

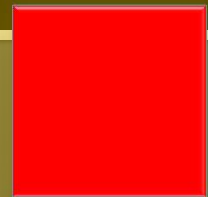
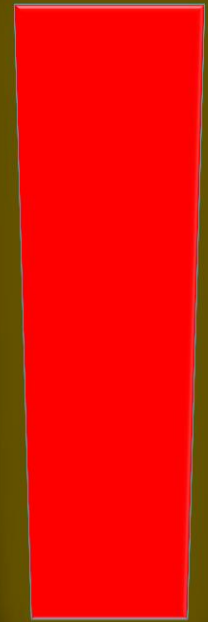
Webová Aplikace v Javě pomocí frameworků

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

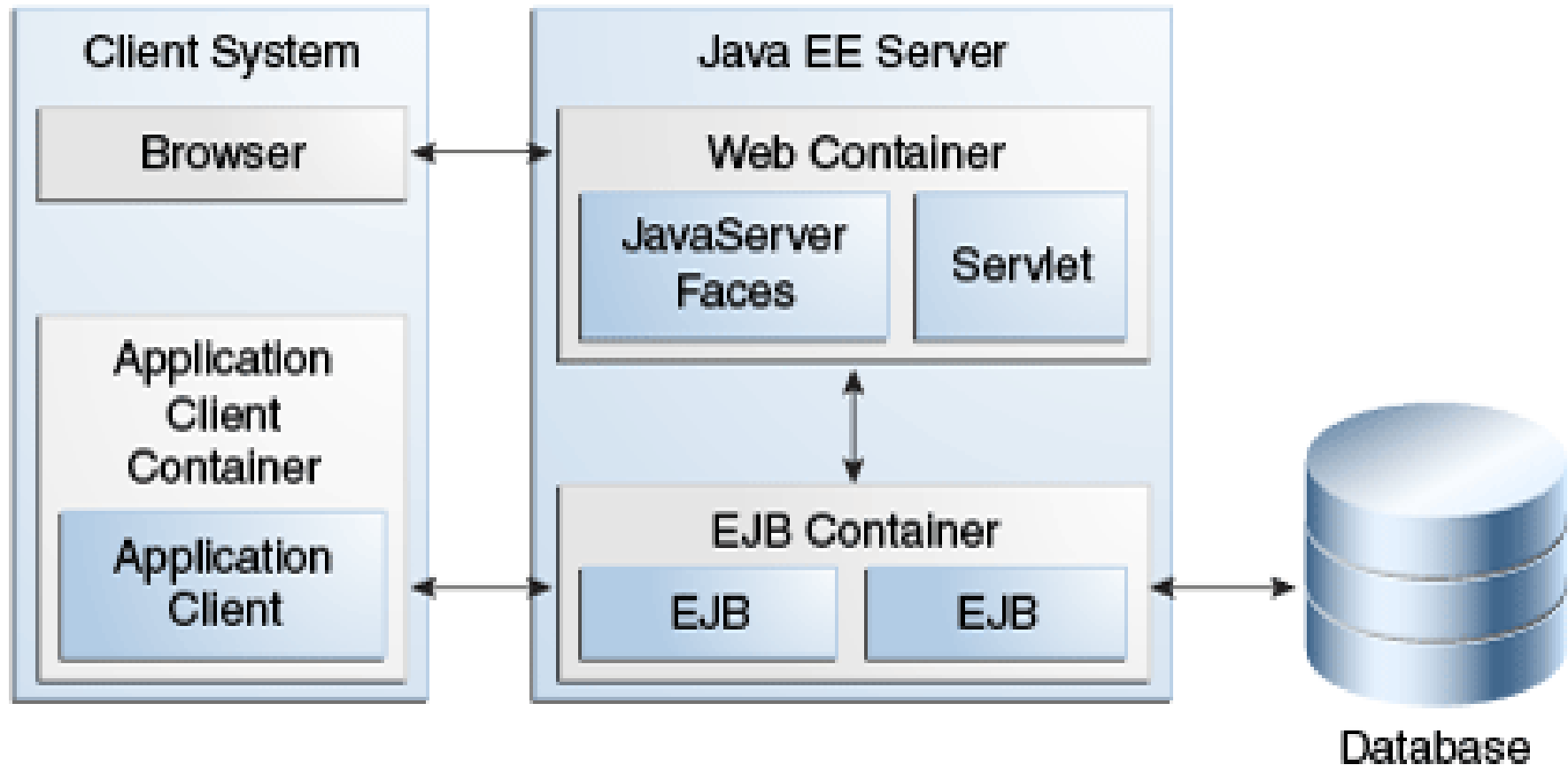
Deployment

- EAR, WAR, RAR
- Anotace namísto některých deskriptorů

Nástroje


