Webová Aplikace v Javě pomocí frameworků

KIV/PIA 2013
Jan Tichava

Přehled technologií

- Java EE

- JSF, PrimeFaces
- Spring
- JPA, EclipseLink

Použití frameworku

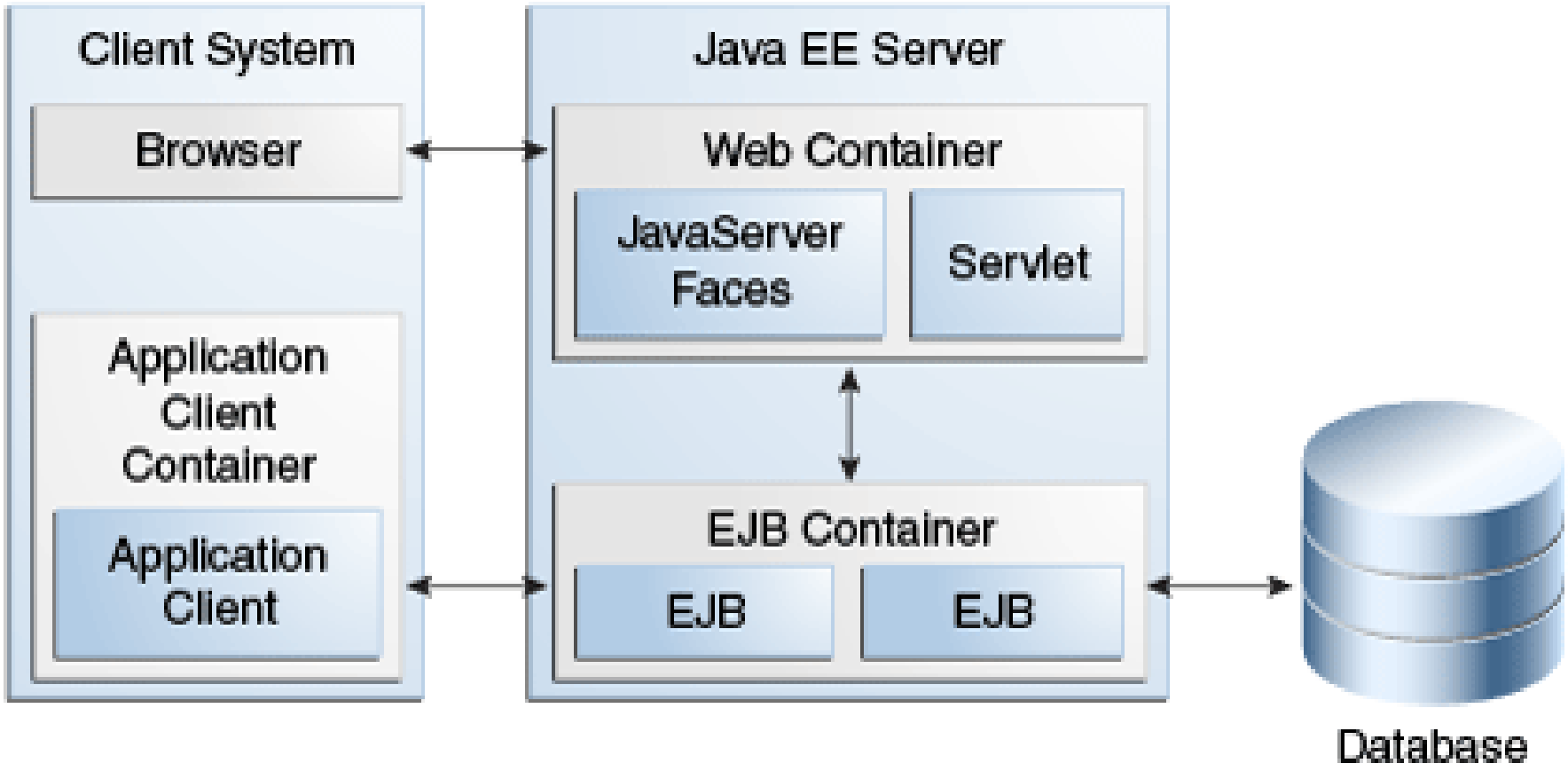
Java EE

Java Platform, Enterprise Edition

Přehled Java EE

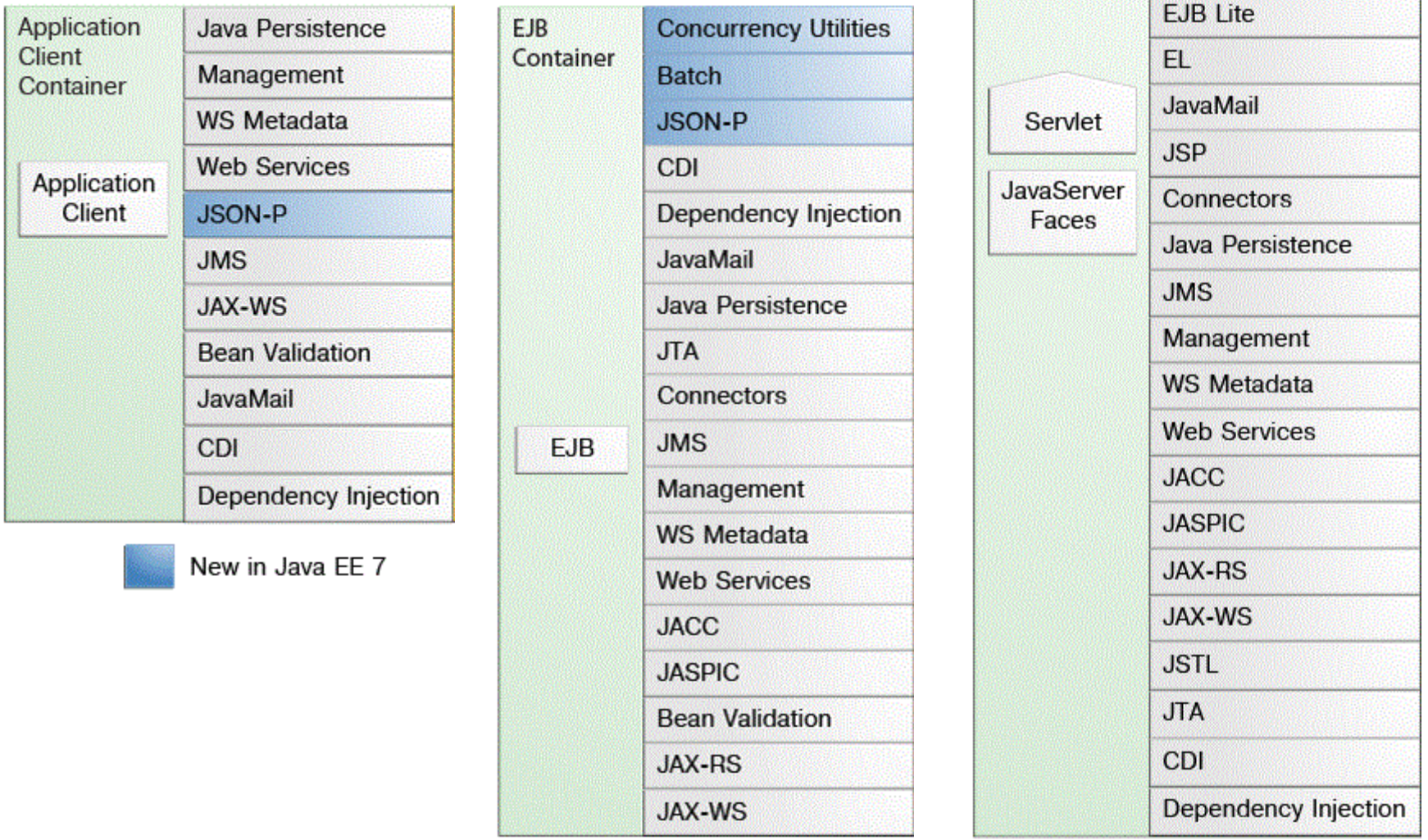
Persistence	• Java Persistence API • Enterprise Java Beans
Zobrazovací vrstva	• JavaServer Faces • Servlety, JSP
Interakce aplikací	• Java Messaging System • Webové služby
Deployment	• EAR, WAR, RAR • Anotace namísto některých deskriptorů
Nástroje	• Zpracování XML, JavaMail API, JNDI, JAAS a další…

