FH Bielefeld
University of
Applied Sciences

Campus Minden

Webbasierte Anwendungen
Java Server Faces I
SS 2015

Dozentin: Prof. Dr.-Ing. Grit Behrens
mailto: grit.behrens@fh-bielefeld.de

Studiengang Informatik Fachbereich Technik

Lehrinhaltsübersicht zu WBA

1. Einführung in Webbasierte Anwendungen
2. Web-Grundlagen, Struktur von Webseiten: CSS, XML, XHTML, HTML5
3. Clientseitige Implementierungstechnologien: Javascript, DOM, Ajax, (Java-Applet)
4. Serverseitige Implementierungstechnologien: JSP, Java-Servlet, Webserver
5. **Webframework: Java Server Faces**

Heute in der Veranstaltung

1. **Model-View-Controler Pattern (aus Sicht des Softwareengineering)**
2. **Java Server Faces Teil 1**
   - **Motivation aus Entwicklersicht**
   - **Marktwirtschaftliche Motivation zu JSF**
   - **Einführungsbeispiel Tic Tac Toe**
   - **Hinweise zur Konfiguration der Entwicklungsumgebung**

Heute in der Veranstaltung

1. **Model-View-Controler Pattern (aus Sicht des Softwareengineering)**
2. **Java Server Faces Teil 1**
   - **Motivation aus Entwicklersicht**
   - **Marktwirtschaftliche Motivation zu JSF**
   - **Einführungsbeispiel Tic Tac Toe**
   - **Hinweise zur Konfiguration der Entwicklungsumgebung**

Grundidee von Design-Pattern

Damit nicht alle Klassen eng miteinander gekoppelt sind, gibt es Ansätze:

- die Aufgaben einer Klasse von der Verwaltung der Klassen, die Informationen dieser Klasse benötigen, zu trennen
- die Erzeugung von Objekten möglichst flexibel zu gestalten
- Interfaces zur Trennung von Implementierung und angebotenen Methoden einzusetzen

- Hierzu werden so genannte Design-Pattern eingesetzt, die für einen bestimmten Aufgabentyp eine flexible Lösung vorschlagen
- oft zitiert: E. Gamma, R. Helm, R. Johnson, J. Vlissides, Entwurfsmuster, Addison-Wesley, 2004 (Gang of Four [GoF]-Buch, hier neuere Auflage)

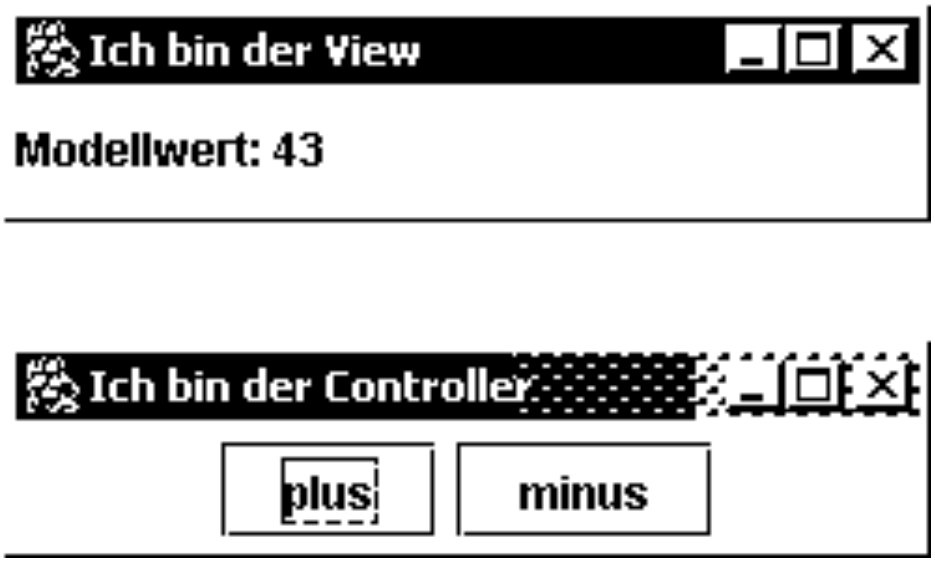
Model-View-Controller

- Typisch für ***graphische Oberflächen*** ist, dass es Objekte zur ***Eingabe*** gibt, die zur ***Bearbeitung der eigentlichen Inhaltsklasse*** führen, die dann eventuell zu ***Änderung der Anzeige*** führen
- Die Aufteilung in die drei genannten Aufgaben führt zum

  Model-View-Controller (MVC)- Ansatz
- MVC wurde ***zuerst in Smalltalk Ende der 80'er*** des vorigen Jahrhunderts eingesetzt:
  - **Model:** Zustandsinformation der Komponente (Inhaltsklasse)
  - **View:** Beobachter des Zustands, um diesen darzustellen; es kann viele Views geben
  - **Controller:** Legt das Verhalten der Komponente auf Benutzereingaben fest
- ***Idee: Controller steuert Änderungen des Modells, Modell teilt allen Views mit, dass eine Änderung aufgetreten ist***
- Hinweis: Spezielle Form des Beobachter-Musters (Observer)

MVC – Beispiel: Anforderungen

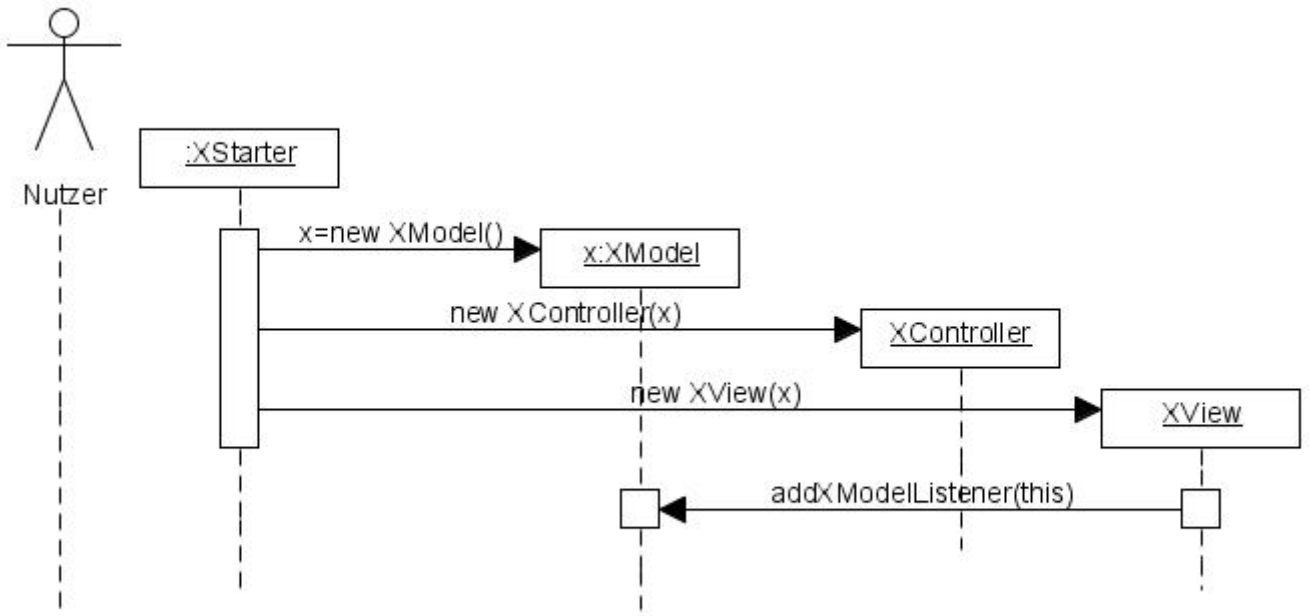
- Realisierung eines MVC-Pattern
  - Modell besteht nur aus einem ganzzahligem Wert
  - View ist ein Fenster, dass diesen ganzzahligen Wert ausgibt
  - Controller erlaubt über zwei Buttons „plus“ und „minus“ eine Werteänderung um jeweils „+1“ oder „-1“ des Modells



MVC – Beispiel: Nutzungsphasen (1/2)

zwei Nutzungsphasen des MVC-Pattern:

1. Phase: Erzeugen und Verbinden der MVC-Objekte
   - allgemein:
     1. zuerst Modell erzeugen,
     2. danach beliebig viele Controller und Views mit Referenz auf Modell
   - im Beispiel: Klasse `XStarter` konstruiert die Objekte der Klassen `XModel`, `XController` und `XView`.