- Zpracování XML, JavaMail API, JNDI, JAAS a další...

Přehled Java EE



Přehled Java EE

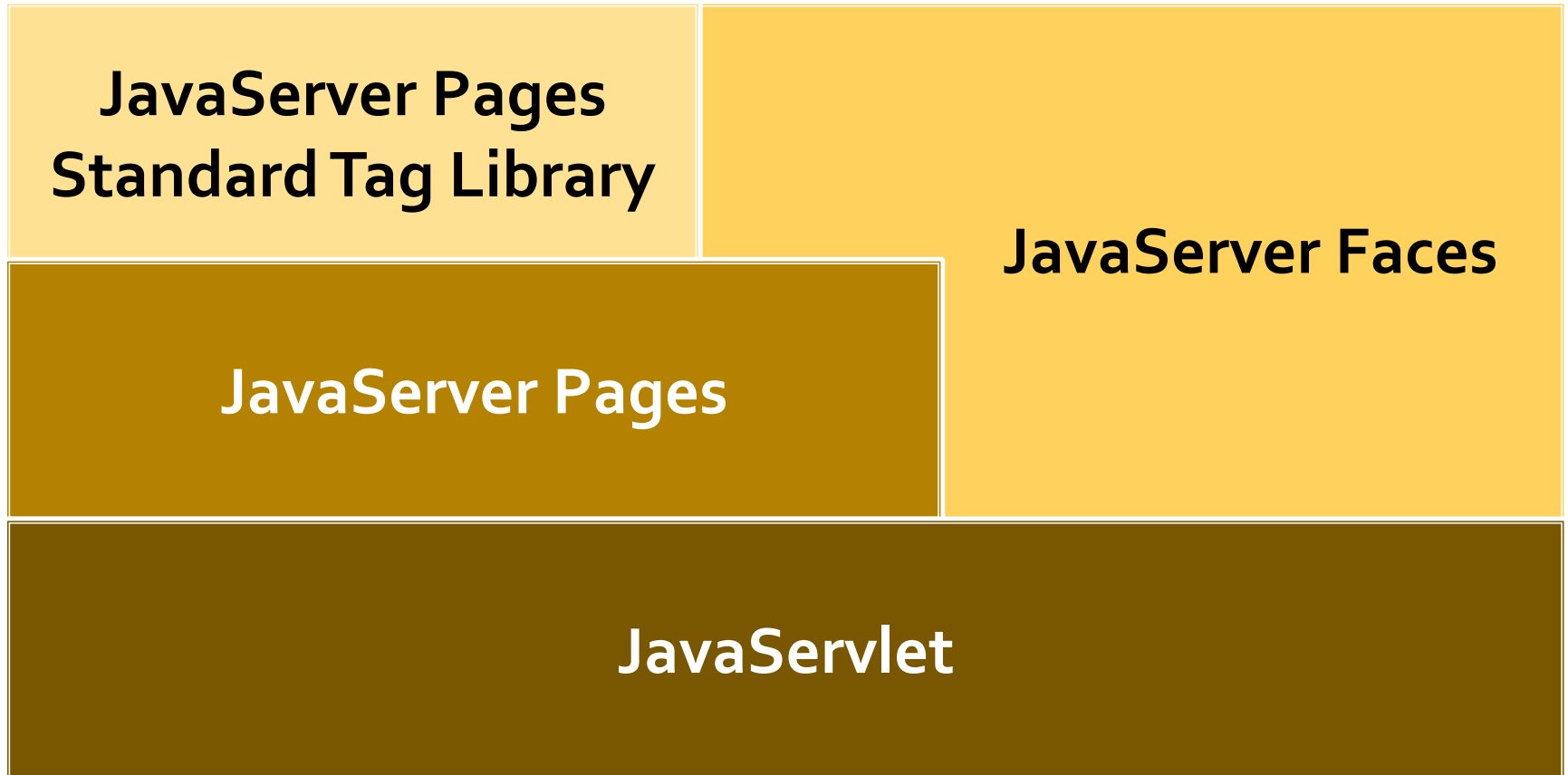
Application Client Container	Java Persistence Management
	WS Metadata
	Web Services
	JSON-P
	JMS
	JAX-WS
	Bean Validation
	JavaMail
	CDI
	Dependency Injection