Přehled Java EE

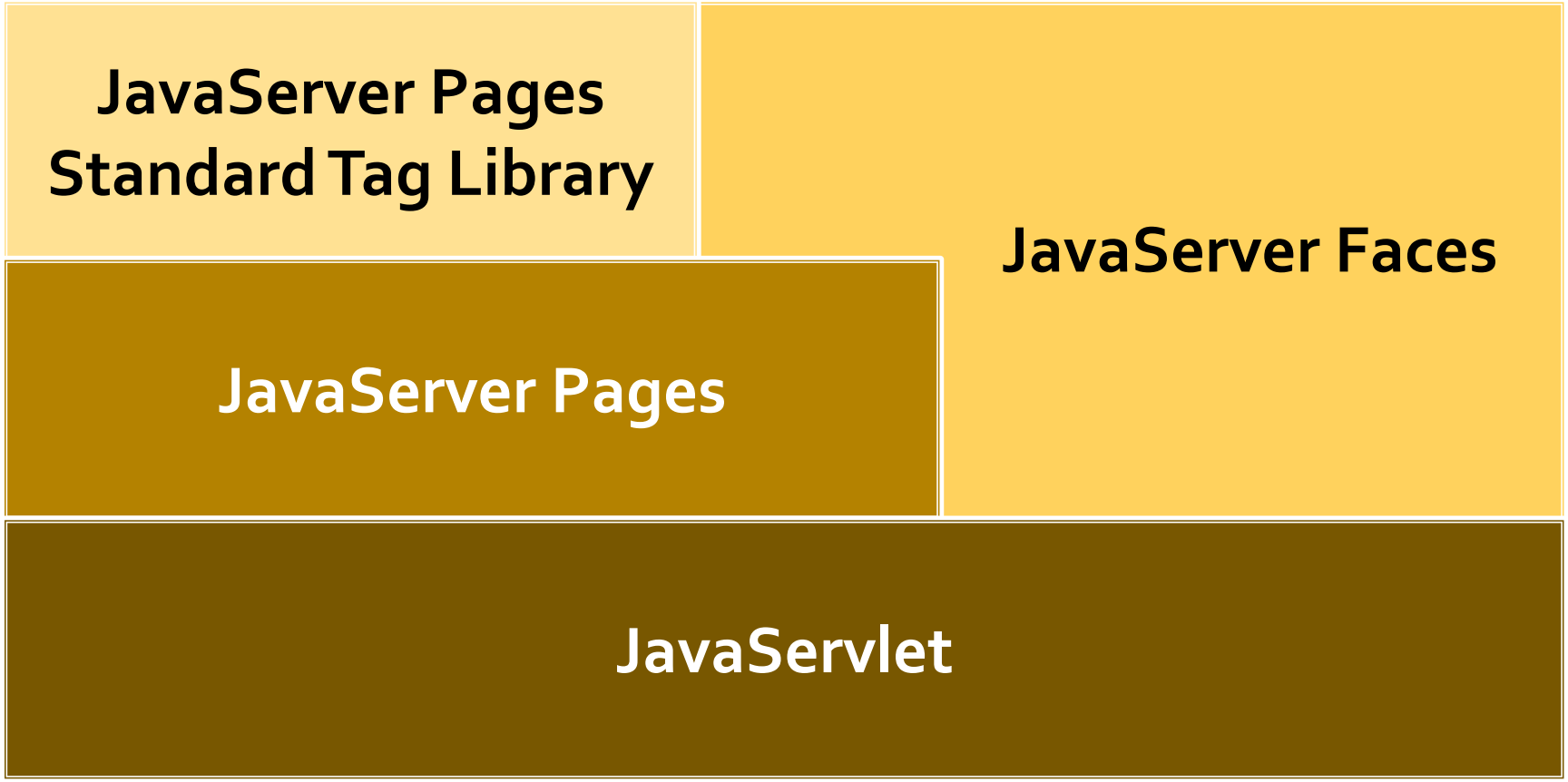


http://docs.oracle.com/javaee/7/tutorial/doc/overviewoo8.htm

Přehled Java EE



Java Web Application



JSF

Java Server Faces

Java Server Faces

- Framework webové prezentační vrstvy
  - Komponentový model (rozšiřitelný)
  - Událostně řízené programování
  - Sada základních UI komponent
- Aktuální verze: 2.2.4
  - Počátky v roce 2001
  - 2004 – JSF 1.0
  - 2009 – JSF 2.0
  - 2013 – JSF 2.2

Výhody JSF

- Čistě oddělené chování a prezentace
- Stav uchován na úrovni komponent
- Snadná vazba uživatelských událostí na kód na straně serveru
- Několik implementací
  - ICEfaces
  - MyFaces
  - RichFaces
  - PrimeFaces

Nevýhody JSF

- Velký framework
- Náročnější na prostředky
- Náročnější na naučení
- Nevhodné defaultní chybové hlášky
- Komplikovaný
  - pokud funguje → super
  - pokud nefunguje → problém
- POST
- Potřeba dalších frameworků

Výhody × nevýhody

Managed Beans

- Automatické vytváření
- JavaBeans
  - Serializovatelné
  - gettery/settery
  - Konstruktor bez argumentů
- Různé rozsahy platnosti
  - application, session, view, request
- Validace dat a obsluha událostí
- Udržování kontextu stránky
- @PostConstruct