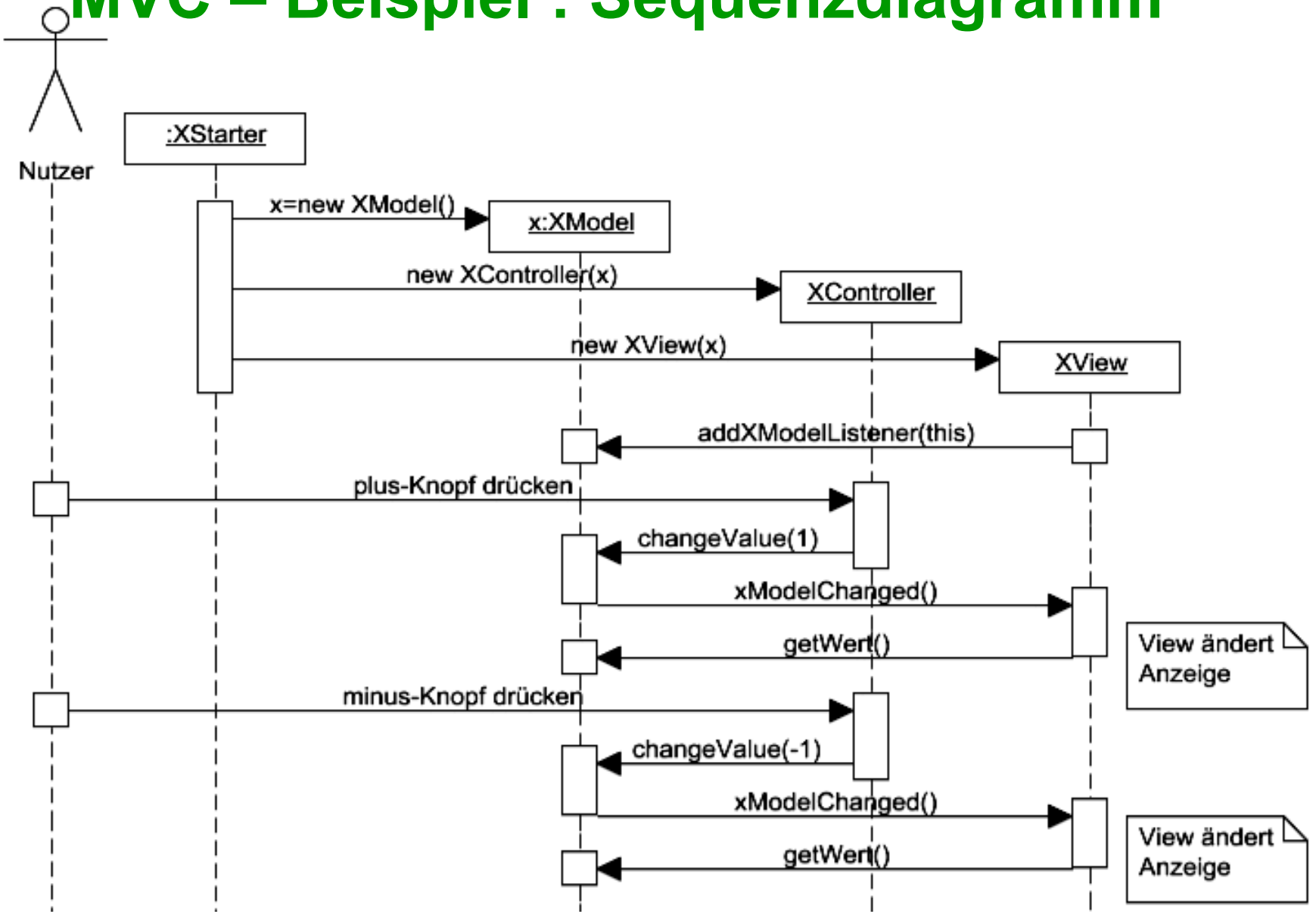
MVC – Beispiel: Nutzungsphasen (2/2)

zwei Nutzungsphasen des MVC-Patterns:

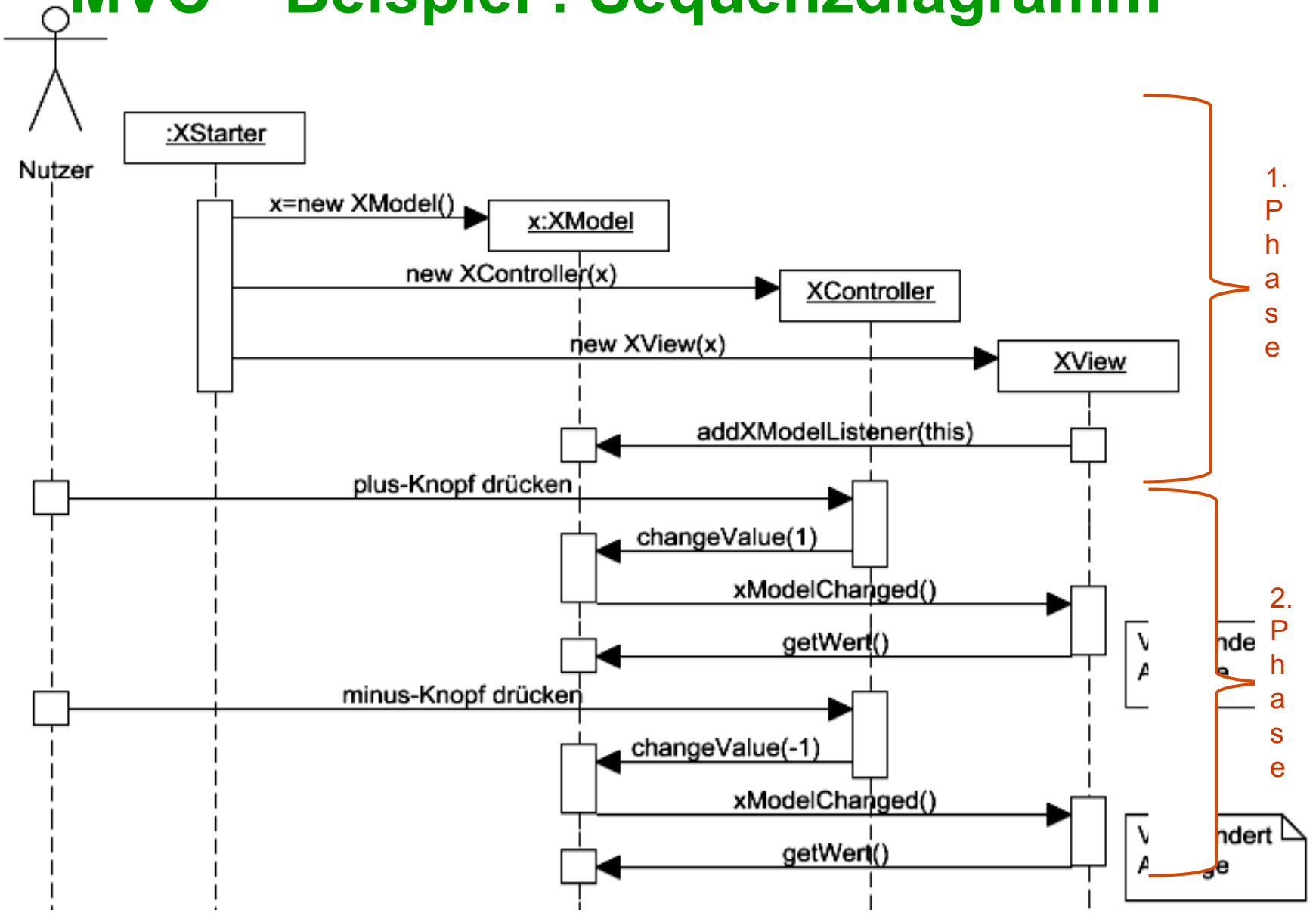
2. Phase: Nutzung der MVC-Struktur
   - allgemein: alle Änderungen gehen von Controller-Objekten aus
     1. Controller schickt Änderungsaufforderungen an das Modell
     2. Modell schickt Änderungen an die referenzierten Views mit zwei Varianten:
        a) Modell teilt in Benachrichtigung alle Werteänderungen mit
        b) View holt sich nach Benachrichtigung die geänderten Werte selbst ab (sinnvoll bei großen Datenmengen)

   - im Beispiel:
     - nach „Plus-Knopf“ drücken veranlasst `XController` eine Werteänderung mit `changeValue()` im `XModel`. Das Model informiert `XView` über eine Werteänderung mit `xModelChanged()`. Die View ändert Ihre Anzeige nach Aufruf von `getWert()`.
     - „Minus-Knopf“ ähnlich