 New in Java EE 7

EJB Container	Concurrency Utilities
	Batch
	JSON-P
	CDI
	Dependency Injection
	JavaMail
	Java Persistence
	JTA
	Connectors
	JMS
	Management
	WS Metadata
	Web Services
	JACC
	JASPIC
	Bean Validation
	JAX-RS
	JAX-WS

Web Container	WebSocket
	Concurrency Utilities
	Batch
	JSON-P
	Bean Validation
	EJB Lite
	EL
	Servlet
	JavaMail
	JSP
	JavaServer Faces
	Connectors
	Java Persistence
	JMS
	Management
	WS Metadata
	Web Services
	JACC
	JASPIC
	JAX-RS
JAX-WS	
JSTL	
JTA	
CDI	
Dependency Injection	

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 x 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}" />`

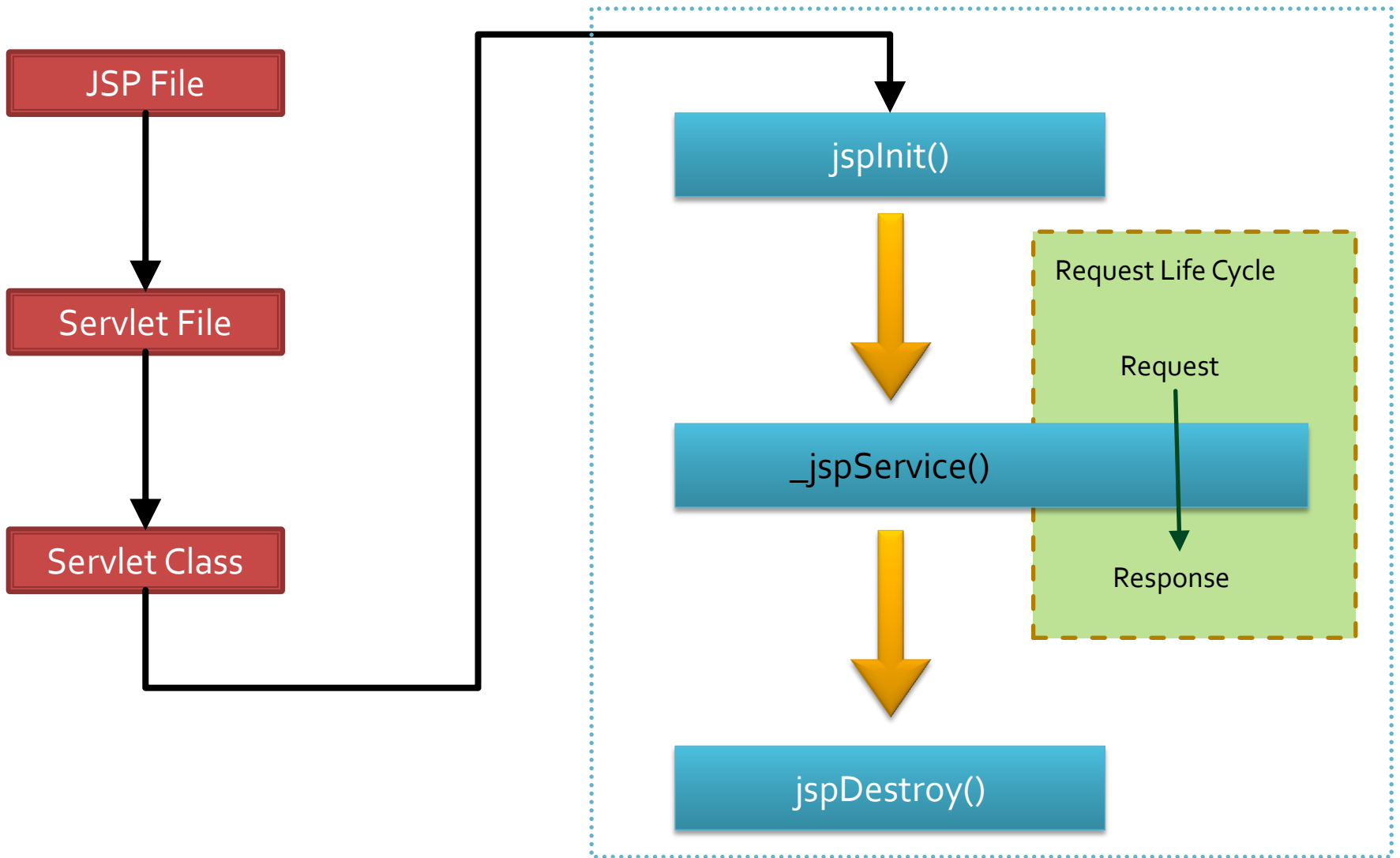
Automatické volání get/set

- `#{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);`