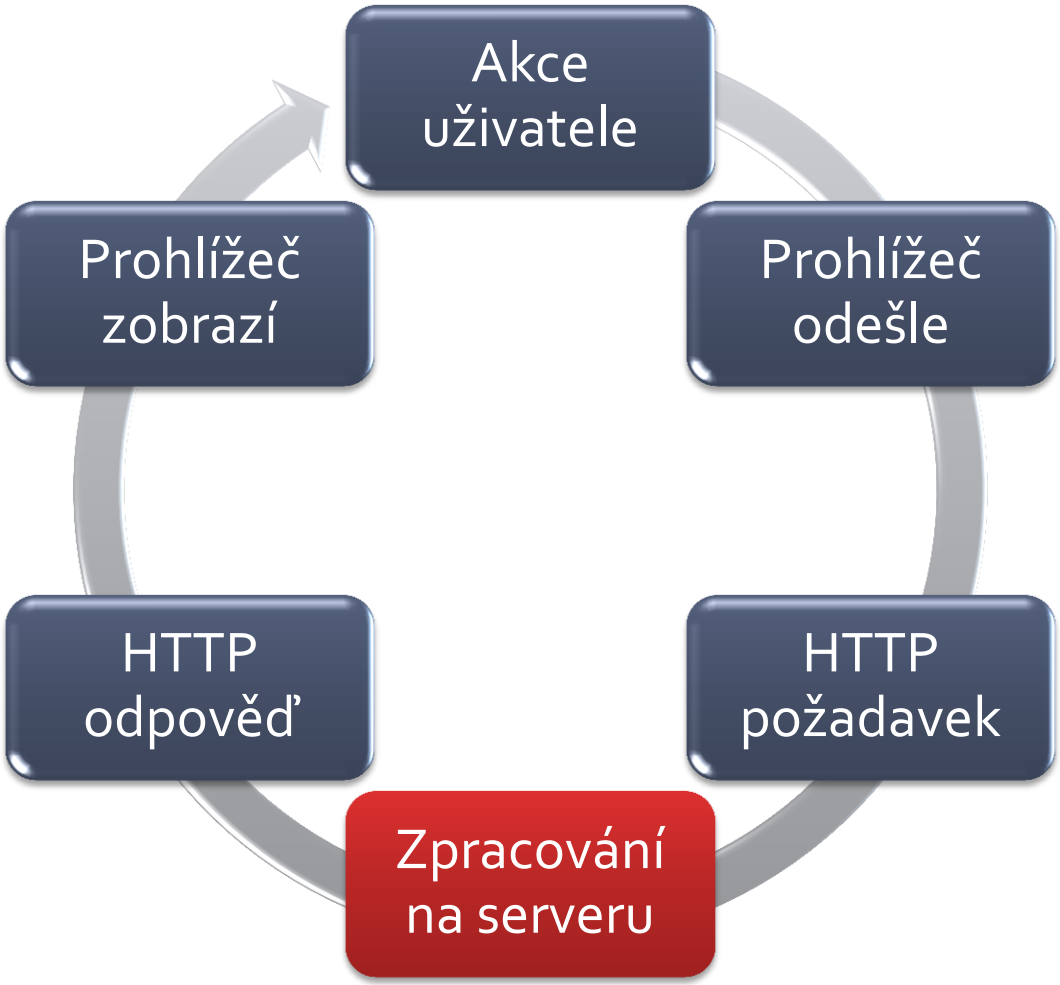
Expression Language

- Uzavřen do #{…}
- Typy výrazů
  - Value Expression
    - Hledá se hodnota
    - rvalue, lvalue
    - Doporučuje se vždy poskytovat getter i setter
    - <h:inputText value="#{user.name}" />
  - Method Expression
    - Hledá se metoda
    - <h:commandButton action="#{user.save}" />

