierung [Sn00] geht es darum, alle Änderungen an Daten langfristig nachvollziehen zu können. Hierfür reicht eine einzige Zeitachse (also die fachliche Gültigkeit eines Objekts) nicht aus, sondern man muss in einer zweiten Zeitachse speichern, wann Änderungen in das System übernommen wurden. Dies wurde mit einer auf JPA basierenden, selbst erstellten API umgesetzt.

Intensiv wird außerdem die *Bean Validation* gemäß JSR 303 verwendet, mit der es möglich ist, viele Plausibilitäten deklarativ in den Entitäten anzugeben.

2.2 Die Serviceschicht

Ein Grundprinzip der Schichtenarchitektur ist eine strikte Aufgabentrennung. So werden Datenbankzugriffe in die Persistenzschicht ausgelagert und die Verarbeitung der geladenen Fachdomänenobjekte wird in einer eigenen Service-Schicht angesiedelt. Diese enthält auch wichtige Querschnittsfunktionen, wie etwa die Transaktionssteuerung. Hier wurde entschieden, das Spring-Framework einzusetzen und zwar gezielt für die *Dependency Injection*, sowie eine deklarative Transaktionssteuerung.

2.3 Die Benutzeroberfläche

Nachdem bei der Architektur insbesondere Webanwendungen im Fokus standen, die neben Batchanwendungen den Großteil der Anwendungen am Staatsministerium ausmachen, musste auch hierfür ein geeignetes Framework gefunden werden. Durch die

zusätzliche Anforderung der Präsentation und Erfassung von Geodaten war dies eine besonders schwierige Herausforderung, für die aber eine gute Lösung gefunden wurde:

Bei der Evaluierung schnitt das Komponenten-Framework *Apache Wicket* [AWI] hinsichtlich der geforderten Bedürfnisse am besten ab und bietet einen guten Kompromiss, um dynamische Webanwendungen umsetzen zu können, ohne bei jedem Fachanwendungsentwickler große JavaScript-Kenntnisse vorauszusetzen.

Es konnte erreicht werden, dass die Plausibilitäten der *Bean Validation* aus der Persistenzschicht von den Masken der Oberfläche automatisch verwendet werden, dem Benutzer gegebenenfalls entsprechende Fehlermeldungen angezeigt werden und somit keine mehrfache Implementierung von Plausibilitäten benötigt wird.

Zum Einsatz kommt auch eine GIS-Komponente, welche die Bayerische Vermessungsverwaltung entwickelt hat und bei der das JavaScript-Framework *OpenLayers* [OL] als Wicket-Komponente gekapselt wurde, so dass sie ohne Browser-Plugin auskommt.

3 Fazit

Ziel einer Architektur bzw. eines Frameworks ist es, die Arbeit der Anwendungsentwickler zu vereinfachen, indem sie auf vorgefertigte Baupläne bzw. Programmbausteine zurückgreifen können. Mit der vorliegenden eGovernment-Architektur konnten bereits komplexe Anwendungen erfolgreich umgesetzt werden. Ein Beispiel ist die Erfassung des Mehrfachantrags im Frühjahr 2013 mit bis zu 6 Mio. Requests pro Tag. Besonders erfolgreich war dabei auch die angestrebte nahtlose Integration von Geodaten in allen Schichten.

Für eine nachhaltige Anwendungsentwicklung wird ein Technologiestack benötigt, der alle bestehenden Anforderungen abdecken kann und der gleichzeitig auch flexibel an neue Bedürfnisse anpassbar ist. Dies ist mit der beschriebenen Architektur gegeben.

Literaturverzeichnis

[AWI] Apache Wicket Framework Homepage, http://wicket.apache.org/

[BK06] Bauer, C.; King, G.: Java Persistence with Hibernate, Manning, 2006

[JTS] Java Topology Suite Homepage, http://www.vividsolutions.com/jts/

[OL] OpenLayers Framework Homepage, http://openlayers.org/

[Sn00] Snodgrass, R.: Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann Publishers, 2000.