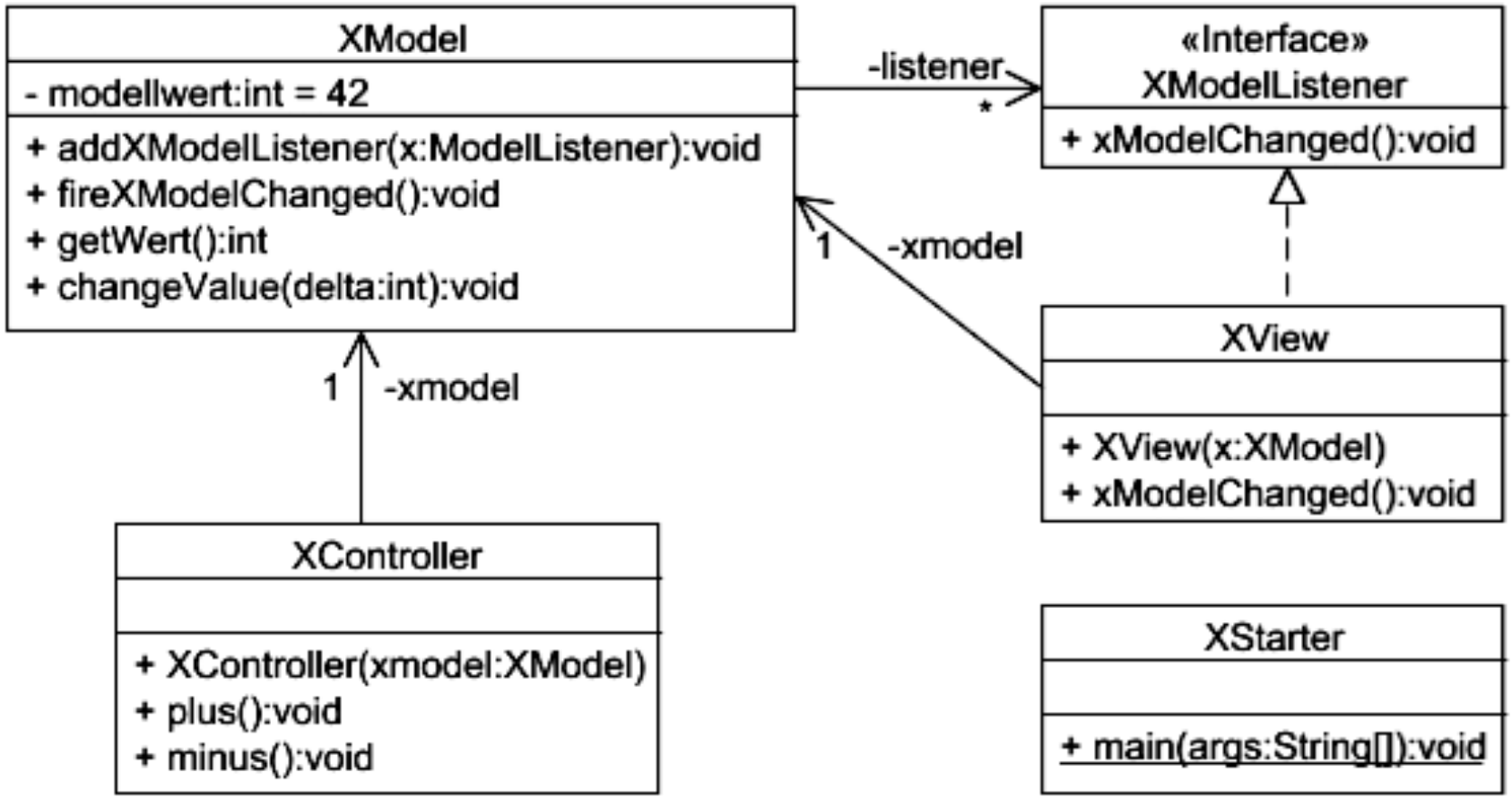
MVC – Beispiel : Sequenzdiagramm



MVC – Beispiel : Sequenzdiagramm



MVC –Beispiel :Klassendiagramm



Einheitliche Schnittstelle für alle Views zur Anmeldung beim Modell und zur Benachrichtigung bei Modelländerungen mit `XModellListener`.

```
public interface XModelListener {
          public void xModelChanged();
}
```

MVC – Beispiel : Realisierung des Modells

```
import java.util.*;
public class XModel{
   private ArrayList<XModelListener>listener=
                                   new ArrayList<XModelListener> ();
   private int modellwert=42;

   //Verwaltung der Listener des Modells
   public void addXModelListener(XModelListener x){
    listener.add(x);
   }

   private void fireXModelChanged(){
        for(XModelListener x:listener)
                x.xModelChanged();
   }

   //Auslesen der Modellinhalte
   public int getWert(){
        return modellwert;
   }

   //Veränderung des Modells
   public void changeValue(int delta){
        modellwert+=delta;
        fireXModelChanged();
   }
}
```

Variable `listener` verwaltet die Views

Variable `modellwert` hält Modell

Methode zum Anmelden der Views beim Modell

Methode zur Änderung der Views durch Controller

Getter für Änderung des Modells

Methode zum Ändern des Modells durch Controller

MVC – Beispiel : Realisierung der View

```
import javax.swing.*;

public class XView extends JFrame implements XModelListener{
   private XModel xmodel;
   private JLabel jlabel= new JLabel("Modellwert:     ");
   public XView(XModel x){
        super("Ich bin der View");
        xmodel=x;
        xmodel.addXModelListener(this);
        //Rest Swing für Anzeige
        getContentPane().add(jlabel);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(250,60);
        setLocation(0,0);
        setVisible(true);
   }
   public void xModelChanged() {
        jlabel.setText("Modellwert: "+xmodel.getWert());
   }
}
```

Konstruktor der View

Exemplarvariable `xmodel` hält Info über zugehöriges Modell

Hinzufügen der View zum Lauschen auf Werteänderungen beim Modell

Implemetieren der Interfacemethode `xModelChanged` aus `XModelListener`

Werteänderung im Model

MVC – Beispiel : Realisierung des Controllers

```
import java.awt.FlowLayout;
import java.awt.event.*;
import javax.swing.*;

public class XController extends JFrame{
   private XModel xmodel;

   public XController(XModel x){
        super("Ich bin der Controller");
        xmodel = x;
        getContentPane().setLayout(new FlowLayout());
        JButton plus = new JButton("plus");
        getContentPane().add(plus);
        plus.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        xmodel.changeValue(1);
                }});
        JButton minus = new JButton("minus");
        getContentPane().add(minus);
        minus.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                        xmodel.changeValue(-1);
                }});
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(250,60); setLocation(0,90);´setVisible(true);
   }
}
```

Modell `XModel` wird im Konstruktor des Controllers bekannt gemacht

Exemplarvariable `xmodel` hält Model

Änderung des Wertes im Modell um „+1“

Änderung des Wertes im Modell um „-1“

MVC – Beispiel : Realisierung Interface und Initialisierungsklasse

```
public interface XModelListener {
   public void xModelChanged();
   /* Anmerkung: alternativ kann man auch geänderte
      Werte als Parameter übertragen */
}

public class XStarter {
   public static void main(String[] args) {
        XModel x= new XModel();
        new XView(x);
        new XController(x);
   }
}
```

Interfaceklasse
XModelListener

Initialisierungsklasse
XStarter

Heute in der Veranstaltung

1. Model-View-Controler Pattern (aus Sicht des Softwareengineering)
2. Java Server Faces Teil 1
   - Motivation aus Entwicklersicht
   - Marktwirtschaftliche Motivation zu JSF
   - Einführungsbeispiel Tic Tac Toe
   - Hinweise zur Konfiguration der Entwicklungsumgebung

Motivation für Java Webframeworks

1) JSP- Technologie erlaubte hochkomplexe Mischungen aus HTML, Java-Code-Scriptlets.
2) HTML-Generierung in Servlets
3) Compilierung von JSP-Code erst zur Laufzeit im Applikationserver (Fehler treten erst dort auf, nicht in der Entwicklung)

-> schlechte Wartbarkeit, Erweiterbarkeit und hohe Fehleranfälligkeit

-> Lösung der Probleme: WEB-Frameworks

- Trennung der Layoutbeschreibung in JSP von Funktionalität (Anwendungslogik) im Model2 – Pattern
- „**Model2-Pattern“:** Übertragung des MVC-Ansatzes in die Webentwicklung mit genauerer Ausprägung von Model(Beans), View (z.B. JSP ‘s) und Controller (Servlet ‘s)

Was sind Java Server Faces I/II

Definition: ***Java Server Faces*** *sind ein Framework für die Entwicklung von Benutzerschnittstellen als Teil einer Java-Web-Anwendung im unternehmenskritischen Umfeld.*

Aus der Spezifikation von JSF:

„... to significantly ease the burdon of writing and maintaining applications that run on a Java application server and render their UI's back to a target client."

Was sind Java Server Faces II/II

1. definieren ein ***Komponentenmodell*** für
   - Benutzerschnittstellenelemente einer Web-Anwendung (wie Swing für lokale Oberflächen)
   - Wiederverwendbare Komponenten in Webanwendungen als Entwicklungsziel
   - **-> Komponenten-Framework für GUI-Elemente**

2) **Hierarchisches Modell**: ***Container-Komponenten*** können andere Komponenten enthalten, Eigenschaften können vererbt werden.

3) **MVC –basiert:**

- **Model** können sein: EJB (Enterprise Java Beans) oder beliebige Java-Objekte,…
- **View** können sein: HTML, JSP
- **Controller** können sein Komponenten und Handler in Java

Java Server Faces und Zukunftssicherheit?