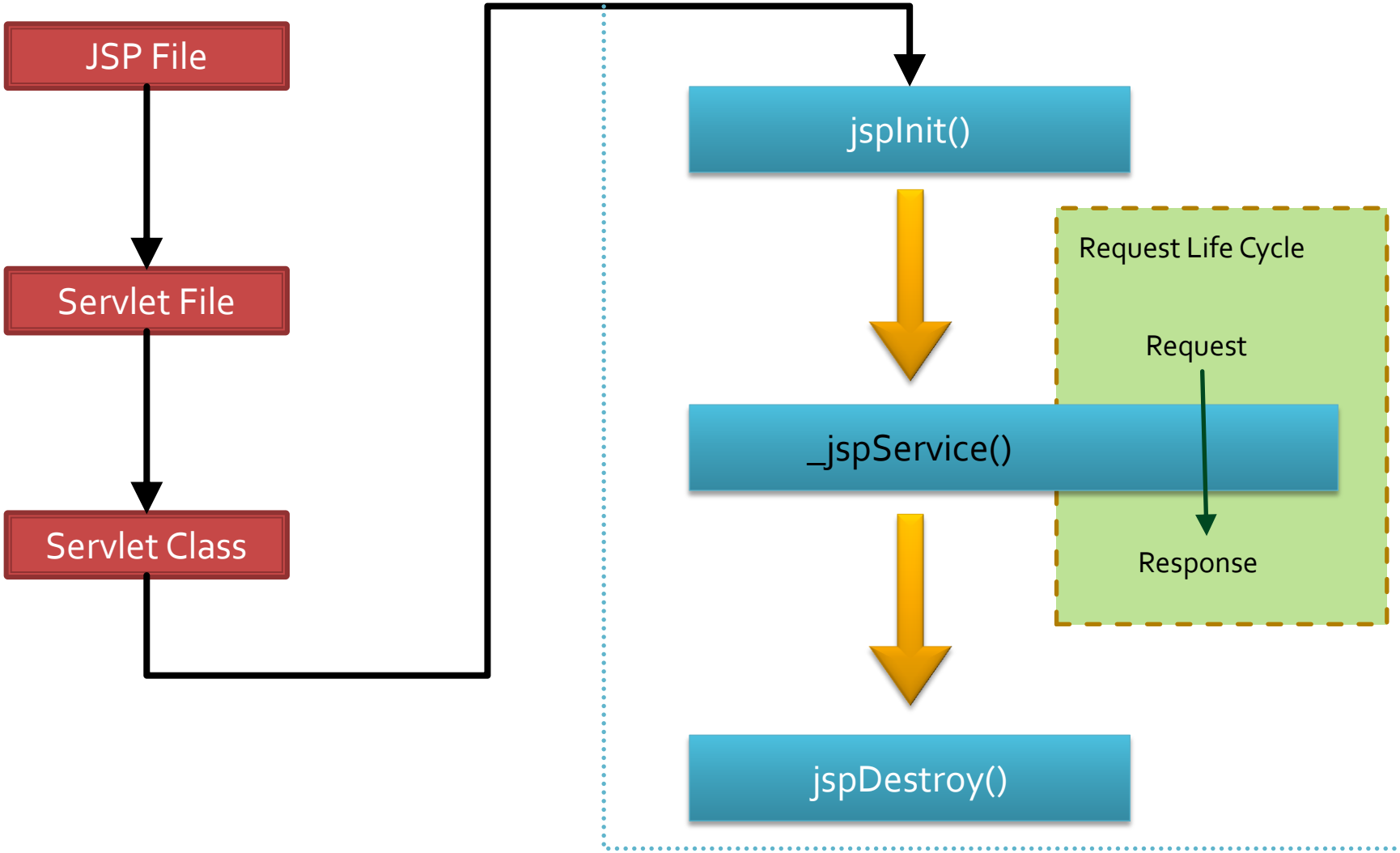
Automatické volání get/set

- #{user.property}
  - private T property;
  - public T getProperty();
  - public void setProperty(T o);
- #{user.booleanProperty}
  - private boolean booleanProperty;
  - public boolean isBooleanProperty()
  - public boolean getBooleanProperty();
  - public void setBooleanProperty(boolean value);

