Eine Abstraktion zur Implementierung von Internet-Portalen

Vereinfachung der Entwicklung durch ein Framework

Ingo Melzer

Abteilung Angewandte Informationsverarbeitung

AGENDA

① Motivation
② Eigenschaften von Portalen
③ Abstraktion von Portalen
④ Abstraktion für die Ausgabe
⑤ Laden von Plug-Ins
⑥ Vom Framework zum Portal
⑦ Anwendung des Portal-Frameworks $\rightsquigarrow$ SLC
⑧ Zusammenfassung

MOTIVATION

- Informationsflut belastet User
- Bedürfnis nach Lösung: Portal
- Abstrakt gesehen sind fast alle Portale gleich
  - Framework für Portale
  - Schnelle Entwicklung eigener Lösungen
  - Flexibilität durch modularen Aufbau
- Idee aus Vorlesungsbeispiel von Andreas Borchert entstanden

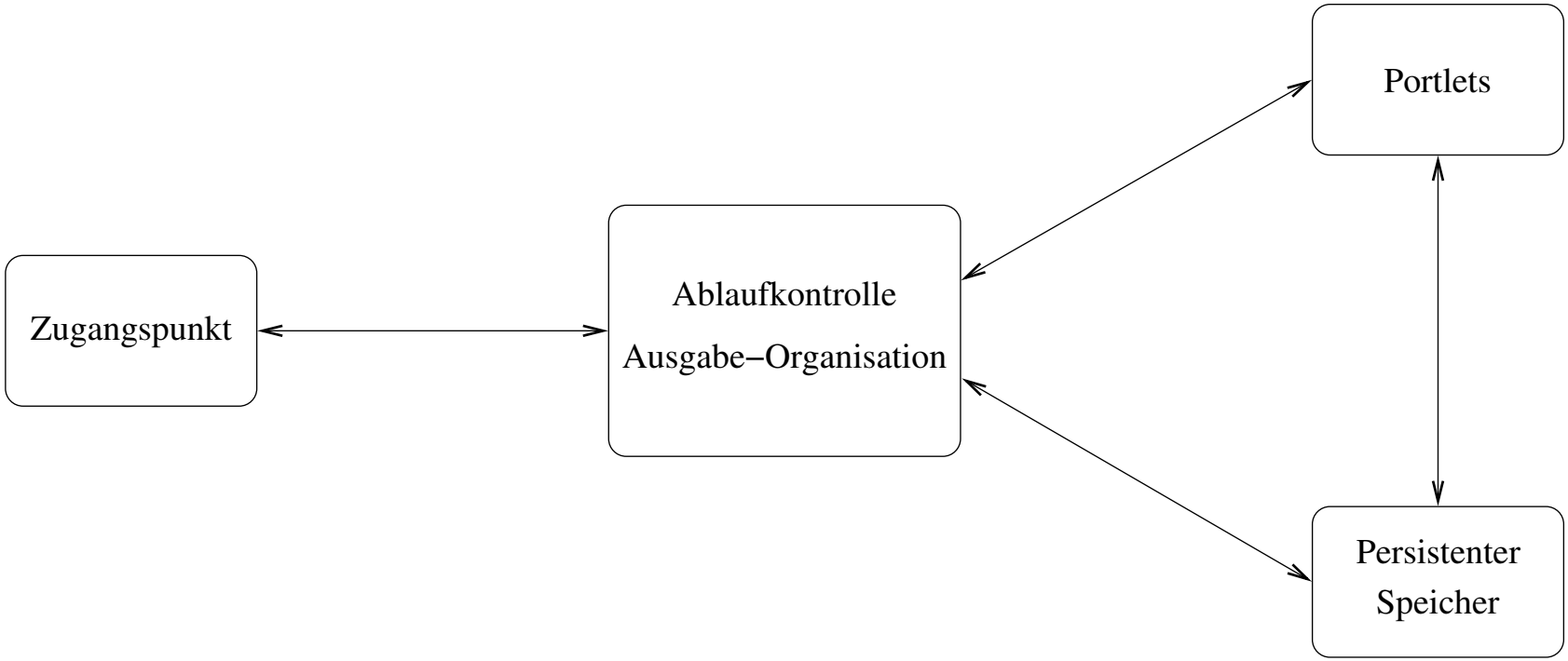
Eigenschaften von Portalen

- Personalisierung inklusive
  - personalisierter Daten mit
  - personalisierter Navigation in einem
  - personalisierten Format
- Sicherheit: Zugriff für Benutzer auf ihre (und nur ihre) Daten
- Navigation: „Einfacher“ Zugriff auf verschiedene Teile eines Portals
- Aufgabenunterstützung: Nur Darstellung gerade wichtiger Daten
- Datenhaltung: Persistente und effektive Datenverwaltung
- Einheitliches und konsistentes GUI: Alle Teile mit gleichartigem, selbsterklärendem Aussehen