- Java-Community-Process (JCP, www.jcp.org) führt Standardisierung durch
- Entwicklungs- und Standardisierungsziel: Bündelung vielfältiger und weit auseinanderlaufender Ansätze zur Entwicklung von Webanwendungen unter der Technologie JAVA
- JavaServer-Faces in Java-Specification-Requests (z.B. JSR 127) definiert
- Beteiligt u.a. Apache, BEA, Borland, Fujitsu, HP, IBM, ILOG, IONA,Macromedia, Novell, Oracle, Siemens, Sun

Historie:

- J2EE (Java2Enterprise Edition)1.4 11/2003 (JSP, Servlets, Enterprise Java Beans aber noch ohne JSF)
- Final Release JSF 1.0 3/2004
- Java EE (umbenannt aus J2EE) 5 Final Release 5/2006 (enthält JSF 1.2)

-> ***Alle Hersteller von JavaEE-Application Servern müssen JSF unterstützen.***

Spezifikationen und Implementierungen

Standards aus dem Java Community Process:

- JSF 1.1 Spec (JSR -127) 2004
- JSF 1.2 Spec (JSR -252) 2006
- JSF 2.0 Spec (JSR -314) 2009
- JSF 2.2 Spec (JSR -344) Mai 2013

Kostenlose Implementierungen zu JSF 1.1:

- JSF Referenzimplementierung 1.1 und 1.2 von SUN
- MyFaces 1.1.3 (http://myfaces.apache.org/) von der Apache Software Foundation auf Apache Tomcat

JSF 2.0 – Referenzimplementierung:

- Mojarra auf Glassfish 3.0 Server (vollständiger JavaEE 6 Application Sever) – in SDK Eclipse verfügbar

JSF hat seit 2006 – 2014 viel Schwung aufgenommen und ist neben Struts zu einer der wichtigsten WebFrameworks geworden.

JSF 2.0 die wichtigsten Neuerungen gegenüber JSF 1.2

- ***Facelets in den Standard*** aufgenommen zur Deklaration der Seiten (im Unterschied zu JSP ‘s)
- Neu: ***Kompositkomponenten*** (Erstellung eigener zusammengesetzter Komponenten ohne Java-Programmierung)
- Erweiterte Möglichkeiten zur Validierung: u.a. durch

  Integration von ***Bean-Validierung***
- Integration von ***Ajax***
- Neu: ***System-Events*** bieten Reaktionsmöglichkeit auf Ereignisse im Lebenszyklus
- Steigerung der Effizienz von JSF-Implementierungen
  - z.B. zwischen Anfragen werden nur noch geänderte Bereiche des Komponentenbaumes gespeichert

Heute in der Veranstaltung

1. Java Server Faces Teil 1

- Motivation aus Entwicklersicht
- **Marktwirtschaftliche Motivation zu JSF**
- Einführungsbeispiel Tic Tac Toe
- Hinweise zur Konfiguration der Entwicklungsumgebung

Marktwirtschaftliche Motivation für JSF

- Großer strategischer Vorteil von JSF:
  - JSF ist ***Teil der Java EE – Spezifikation, seit Version 5***
  - Alle Webserver, die den Standard erfüllen, bringen JSF mit

Problem:

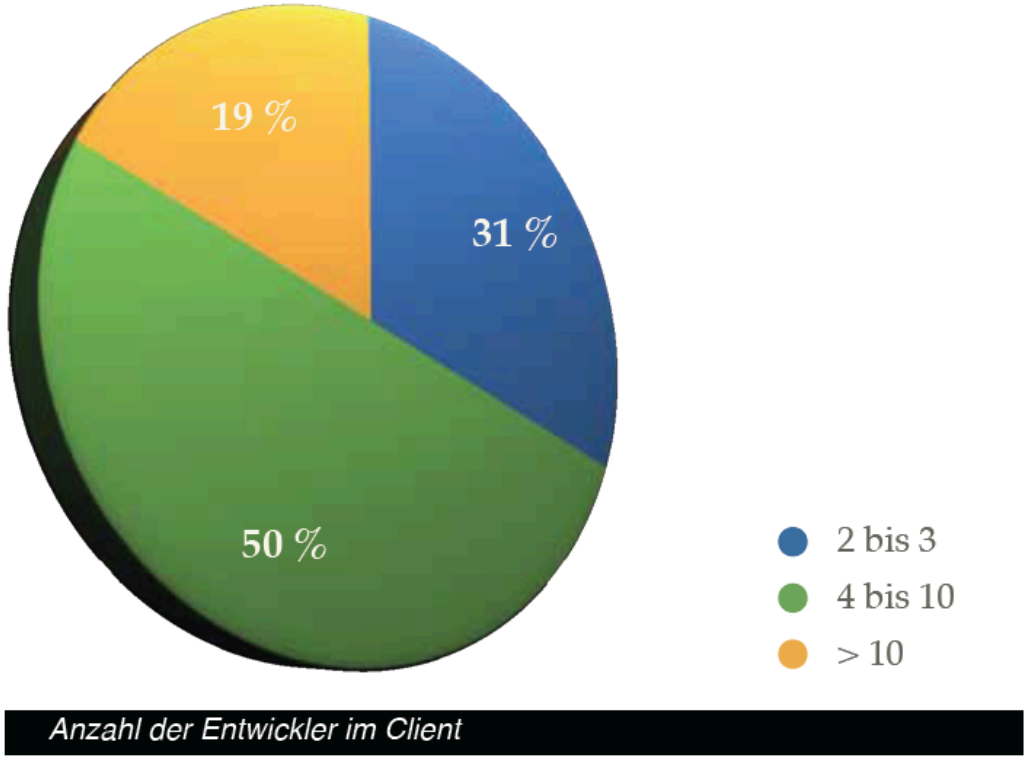
- Man kann einer Webapplikation von außen in der Regel erst mal nicht ansehen, ob sie mit JSF geschrieben ist
- Automatische statistische Analyse zum Einsatz von WebFrameworks schwer möglich

Lösung:

- ***Marktstudien mit Befragungen*** müssen durchgeführt werden.
  - z.B. http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/ ?
- ***Jobscreening*** möglich (ein paar Beispiele vom Juni .2014folgen)

Marktstudie mit Befragung (Ausschnitte)

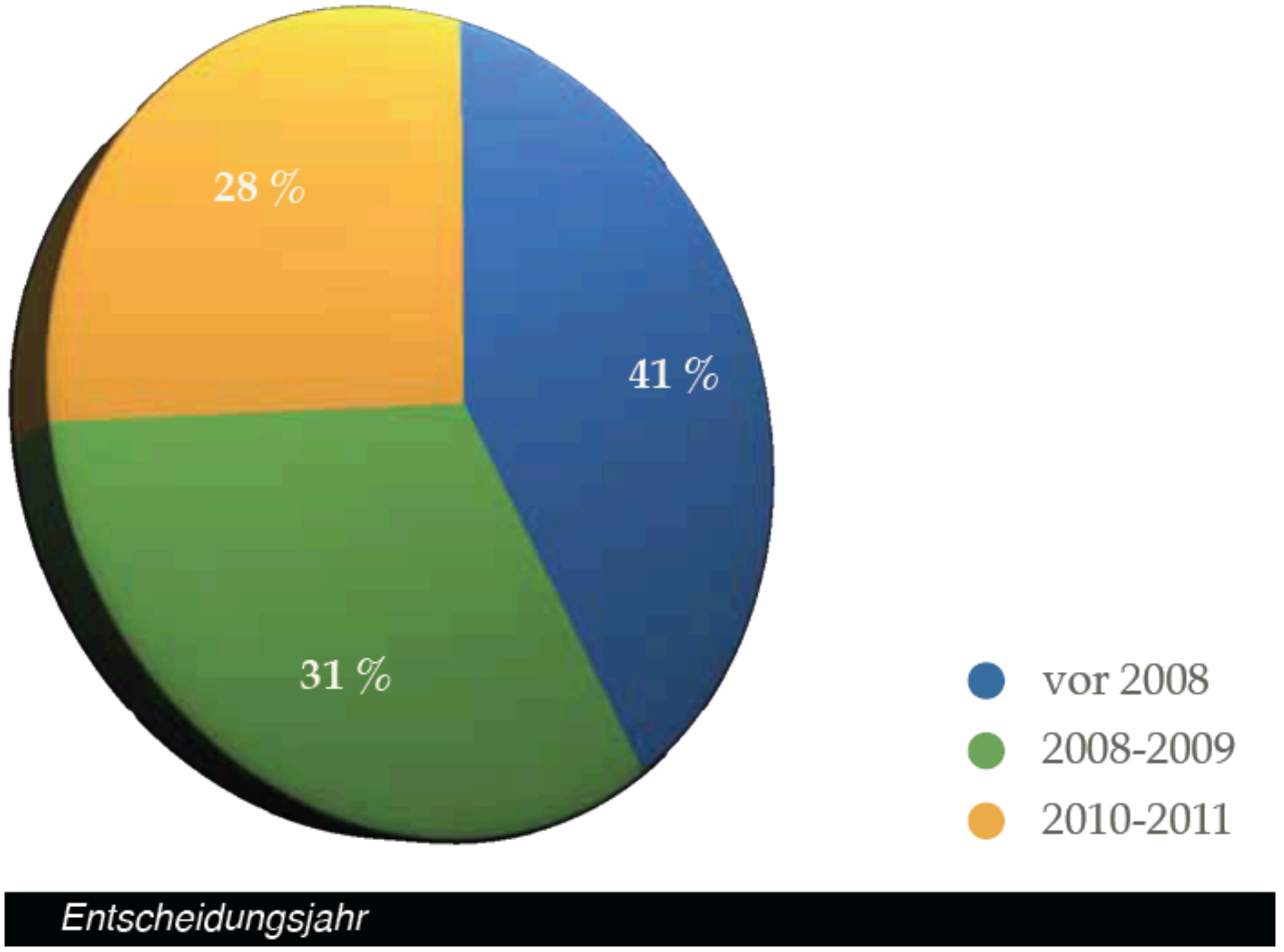
- Beratungsfirma macht jährlich eine ***Online-Befragung*** über: ***Hintergründe zur Auswahl von WebFrameworks***
- *Auswertung von 2011 kostenlos:*
  - *52 Teilnehmer* als Vertreter für Webentwicklerteams von insgesamt ca. *250 Entwicklern*