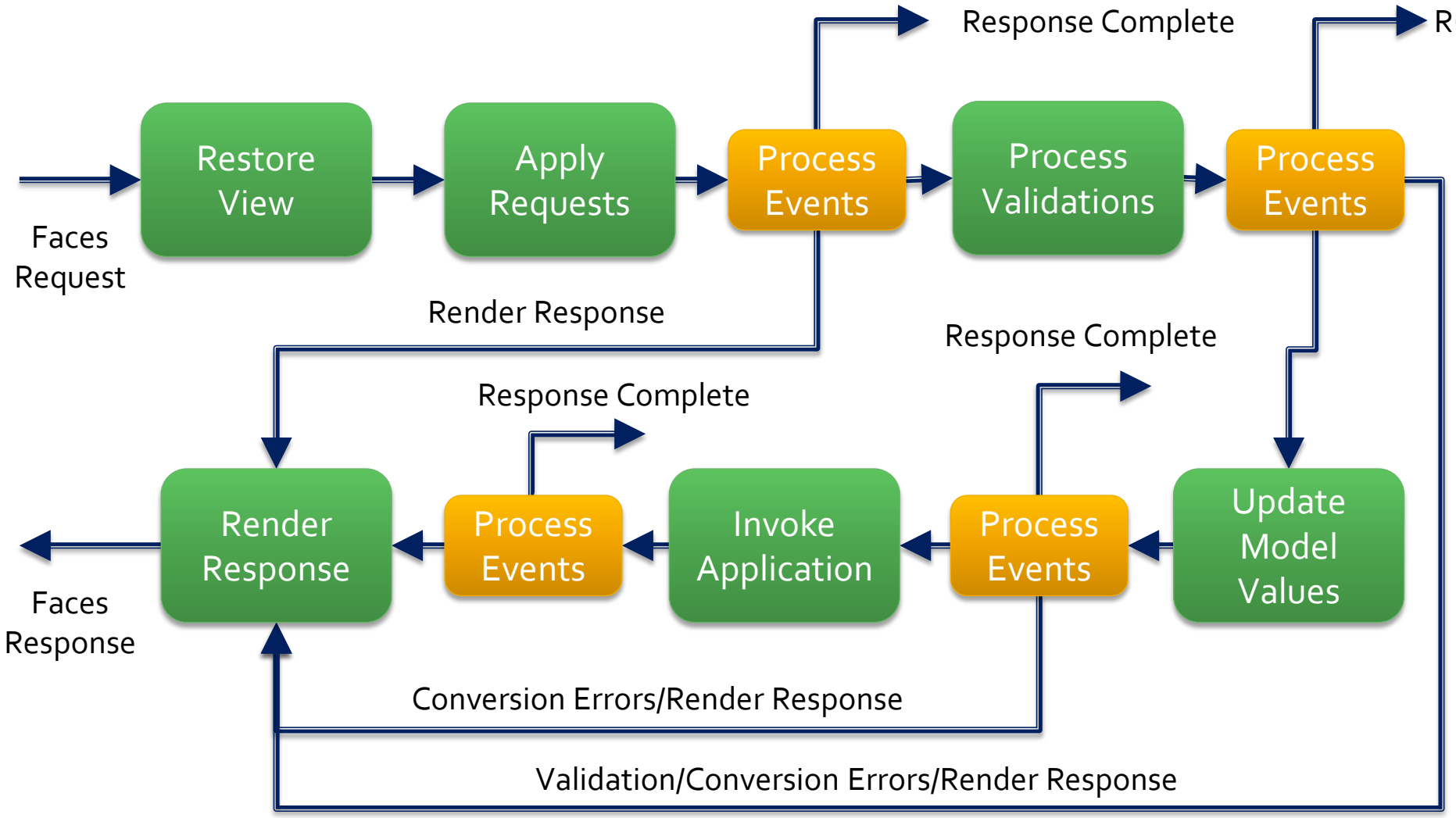
Opakování: Request/Response



Opakování: JSP Life Cycle



JSF Life Cycle



JSF MVC

Model

- Managed Bean

View

- JSF UI komponenty

Controller

- Faces Servlet

JSF MVC

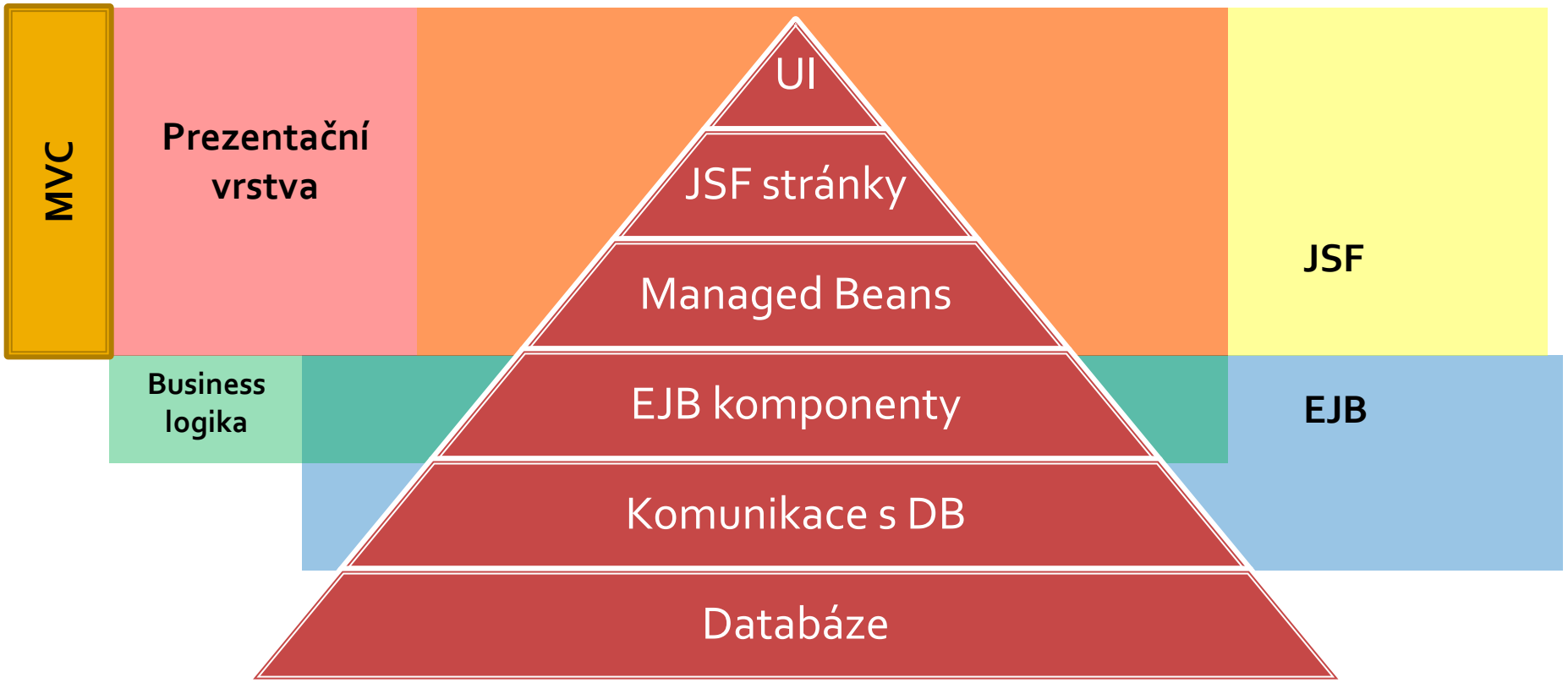
PATŘÍ

- Události
- Formátování výstupu
- Validace
- Navigace
- Lokalizace
- Zobrazování chyb
- Volání business logiky