Meine Definition eines Portals

Ein Portal ist ein System, das einem Benutzer einen einfachen, sicheren und personalisierten Zugriff auf die Informationen, die er gerade benötigt, gewährt.

Einfaches, abstraktes Skelett eines Portals

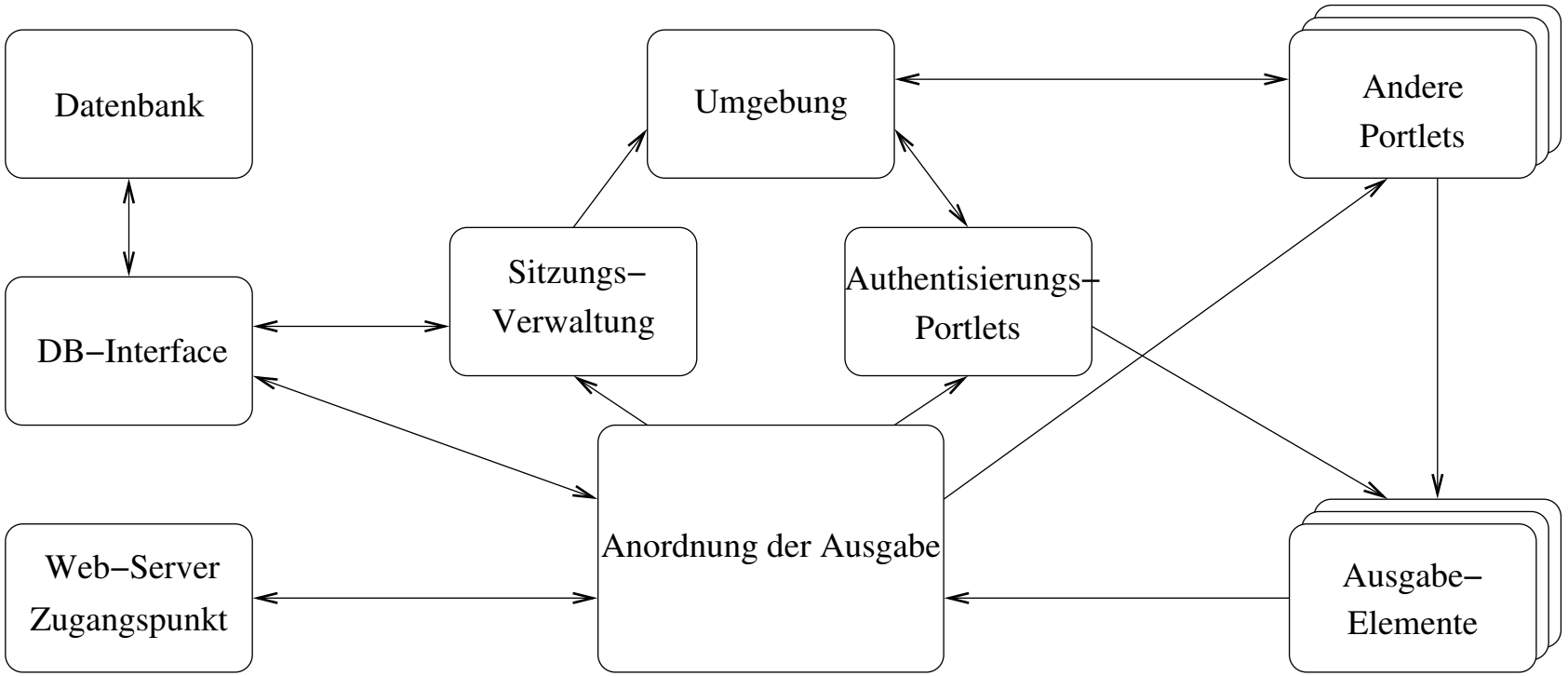


Der zentrale Teil ist verantwortlich für

- Ausgabe-Generierung/-Organisation
- Kontrollfluss bzw. Ablaufkontrolle

Notwendig: Persistenter Speicher

Anatomie eines Portals



- Umgebung um Zugriffsmöglichkeit auf die Datenbank erweitert
- Gleiche Ausgabe-Elemente $\rightsquigarrow$ einheitliches Erscheinungsbild