Quelle: http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/

Marktstudie mit Befragung (Ausschnitte)

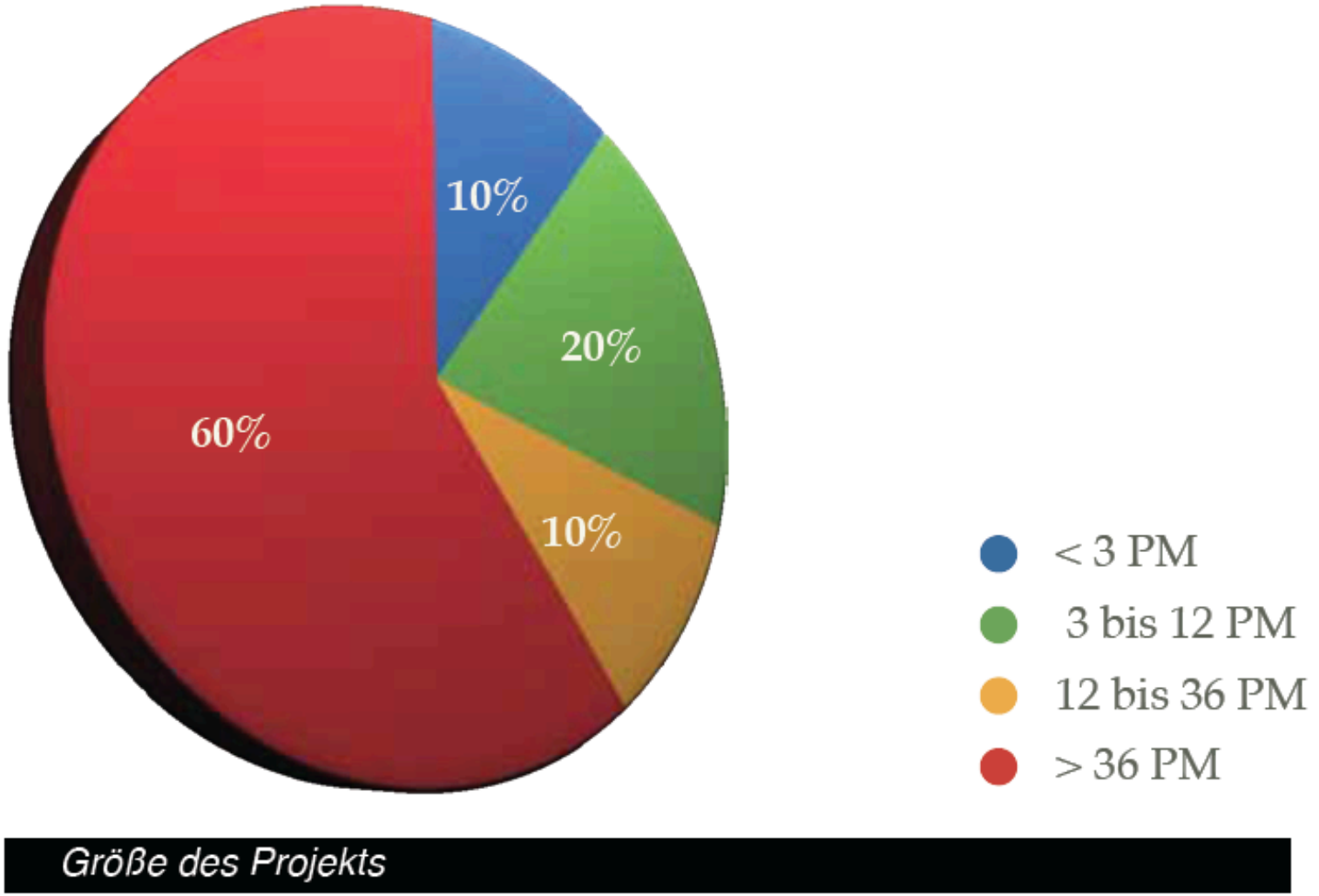
- Wann wurde die Entscheidung für das WebFramework in der Firma getroffen?



Quelle: http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/

Marktstudie mit Befragung (Ausschnitte)

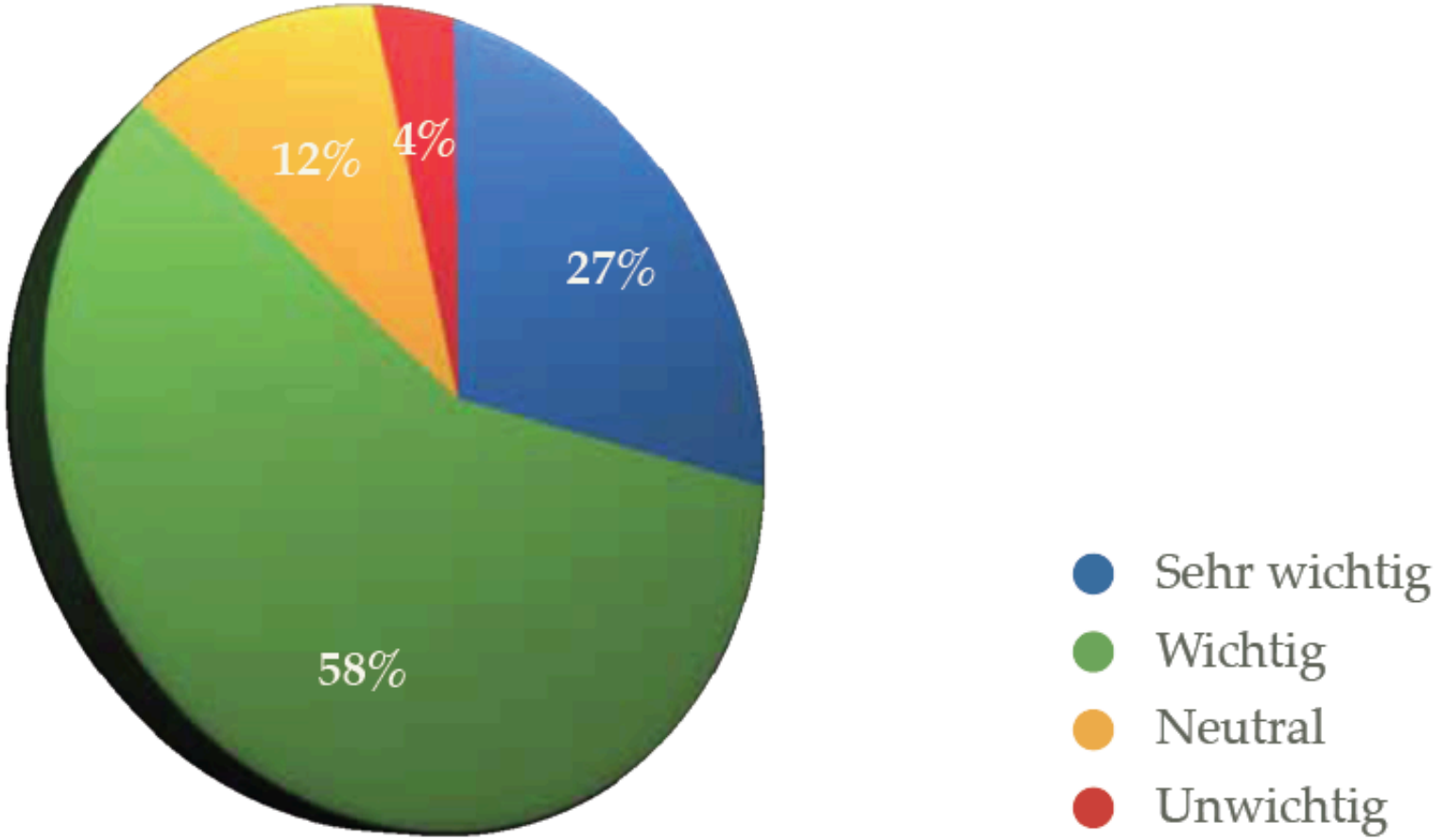
- Frage zur Investitionsgröße des Webentwicklungsprojektes