NEPATŘÍ

- Operace nad DB
- Výpočty
- Business logika
- „Transakce“

JSF v architektuře Java EE

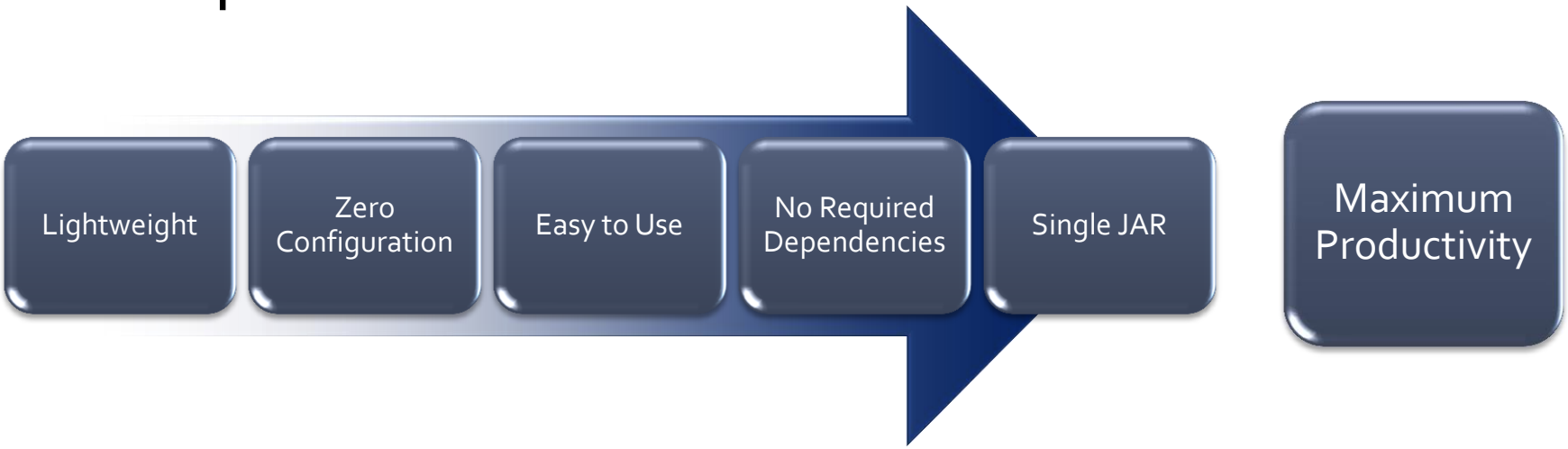


PrimeFaces

The myth says one day a JSF component library will come and rule them all

PrimeFaces

- Prime Teknoloji, podzim 2008
  - http://primefaces.org/
- PrimeMobile
- Open-Source
- Enterprise



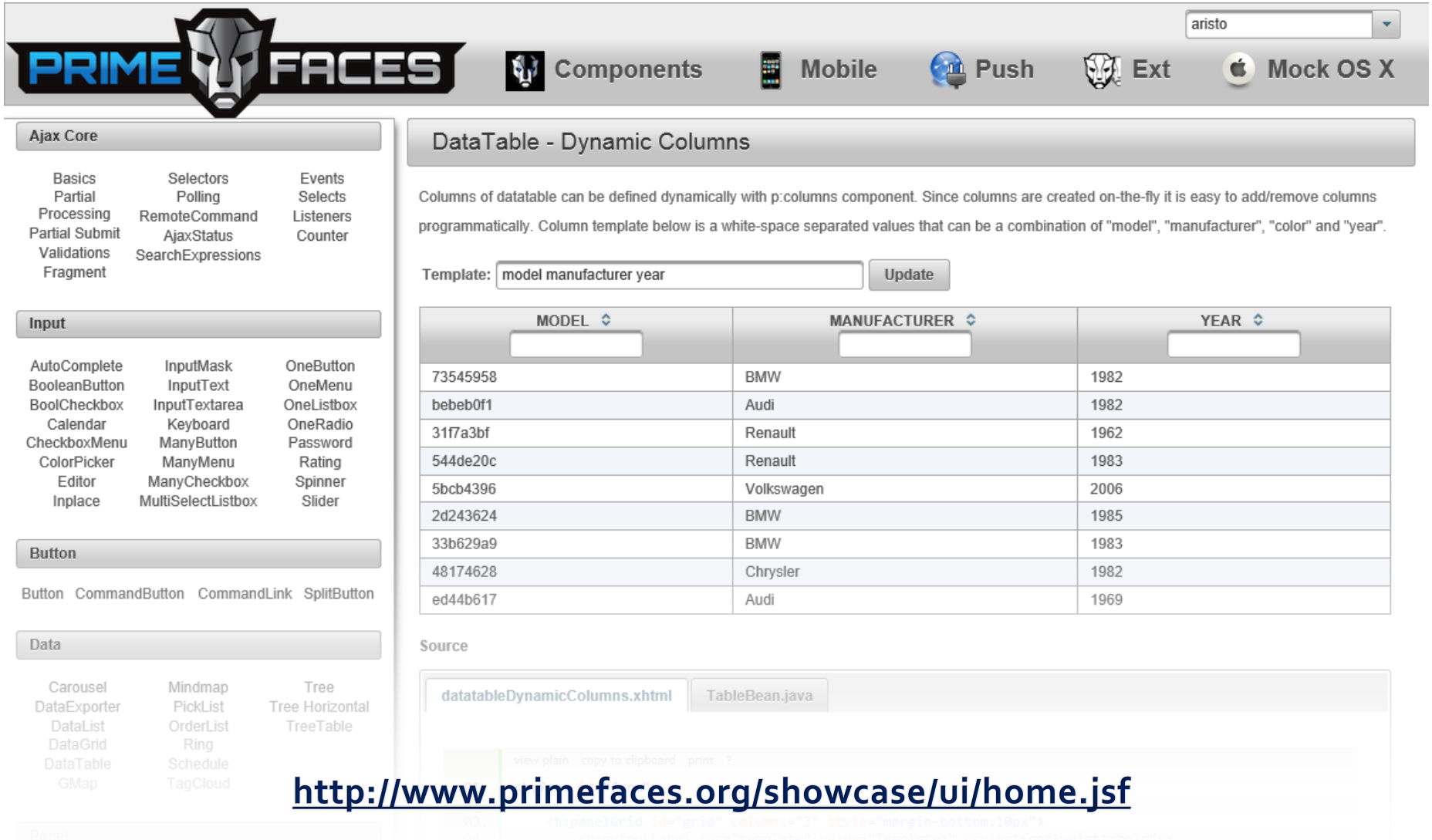
PrimeFaces

- 100+ UI komponent
- HTML5
- AJAX
  - Server API: JSF 2.2
  - Klient API: jQuery
- Témata (skiny)
- „inspirací“ pro IceFaces
- Push notifikace

PrimeFaces

- Stable
  - 4.0.0
  - 4.0.4
- RoadMap
  - polovina 2013 – .NET
  - 1. čtvrtletí 2014 – 4.1
- PrimeUI
- PrimeFaces Extensions