Abstraktion für die Ausgabe

Wünsche:

- Erlaube Änderungen bis zur Ausgabeerzeugung
- Generiere Ausgabetext so spät wie möglich
- Verwende eine Schnittstelle zur Ausgabeerzeugung
  $\rightsquigarrow$ erlaubt verschiedene Ausgabesprachen und erleichtert Änderungen
- Ausgabe-Elemente müssen als Container agieren können

Ergo:

$\rightsquigarrow$ Trennung von Inhalt und Darstellung

$\rightsquigarrow$ Unabhängigkeit vom Ausgabemedium (Web, WAP, PDA, ...)

$\rightsquigarrow$ Einheitliches Erscheinungsbild

$\rightsquigarrow$ Intern: Ausgabe als Baum von Ausgabe-Elementen;
beim Traversieren wird Ausgabetext erzeugt

PORTLETS ALS PLUG-INS

- Portlets bei der Entwicklung des Portals oft unbekannt
  $\rightsquigarrow$ Hinzufügen muss ohne Neubau des Portals möglich sein
- Fehlerhafte Portlets dürfen System nicht beeinträchtigen

LÖSUNGSANSATZ:

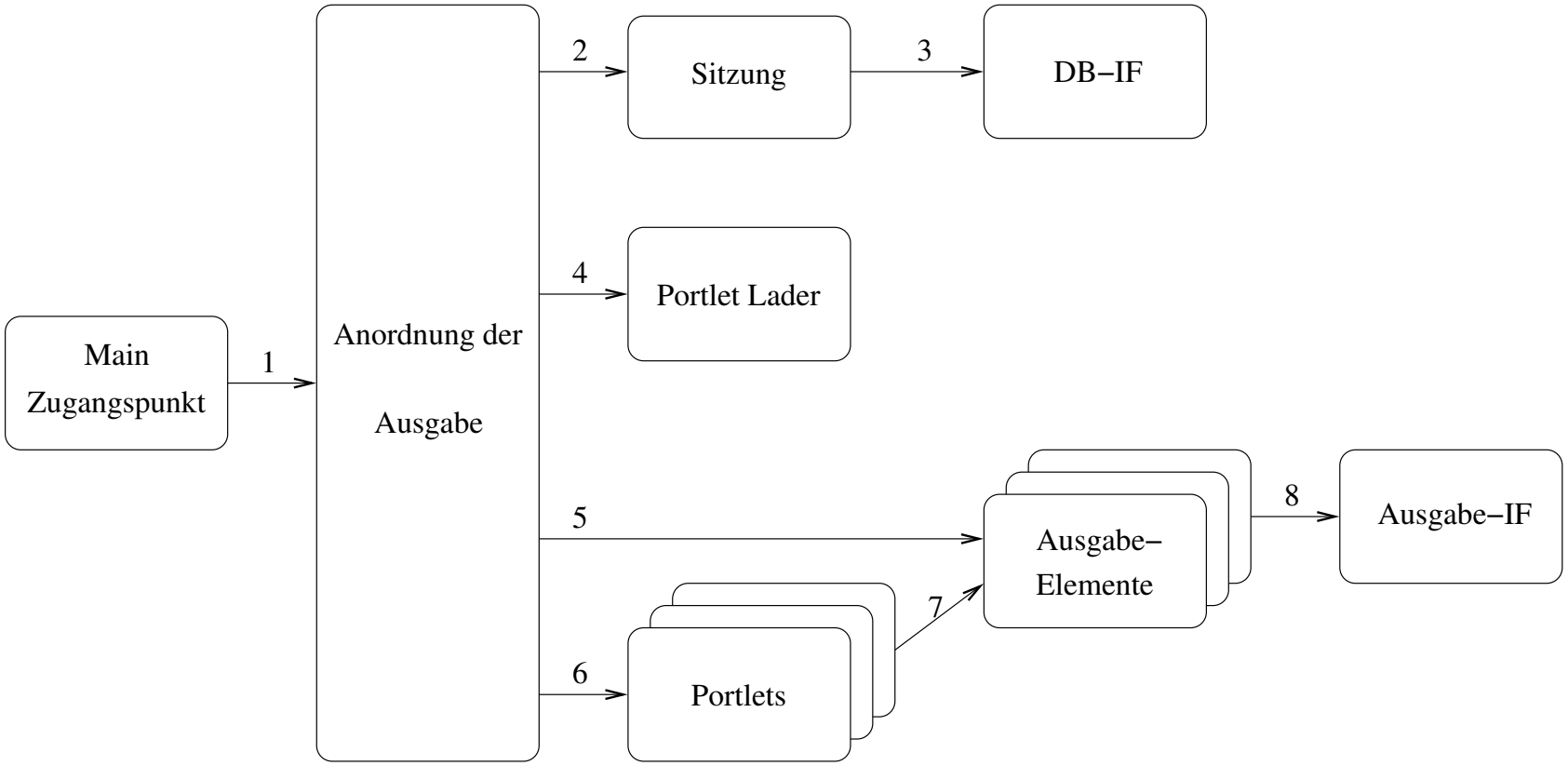
- Durchsuchen von möglichen Quellen nach Portlets
- (Dynamisches) Laden von Kandidaten zur Laufzeit
- Portlets müssen Erweiterung der Portlet Basis-Klasse sein
- Ausnahmebehandlung für Portlets