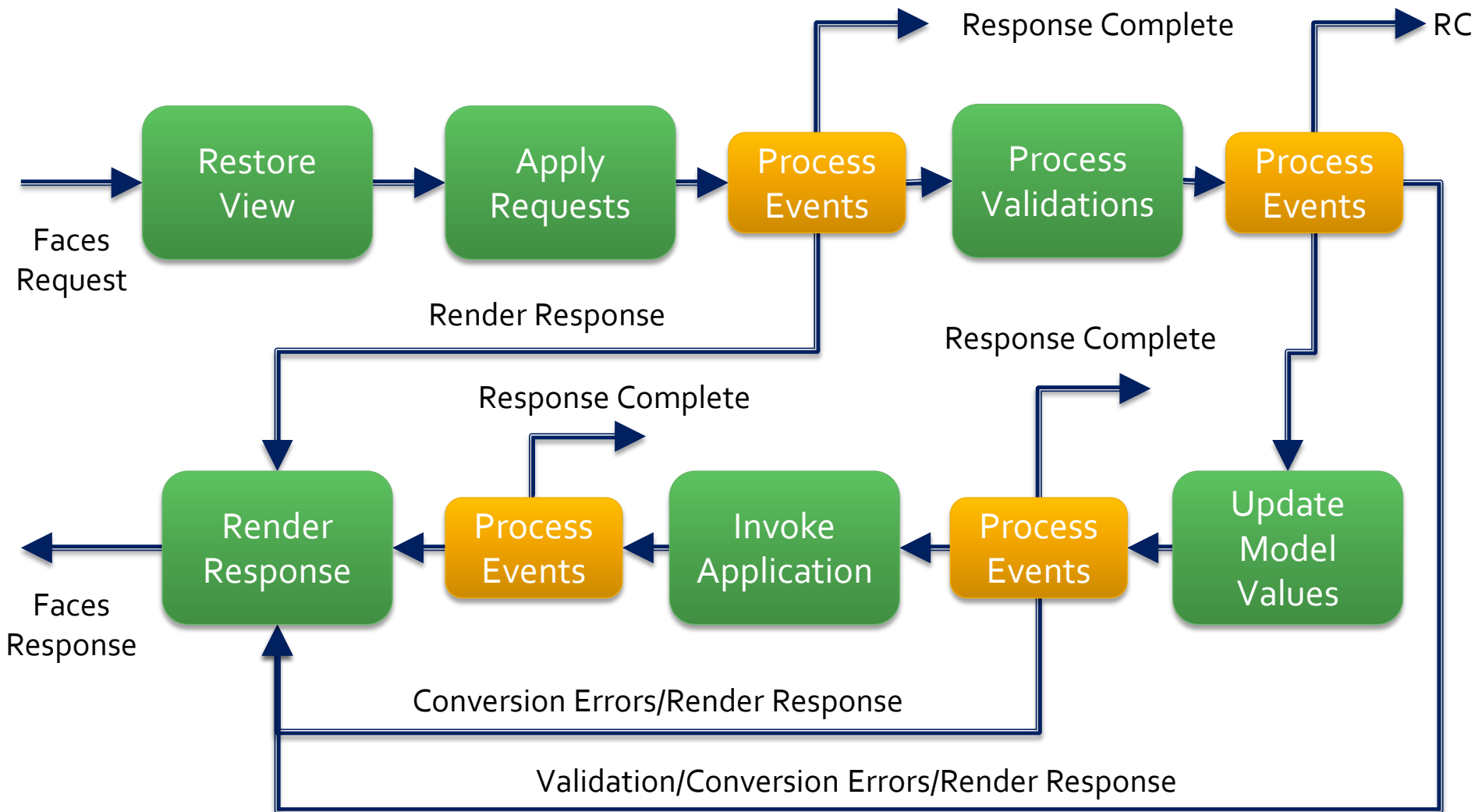
Opakování: Request/Response



Opakování: JSP Life Cycle



JSF Life Cycle



JSF MVC

Model

- Managed Bean

View

- JSF UI komponenty

Controller

- Faces Servlet

JSF MVC



PATŘÍ

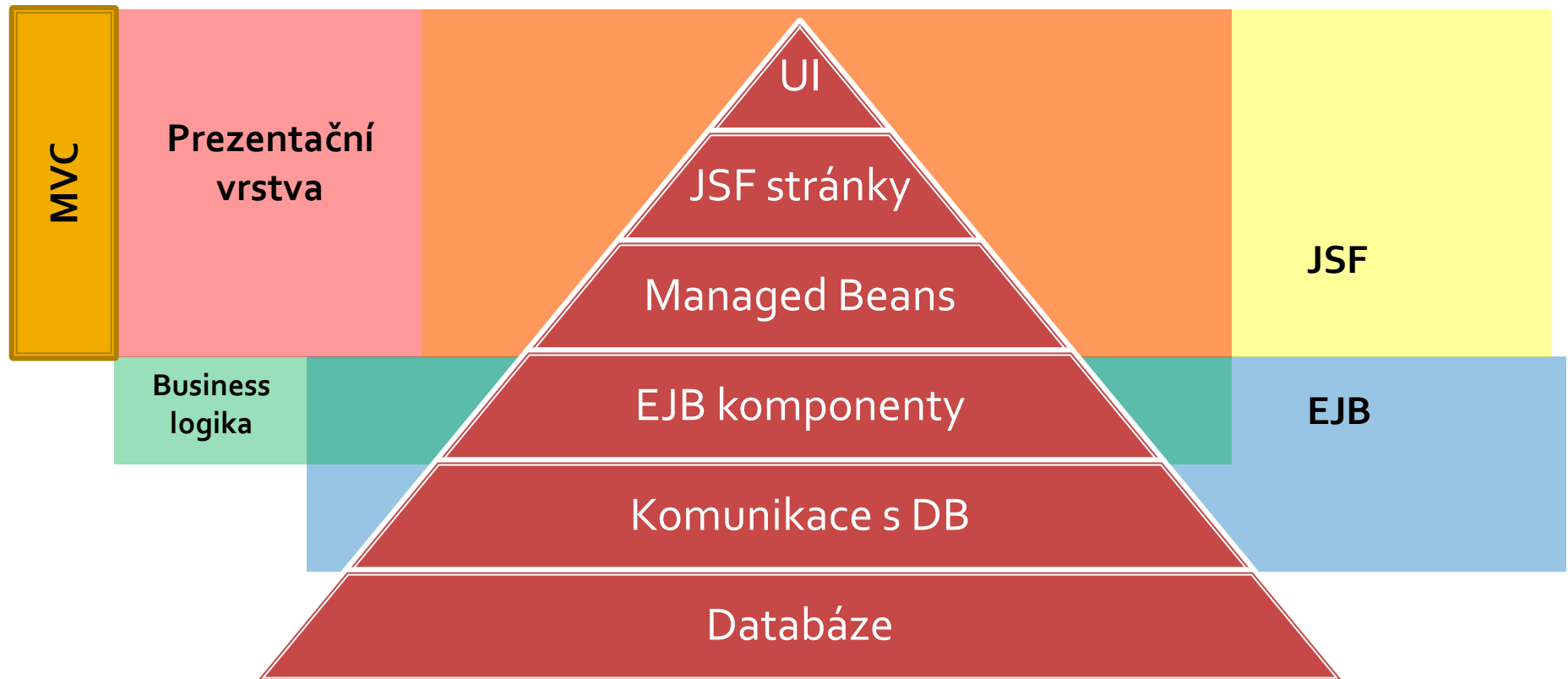
- Události
- Formátování výstupu
- Validace
- Navigace
- Lokalizace
- Zobrazování chyb
- Volání business logiky