Quelle: http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/

Marktstudie mit Befragung (Ausschnitte)

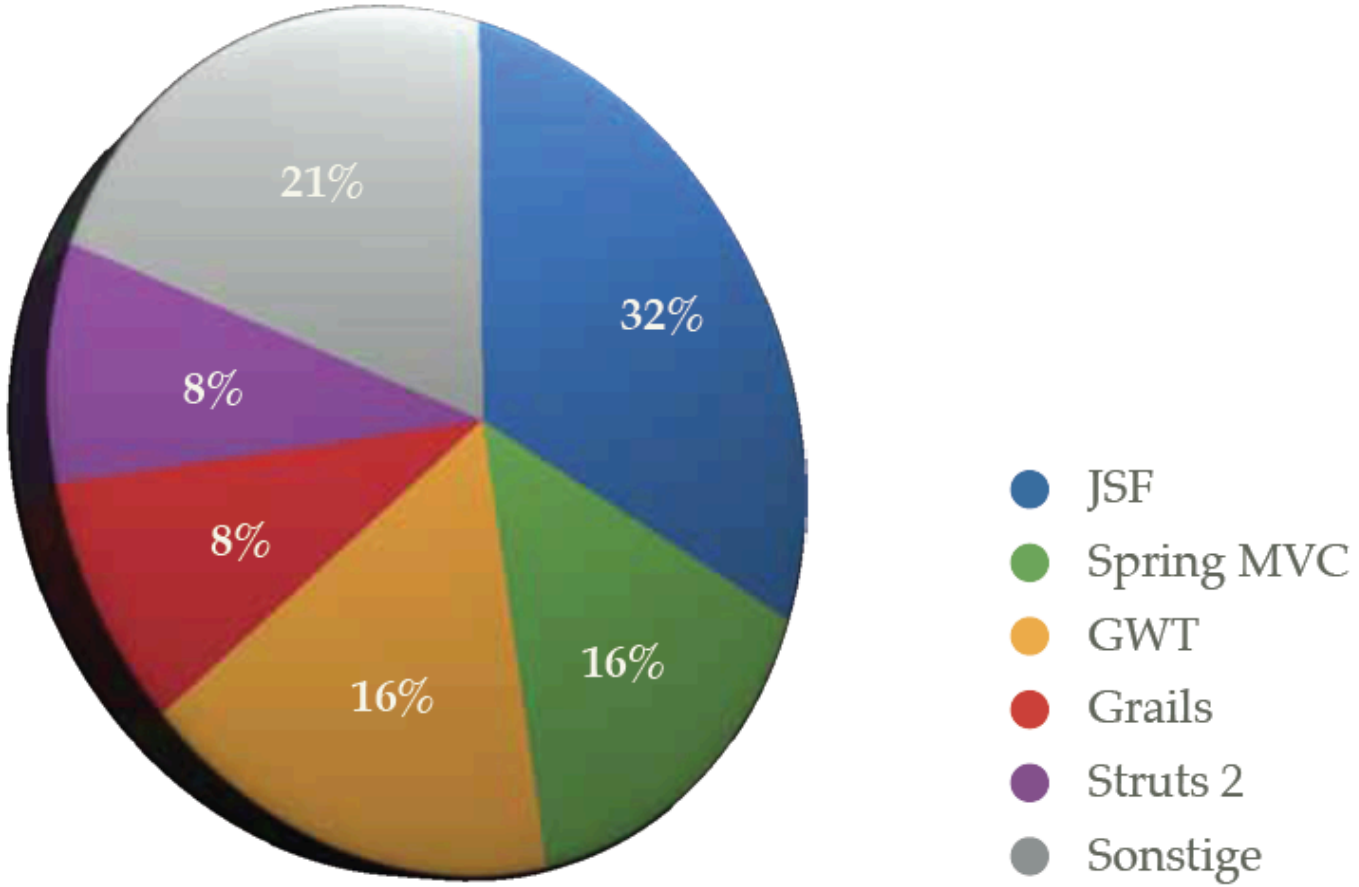
- Frage nach Erfordernissen des Refactoring über mehrere Schichten



Refactoring über mehrere Schichten der Anwendung

Quelle: http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/

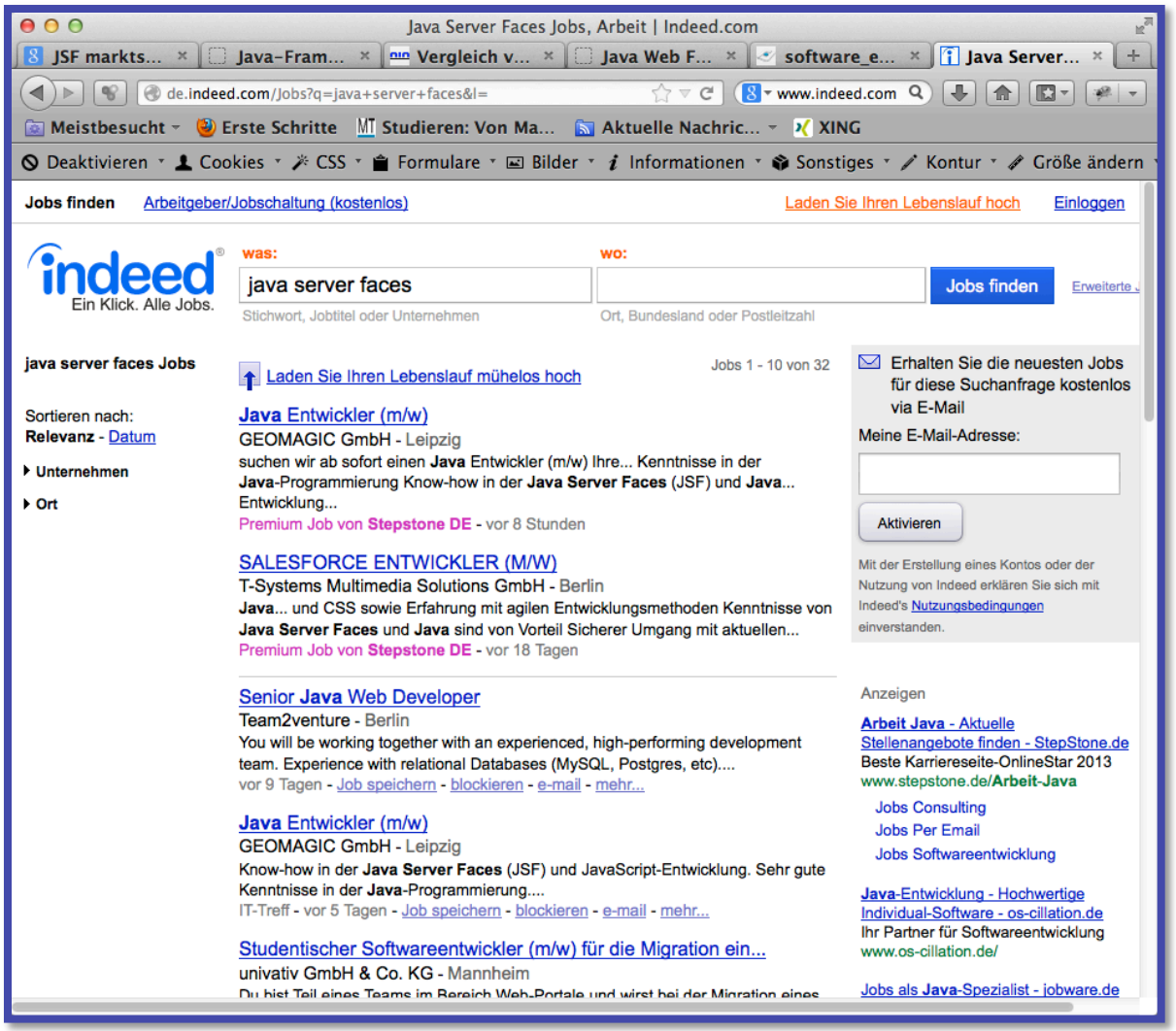
Marktstudie mit Befragung (Ausschnitte)



Eingesetztes Webframework

Quelle: http://www.oio.de/kompass/ueberblick-java-web-frameworks-vergleich/

Jobscreening (Beispiel, Juni 2014)



32 Suchergebnisse (bundesweit) für „Java Server Faces“, 673 für „JSF“
5 Suchergebnisse für „JSF“ (Bielefeld) u.a. bei Bertelsmann AG Bielefeld, b-next engineering GmbH Herford