Portlets als Plug-Ins

`Portal::Portlets::scan`

```
 1  sub scan {
 2      my $dir = new IO::Dir $_[0];
 3      die "Unable to scan $_[0]: $!" unless defined $dir;
 4      my $filename; my %portlets = ();
 5      while (defined($filename = $dir->read)) {
 6          next unless $filename =~ /^([A-Z].*)\.pm$/;
 7          my ($name, $portlet) = ($1, "");
 8          my $module = "Portal::Portlet::$name";
 9          eval qq{
10              use $module;
11              \$portlet = new $module;
12              die "Wrong Interface"
13                  unless (\$portlet->isa(\'Portal::Portlet::Base\'));
14          };
15          if ($@) { warn "$module ignored: $@"; next; }
16          $portlets{$name} = $portlet;
17      }
18      return \%portlets;
19  }
```

Ablauf beim Aufruf eines Portals



Nummerierung: Reihenfolge der Aufrufe

Modularer Aufbau

Module:

- Server: Kommunikation mit Klient
- Fast beliebige Datenbank
- Gekapselte Portlets für eigentliche Funktionalität
- Ausgabe-Elemente: Trennung von Darstellung und Inhalt
- Inferface für Ausgabe: Unterstützung verschiedener Ausgabe-Sprachen und Medien
- Interface für Eingabe: Anpassung an Server-Protokolle