NEPATŘÍ

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

JSF v architektuře Java EE



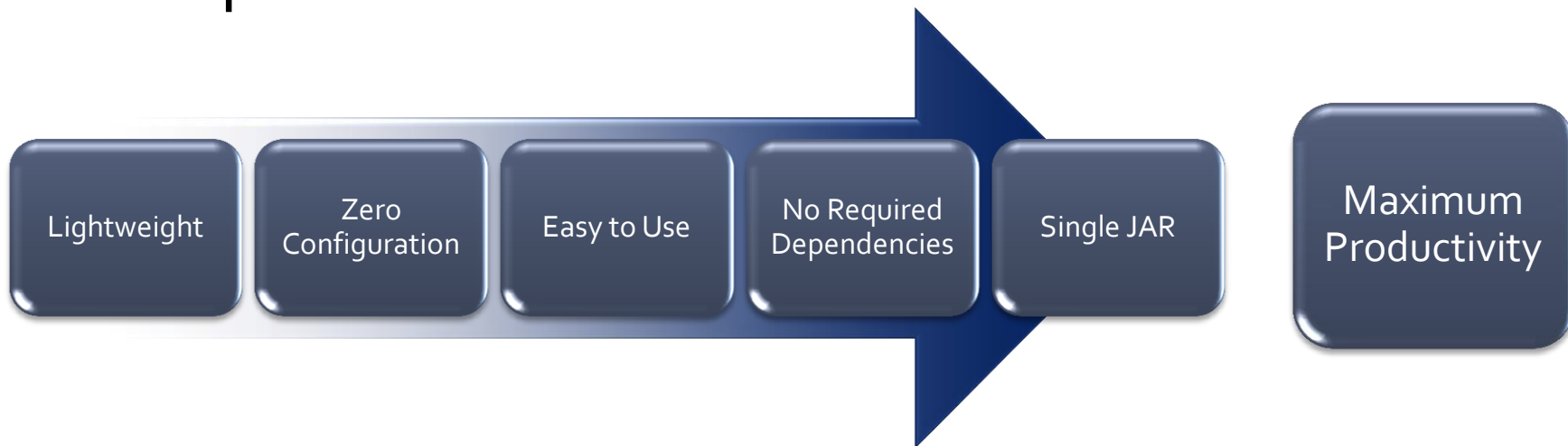
PrimeFaces

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



aristo



Components



Mobile



Push



Ext



Mock OS X

Ajax Core

- Basics
- Partial Processing
- Partial Submit
- Validations
- Fragment
- Selectors
- Polling
- RemoteCommand
- AjaxStatus
- SearchExpressions
- Events
- Selects
- Listeners
- Counter

Input

- AutoComplete
- BooleanButton
- BoolCheckbox
- Calendar
- CheckboxMenu
- ColorPicker
- Editor
- Inplace
- InputMask
- InputText
- InputTextarea
- Keyboard
- ManyButton
- ManyMenu
- ManyCheckbox
- MultiSelectListbox
- OneButton
- OneMenu
- OneListbox
- OneRadio
- Password
- Rating
- Spinner
- Slider

Button

- Button
- CommandButton
- CommandLink
- SplitButton

Data

- Carousel
- DataExporter
- DataList
- DataGrid
- DataTable
- GMap
- Mindmap
- PickList
- OrderList
- Ring
- Schedule
- TagCloud
- Tree
- Tree Horizontal
- TreeTable

DataTable - Dynamic Columns

Columns of datatable can be defined dynamically with p:columns component. Since columns are created on-the-fly it is easy to add/remove columns programmatically. Column template below is a white-space separated values that can be a combination of "model", "manufacturer", "color" and "year".