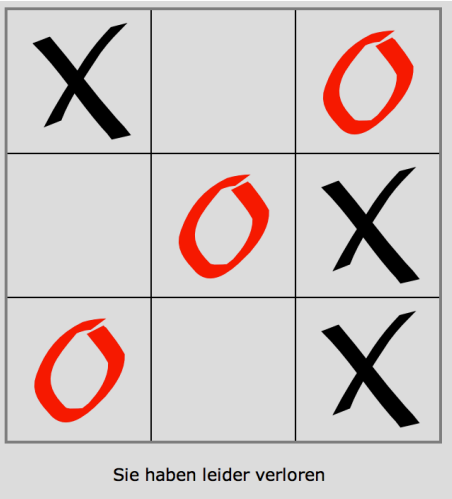

Softwareentwickler (m/w) Java/JEE für Front- und Backend
PlusServer - Köln
Konzeption und Implementierung effizienter Softwarelösungen zur Prozesssteuerung und Datenverwaltung unter Verwendung von Java/JEE in einer serviceorientierten Architektur
• Entwurf und Implementierung von Oberflächen mittels JSF, HTML, JavaScript (Frameworks)
• Aktive Mitgestaltung einer modernen und skalierbaren Softwarearchitektur Ü
• bernahme von Verantwortung für Arbeitspakete in Projekten mit agiler Vorgehensweise
• Direkte Einflussnahme auf die Servicelandschaft und die eingesetzten Technologien
• Identifikation und Umsetzung von Optimierungen
• Durchführung von Kompone
Köln
Java Web Developer - Java, GWT, JSP, JSF, Berlin
Teilen
Gehaltsbenchmark
Nur für registrierte Mitglieder!
Ort
Berlin
Karrierelevel
Senior Fachkraft / Projektleiter
Branche
Softwareunternehmen
Details zum Job Java Web Developer - Java, GWT, JSP, JSF, Berlin in Berlin:
Java Web Developer - Google Web Toolkit, Java, Web, GWT, Software, Berlin My client are a leading software development company and are searching for a talented Java Web developer, with GWT experience, to join their talented team, in the Berlin office. As a senior, hands on professional, you can expect a salary of €55,000 - €75,000. Main responsibilities: - Responsible for new features from conception, implementation, test refinement and launch - Implementation of web application functions, spanning GWT-based front-ends - Implement mobile application modules for iOS and Android - Working with architects, product management and end users Main requirements: - 3+ years experie
Berlin
Java Enterprise Developer Frontend (m/w)
Wir suchen ab sofort einen Java Enterprise Developer Frontend (m/w)
Goodgame Studios ist die am schnellsten wachsende Spielefirma Europas und eines der erfolgreichsten Technologieunternehmen in Deutschland. Mit mehr als 800 Mitarbeitern entwickeln wir am Standort Hamburg innovative Online Games, die von über 200 Millionen Spielern in über 200 Ländern und in 25 Sprachen weltweit gespielt werden. Die Grundlage unseres Erfolgs bildet unser talentiertes und motiviertes Team aus Experten und Neueinsteigern. Werde Teil unserer Erfolgsstory und bewirb Dich jetzt!
Als Java Enterprise Developer Frontend (m/w) arbeitest Du in einem interdisziplinären Team an der Entwicklung und Wartung eines neuen Frontends auf JSF-Basis für den unternehmensinternen Einsatz. Du bringst eigene Ideen und Deine Erfahrung gezielt in die Konzeption und den Entwurf neuer Komponenten ein und sorgst dafür, dass die Systeme reaktionsschnell und hochverfügbar sind.
Dein Job:
• Projektarbeit in einem modernen und agilen Umfeld auf der Technologieplattform Java EE 7
• Web Frontend-Entwicklung mit JSF 2.2, Primefaces, Ajax und WebSockets
• Backend-Anbindung mit CDI
• Arbeiten an einem Top-Prio-Projekt nah am Management zur Skalierung des Unternehmenserfolgs
Dein Profil:
• Berufserfahrung im Bereich Web Development und mit umfangreichen Java-Projekten
• Kenntnisse in der Arbeit mit einem Application Server (WildFly, JBoss)
• Erfahrungen mit WebSockets sind hilfreich, aber nicht Voraussetzung
• Spaß am Arbeiten in interdisziplinären Teams und mit modernen Technologien
• Gute Deutsch- und Englischkenntnisse
Hamburg

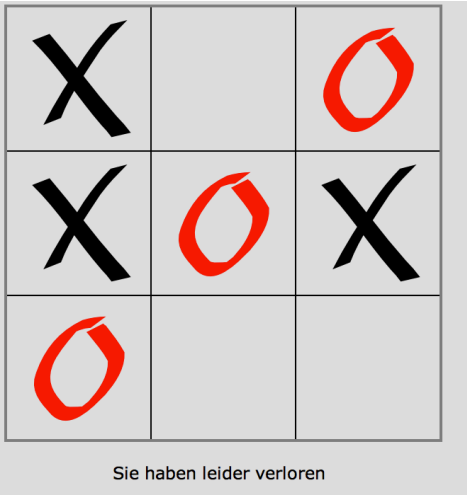
Heute in der Veranstaltung

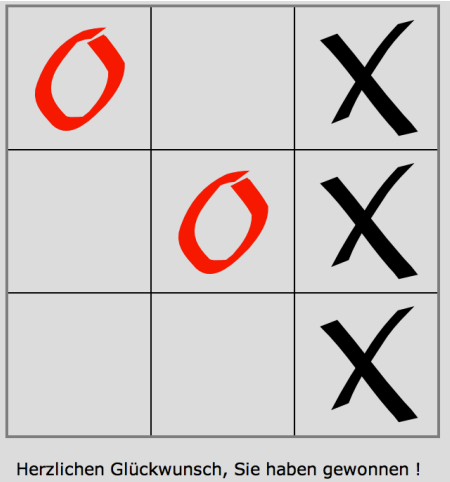
1. **Java Server Faces Teil 1**
   - Motivation aus Entwicklersicht
   - Marktwirtschaftliche Motivation zu JSF
   - **Einführungsbeispiel Tic Tac Toe**

Kinderspiel Tic Tak Toe mit JSF ?

- **Amerikanisches Kinderspiel, bekannt durch Film War-Games**
- **Zwei Spieler besetzen abwechselnd Feld auf 3×3 Spielfeld**
- **Sieger ist, wer zuerst drei Felder horizontal, vertikal oder diagonal besitzt**
- **Computer (Kreis) und Benutzer (Kreuz) spielen gegeneinander**







Implementierung (einfache Spielelogik)

- *aus Sicht des Lehrbeispiels ist das Business-Modell weniger relevant – nur die Oberfläche (View) wird betrachtet*
- **Felder durchnummeriert von 0 bis 8**
- **Feld entweder leer oder von einem Spieler belegt:**
  - **`KREIS (Computer) , KREUZ (User) , LEER`**
- **Methoden `isGewonnen()`, `isVerloren()` aus Computersicht**
- **`isFertig()` Test auf Spielende**
- **`waehleZug()` für Rechner, `setzte(int)` für Mensch**
- **`getBrett()` gibt Brett zur Darstellung zurück**

Interface Brett (Listing)

```
public interface Brett {
public static final Boolean KREIS = Boolean.TRUE ; //Rechner
public static final Boolean KREUZ = Boolean.FALSE ; //Spieler
public static final Boolean LEER = null ;

public boolean isGewonnen (); // Computer hat gewonnen
public boolean isVerloren (); // Computer hat verloren
public boolean isFertig (); // Spielende

public void waehleZug (); // Computer wählt neuen Zug

// Spieler setzt auf Feld i (0 bis 8)
public void setze (int i) throws IllegalArgumentException;
                                          //Feld bereits belegt
public Boolean [] getBrett (); // Brettbelegung als boolesches
                                    array
}
```

Tic-Tac-Toe als JSF-Seite (Teil 1)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
<head>
        <title>Tic Tac Toe</title>
        <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

... *weiter nächste Folie...*

- im `<html>`-Tag wird der Namensraum `h` eingebunden für jsf-Tags-Bibliothek
- Alle Tags mit Präfix `h` sind jsf-tags
- JSF besteht aus 2 Bibliotheken:
  - `h` : jsf;
  - `f` : Kernbibliothek (in diesem Bsp. nicht eingebunden)

Tic-Tac-Toe als JSF-Seite (Teil 2)

```
<body>
    <h:form>     //erzeugt später html-Formular
        <h:panelGrid columns="3"> //Tabelle mit drei Spalten
            <h:commandButton id="feld-0" image="#{tttHandler.image[0]}"
                  actionListener="#{tttHandler.zug}" />
            <h:commandButton id="feld-1" image="#{tttHandler.image[1]}"
                  actionListener="#{tttHandler.zug}" />
             ...
             ...alle neun Spielfelder 0 bis 8 ...
             ...
            <h:commandButton id="feld-8" image="#{tttHandler.image[8]}"
                 actionListener="#{tttHandler.zug}" />
        </h:panelGrid>
        <br />
        <h:outputText id="meldung" value="#{tttHandler.meldung}" />
        //Ausgabe des Spielresultats
        <br /><br />
        <h:commandButton value="Neues Spiel" action="#{tttHandler.neuesSpiel}"/>
        // Schaltfläche für Neubeginn des Spiels
    </h:form>
</body>
</html>
```

In dem Tag `<h:panelGrid columns="3">` enthaltene Elemente `<h:commandButton>` werden zeilenweise auf drei Spalten aufgeteilt.