VOM FRAMEWORK ZUM PORTAL

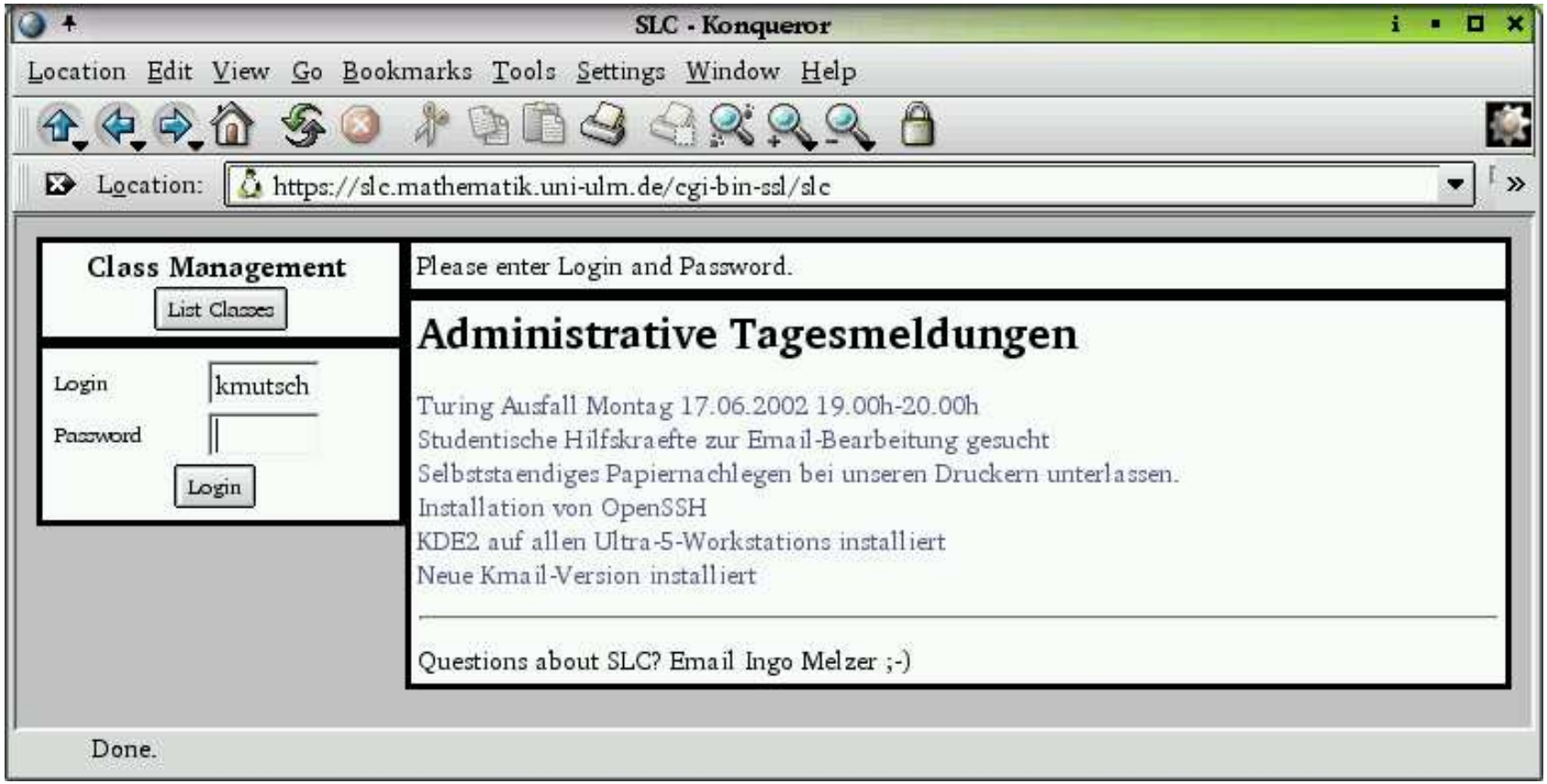
Generelle Funktionalität von Portalen (Sitzungsverwaltung, Portlet-Lader, . . .) bereits Teil des Frameworks

- Wahl der Basis: (Web-)Server, Datenbank, OS
- Authentisierungsportlets anpassen (NIS, Smartcard, . . .)
- Anpassung der Ausgabe-Elemente an das gewünschten Format
- Implementierung der zusätzlichen Funktionalität in neuen Portlets, z. B. Einkaufswagen für Shop

SLC

- SLC: Student, Lecture, and Class Management
  System zur Verwaltung von Vorlesungen,
  Computerzugangsberechtigungen, Druckerkontingenten, ...
- Demonstration des Einsatzes des Frameworks
- Portal SLC baut auf dem Framework auf
- Wird an dieser Fakultät mit täglich über 2 000 Zugriffen eingesetzt
- In Perl geschrieben
  - erlaubt dynamisches Laden von Modulen
  - läuft auf fast allen Plattformen
  - objekt-orientiert
  - beliebige Datenstrukturen
  - auch in Zukunft von der Abteilung wartbar

STARTSEITE VON SLC



Hier aktiv: Vorlesungs-, Login- und News-Portlet

ZUSAMMENFASSUNG

- Analyse der grundsätzlichen Struktur von Portalen $\rightsquigarrow$ Abstraktion
- Framework für Portale, Implementierung von Funktionen, die alle Portale brauchen (Sitzungsverwaltung, Zugriff auf Daten(banken))
- Erarbeiten eines allgemeinen Konzepts für Plug-Ins und Anwendung auf Portlets, die beim Start des Portals unbekannt sein können
- Entwicklung einer Abstraktion für GUI und Implementierung daraus resultierender Ausgabe-Elemente
- Ausbau des Frameworks zu SLC, produktiver Einsatz in der Fakultät
- Open Source Lösung $\rightsquigarrow$ konfigurierbar, da Quellen vorhanden