

Praktikum DBAE / WI

Moderne Frameworks

Pascal Reuss, M.Sc.

Raum C033 Spl.
Email: reusspa@uni-hildesheim.de

Moderne Frameworks

Was ist ein Framework?

- Ein Programmiergerüst, das den Entwickler bei der Anwendungsentwicklung unterstützt
- Enthält Entwurfsmuster, Bibliotheken und Laufzeitumgebungen
- Gibt die Anwendungsarchitektur / Designstruktur vor
- Wiederverwendbarkeit

Moderne Frameworks

Verschiedene Arten von Frameworks (keine strikte Trennung)

- Application Frameworks (horizontal)
- Domain Framework (vertical)
- Class Framework
- Komponenten Frameworks
- Coordination Frameworks
- Tests Frameworks
- Webframeworks

Moderne Frameworks

Vorteile

- Leistungsfähigkeit
schneller, einfacher, effizienter
- Sicherheit
Security-Implementierungen, Lücken werden meist schnell behoben
- Kosten
Viele Frameworks sind kostenlos verfügbar
- Unterstützung
meist große Community, Support-Teams und Dokumentationen

Moderne Frameworks

Nachteile

- Sprache wird nicht gelernt
Es wird die Benutzung des Frameworks gelernt, weniger die Sprache, die dahinter steht
- Einschränkung
Frameworks arbeiten nach bestimmten Mustern, an diese ist der Entwickler gebunden
- Code ist öffentlich
Der Code eines Frameworks und die verwendeten Mechanismen sind öffentlich, dies kann ausgenutzt werden

Enterprise Java Beans

- Komponenten Framework
- Standardisierte Komponenten
- Vereinfachen Entwicklung verteilter Softwaresysteme
- Unterstützt Sicherheit, Persistenz, Transaktionenm etc.
- Remote oder Lokaler Zugriff
- Entity Bean, Session Bean, Message Driven Bean, Web Services
- Aktuelle EJB 3.1

Enterprise Java Beans

Vorteile:

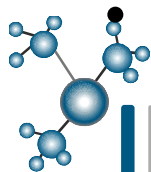
- Komponenten vollständig spezifiziert
- Integration in J2EE
- Skalierbarkeit
- Zugriff auf Ressourcenverwaltungssysteme

Nachteile:

- Komplexe Spezifikation
- Sehr große Dokumentation
- Zeitaufwand für Entwicklung erhöht sich
- Code wird komplexer
- Design kann zu kompliziert werden
- Spezifikationsänderungen möglich

Spring

- Open Source Applikationsframework
- (Plain-Old-Java-Objects) POJOs werden als Spring Beans verwaltet
- Kerntechnologie: Dependency Injection / Inversion of Control
Konfiguration von Abhängigkeiten zwischen Beans außerhalb des Quellcodes
- Aspektorientierte Programmierung
Modularisierung von Aspekten, Implementierung an einer Stelle, Anwendung an vielen Stellen (Logging, Sicherheit, Transaktionen)
- Konfiguration mittels XML und/oder Annotations



• Template für APIs

Spring

Vorteile:

- Kann mit vielen anderen Technologien kombiniert werden
- Einfacher als EJB
- Definierte Sicherheitsmechanismen
- Hohe Ausfallsicherheit
- Hohe Skalierbarkeit

Nachteile:

- Einarbeitungszeit
- Durch Konfigurierbarkeit weniger Prüfungen zur Compile-Zeit

Struts

- Open Source Framework für Java Webanwendungen
- Verarbeitet HTTP Anfragen in standardisierten Prozessen
- Nutzt Servlets, JavaBeans, XML, Apache Commons, etc
- Model-View-Controller Entwurfsmuster
- Unterstützt View und Controller

Struts

Vorteile:

- Bessere Wartbarkeit
- Leichtere Modifizierbarkeit
- Höhere Sicherheit durch Rollenkonzept
- Höhere Benutzerfreundlichkeit durch Exception Handling

Nachteile:

- Einarbeitungszeit
- Durch Konfigurierbarkeit weniger Prüfungen zur Compile-Zeit

Java Server Faces

- Framework zur Entwicklung grafischer Oberflächen für Webanwendungen
- Basiert auf Servlets und JSPs
- Klare Trennung zwischen Oberfläche und Anwendungslogik
- Facelets – Vorlagen für Webseiten
- Template Engine erzeugt die Webseite aus Facelets und füllt sie mit Daten aus Java Klassen

Java Server Faces

Vorteile

- Viele Bibliotheken
- Gute Trennung von Präsentation und Logik
- Kombinierbar mit anderen Technologien

Nachteile

- Hoher Ressourcenverbrauch
- Einarbeitungszeit
- Schwierig zu debuggen

Hibernate

- Open Source Framework für Java Anwendungen
- Object-Relational-Mapping
- Ermöglicht Objekte in Datenbanken zu speichern und aus Datensätzen Objekte zu erzeugen
- Kompatibel zu vielen verschiedenen Datenbanken
- SQL-Statements können vom Framework generiert werden
- Annotations, Validator, Suchfunktionen (mit Lucene), etc.

Hibernate

Vorteile

- Gute Datenbankunterstützung
- Keine Voraussetzungen an Laufzeitumgebung
- Sehr flexibel
- Viele Zusatzpakete
- Sehr effiziente SQL Statements
- Große und aktive Community

Nachteile

- Hoher Aufwand bei komplexen Anwendungen
- Nicht immer alle datenbankspezifischen Optimierungen nutzbar
- Mögliche Performance-Einbußen
- Synchronisationsprobleme bei Zugriff mit und ohne Hibernate