Erläuterungen zu Tic-Tac-Toe als JSF-Seite

`<h:form>`
Falls Interaktivität gewünscht, muss es ein Formular sein (HTML-Formular)

`<h:panelGrid>`
Tabelle mit Angabe der Spaltenanzahl **`columns="3"`**

`<h:commandButton>`
Schaltfläche mit Id, Image und Action-Listener

`tttHandler`
Eine Backing-Bean oder Handler zur Kommunikation mit Java

`#{ ... }`
Ausdrücke der JSF-Expression-Language

Erläuterungen zum tttHandler

- bindet Properties eines Java-Objektes an UI-Komponenten
- Properties sind in JSF Attribute nach der Java-Bean-Spezifikation mit
  getter- und ***setter***-Methoden

- die Attribute des Java-Objektes:
  - `public String getMeldung()` *//getter für Meldung*
  - `public String[] getImage()` *//getter für Attribut image*
  - `public void zug(ActionEvent)` *//Event listener*
    //Methode zug
    //für das Drücken der 9 Knöpfe
    ///wichtigste Methode im Spiel
  - `public String neuesSpiel()`

- Verbinden der jsf-Seite mit Java erfolgt komponentenbasiert über Java-Bean.
- JSF hat das Event-Source/Event-Listener-Modell von AWT und SWING übernommen.

Codeausschnitt der Klasse TicTacToeHandler.java

```
@ManagedBean(name = "tttHandler")
@SessionScoped
public class TicTacToeHandler {

private static final String KREIS = "/pages/images/kreis.png ";
private static final String KREUZ = "/pages/images/kreuz.png ";
private static final String LEER = "/pages/images/leer.png";

private Brett brett ;                    //Instanzvariable der Modellklasse brett
private String meldung ;                 //Instanzvariable  der Modellklasse meldung

public TicTacToeHandler () {
   brett = new BrettImpl ();
   }

public void zug ( ActionEvent ae) {     // nächste Folie
  ...
   }
public String [] getImage () {...}       // getter Image übernächste Folie

public String neuesSpiel() {...}
public String getMeldung() {return meldung;} //getter Meldung
public void setMeldung(String meldung) {this.meldung = meldung;} //setter Meldung
}
```

Methode zug(ActionEvent)

```
public void zug ( ActionEvent ae) {
     if ( brett.isFertig ())
         return ;
         try {
         brett.setze ( new Integer (ae.getComponent().getId().split ("-")[1]));
         if ( brett.isVerloren ()) {
         meldung = " Herzlichen Glueckwunsch , Sie haben gewonnen ";
         return;
         }
         brett.waehleZug();
         if (brett.isGewonnen()) {
                  meldung = " Sie haben leider verloren ";
         }
         } catch ( Exception e) {
         log.info (" Kein Spielerzug ausgefuehrt ");
         }
}
```

- Methode zug(ActionEvent) ist eine typische ***Controller-Methode***
- Aufruf bei Mausklick durch Spieler auf das Spielfeld

Methode `getImage()`

```
public String[] getImage() {
 String[] feld = new String[9];
 for (int i = 0; i < brett.getBrett().length; i++) {
   if (brett.getBrett()[i] == Brett.KREIS) {
      feld[i] = KREIS;
   } else if (brett.getBrett()[i] == Brett.KREUZ) {
      feld[i] = KREUZ;
   } else {
      feld[i] = LEER;
          }
  }
  return feld;
}
```

- liefert array von Dateinamen für die Darstellung der Bilder für die drei versch. Belegungen

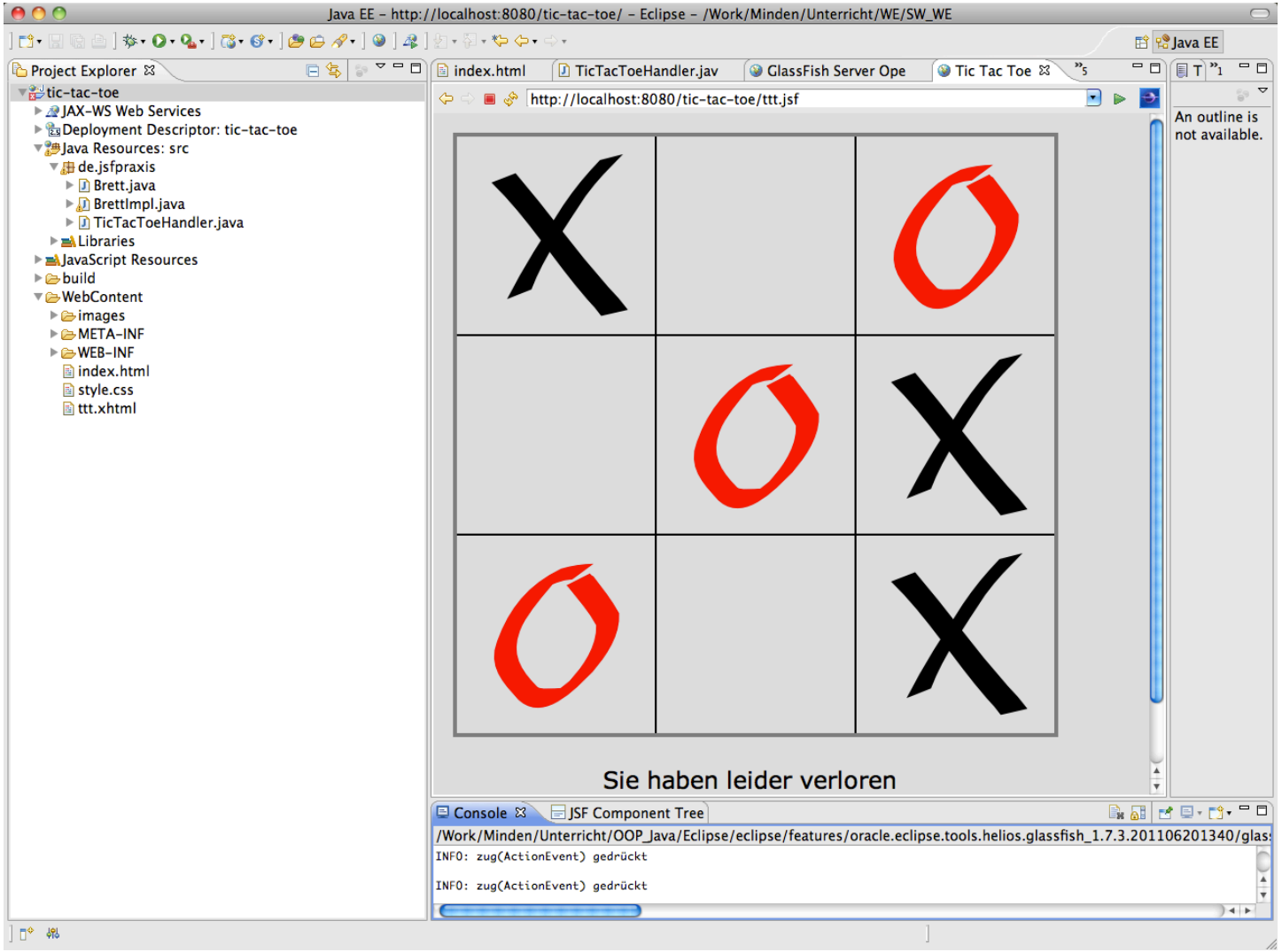
Automatisierte Java-Bean – Verwaltung im ersten JSF-Beispiel

- JSF-Implementierung erzeugt Objekt des **`tttHandler`** automatisch bereits vor dem ersten Zugriff
- JSF definiert automatisierte Bean-Verwaltung durch Annotation:

```
@ManagedBean

@SessionScoped      //Objekt bleibt über die Lebenszeit der Session bestehen
```

Tic-Tac-Toe Demo



Literatur für JSF (1/2)

Bernd Müller
JavaServer Faces 2.0
Ein Arbeitsbuch für die Praxis
Hanser Verlag
ISBN 978-3-446-41992-6

Literatur für JSF (2/2)

- Martin Marinschek, Michael Kurz, Gerald Mülan:***„Java Setrver Faces 2.0, Grundlagen und erweiterte Konzepte*** “ dpunkt.verlag, Heidelberg 2009 ISBN 3-89864-234-8