PrimeFaces ShowCase



http://www.primefaces.org/showcase/ui/home.jsf

Stručně Spring

Let's build a better Enterprise

Spring

- open-source framework
- historie
  - červen 2003 – první verze
  - březen 2004 – verze 1.0
  - listopad 2013 – verze 3.2.5
- budoucnost
  - konec roku 2013 – verze 4.0
  - aktálně 4.0 RC1

Není to tak jednoduché

- modulární framework
- části vyvíjené nezávisle na sobě
  - a nezávisle verzované

Spring Framework

- dependency injection
- transaction management
- webové aplikace, RESTful web service
- data access, messaging
  - podpora pro JDBC, JPA, JMS
- AOP
- testování

Spring Security

- úplné a rozšiřitelná podpora pro autentifikaci a autorizaci
- ochrana proti případným útokům
- integrace se Servlet API
- podpora ověřování vůči AD a LDAP

Spring Data

- Spring Data JPA
- Spring Data MongoDB
- Spring Data JDBC Extensions
- Spring Data REST

- a další

JPA, EclipseLink

Java Persistence API

JPA – Java Persistence API

- od vývojářů EJB
- květen 2006 verze 1.0
- mnoho implementací
- obsahuje:
  - API – javax.persistence
  - Java Persistence Query Language (JPQL)
  - ORM
- více informací: Java EE 7 Tutorial

EclipseLink

- jedna z implementaci JPA
- open-source
- EL byl referenční implementací JPA 2.0
  - eclipse.org/press-release/...
- verze 2.5.1

EclipseLink a Eclipse

- generování
  - tabulka → entita
  - entita → tabulka
- jednoduchý viewer databáze
- okamžitá kontrola entit
  - zda v tabulce existuje sloupek s daným názvem

Entita

```java
@Entity
@Table(name = "users", schema = "pia")
@NamedQuery(name = "User.findAll", query = "SELECT u FROM User u")
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @Column(unique = true, nullable = false)
    private Integer id;

    @Column(name = "user_name", nullable = false, length = 64)
    private String userName;

    public Integer getId() { return this.id; }
    public void setId(Integer id) { this.id = id; }

    public String getUserName() { return this.userName; }
    public void setUserName(String userName) { this.userName = userName; }
}
```

JPQL – „SQL" zápis

- Jednoduchý dotaz

```
public List<Sensor> getSensors() {
  return this.em.createQuery(
      "SELECT s FROM Sensor s ORDER BY s.id",
      Sensor.class)
      .getResultList();
}
```

- Složitější dotaz

```
public Temperature getLastTemperature(Sensor sensor) {
  return this.em.createQuery(
    "SELECT t
     FROM Temperature t
     WHERE t.sensor = :sensor
     ORDER BY t.observed DESC",
    Temperature.class)
    .setParameter("sensor", sensor)
    .setMaxResults(1).getSingleResult();
}
```

JPQL – Jiný zápis

```
public Long getTemperatureRecordLastDay(Sensor sensor) {
  CriteriaBuilder criteriaBuilder = this.em.getCriteriaBuilder();
  CriteriaQuery<Long> criteriaQuery=criteriaBuilder.createQuery(Long.class);

  Root<Temperature> temperatureLogRoot=criteriaQuery.from(Temperature.class);
  Date before24Hour=new Date(new Date().getTime()-TimeUnit.DAYS.toMillis(1));

  Predicate predicateDate = criteriaBuilder.greaterThanOrEqualTo
                    (temperatureLogRoot.<Date> get("observed"), before24Hour);

  Predicate predicateSensor =
              criteriaBuilder.equal(temperatureLogRoot.get("sensor"), sensor);

  criteriaQuery.select(criteriaBuilder.count(temperatureLogRoot));
  criteriaQuery.where(criteriaBuilder.and(predicateDate, predicateSensor));

  return this.em.createQuery(criteriaQuery).getSingleResult();
}
```

Praktická část

Jednoduchá ukázková aplikace

Ukázka

```
<ui:param name="title" value="#{msg['admin.session.title']}" />
<ui:param name="icon" value="menu-session-32.png" />

<ui:define name="content">
<p:growl id="messages" showDetail="true"/>

<p:data  e id="u  var="u"  "#{u  .u  s}"  el  tion  #{users
<!-- <p  x event="r  lect"  ste  ="#  ers.  Row  e  " u  date=":

<p:ajax  t="rowEdit  ste  s.  dit}  upd  =":  mData:mes
<p:ajax  t="rowEdit  el"  stener="#{u  rs.o  ancel}" update=":formD

<p:colum  derText=  sg['us  tName']}">
<p:cellE  >
<f:facet  ">
<h:output  alue="#{u.firstName}" />
</f:facet>
<f:facet name="input">
<p:inputText id="firstNameInput" value="#{u.firstName}" style="width:100
</f:facet>
```

Demo

Děkuji za pozornost

Doplňující dotazy: zcu@tichava.cz
Běžící ukázková aplikace: http://pia2013.tichava.cz
V případě zájmu o tisk této prezentace napište, změním na šablonu vhodnější pro tisk.

user/pass
admin/a
guest/g