Template:

MODEL	MANUFACTURER	YEAR
73545958	BMW	1982
bebeb0f1	Audi	1982
31f7a3bf	Renault	1962
544de20c	Renault	1983
5bc4396	Volkswagen	2006
2d243624	BMW	1985
33b629a9	BMW	1983
48174628	Chrysler	1982
ed44b617	Audi	1969

Source

[datatableDynamicColumns.xhtml](#) [TableBean.java](#)

<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
 - aktuálně 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 implementací 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

```
@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);

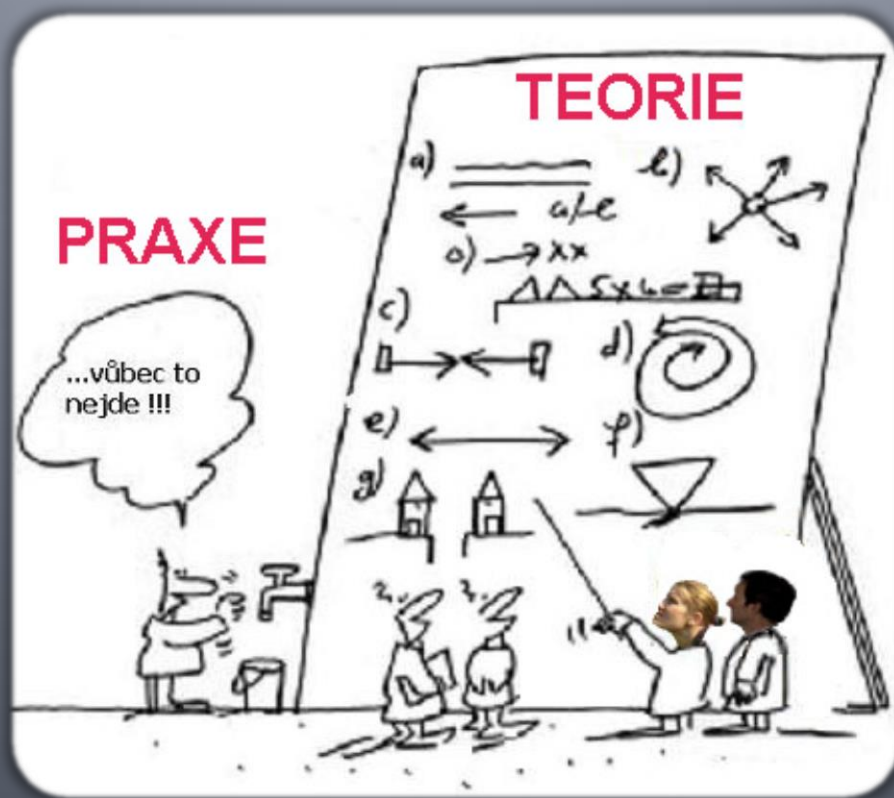
    Predicate predicateSensor =
        criteriaBuilder.equal(temperatureLogRoot.get("sensor"), sensor);

    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:dataTable id="users" var="u" value="#{users}" relation="#{users}"/>
<!-- <p:select event="rowSelect" listener="#{users.rowSelect}" update=":" />

<p:ajax event="rowEdit" listener="#{users.rowEdit}" update=":form:Data:messages"/>
<p:ajax event="rowEdit" listener="#{users.rowCancel}" update=":form:Data:messages"/>

<p:column headerText="#{msg['users.firstName']}">
<p:cellEditor>
<f:facet name="input">
<h:outputText value="#{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`