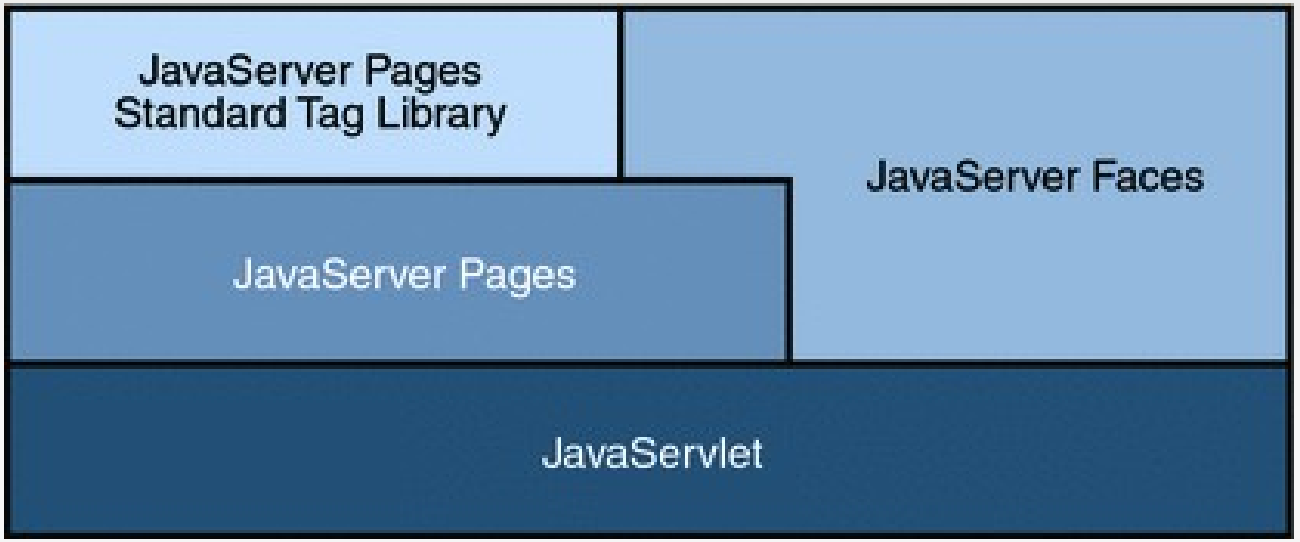
Lukáš Zapletal
liberix.cz

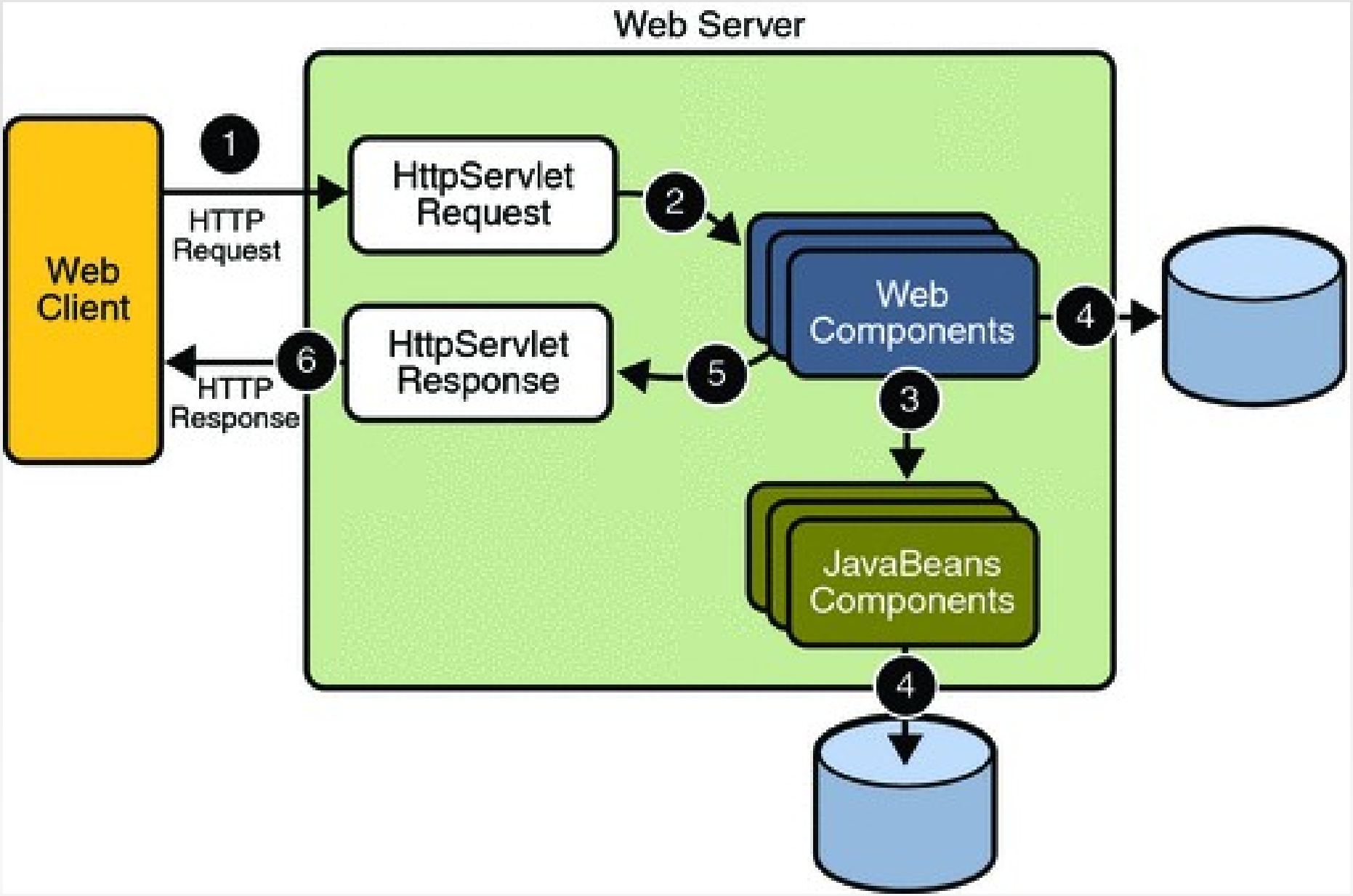
Platforma Java 2 Enterprise Edition
vývoj webových aplikací

J2EE - webové aplikace

- hlavní komponentou u webového stacku J2EE je Servlet
- nad touto technologií stojí JSP a rozšiřující knihovny (zejména nové značky)
- a dále nejnovější technologie JSF



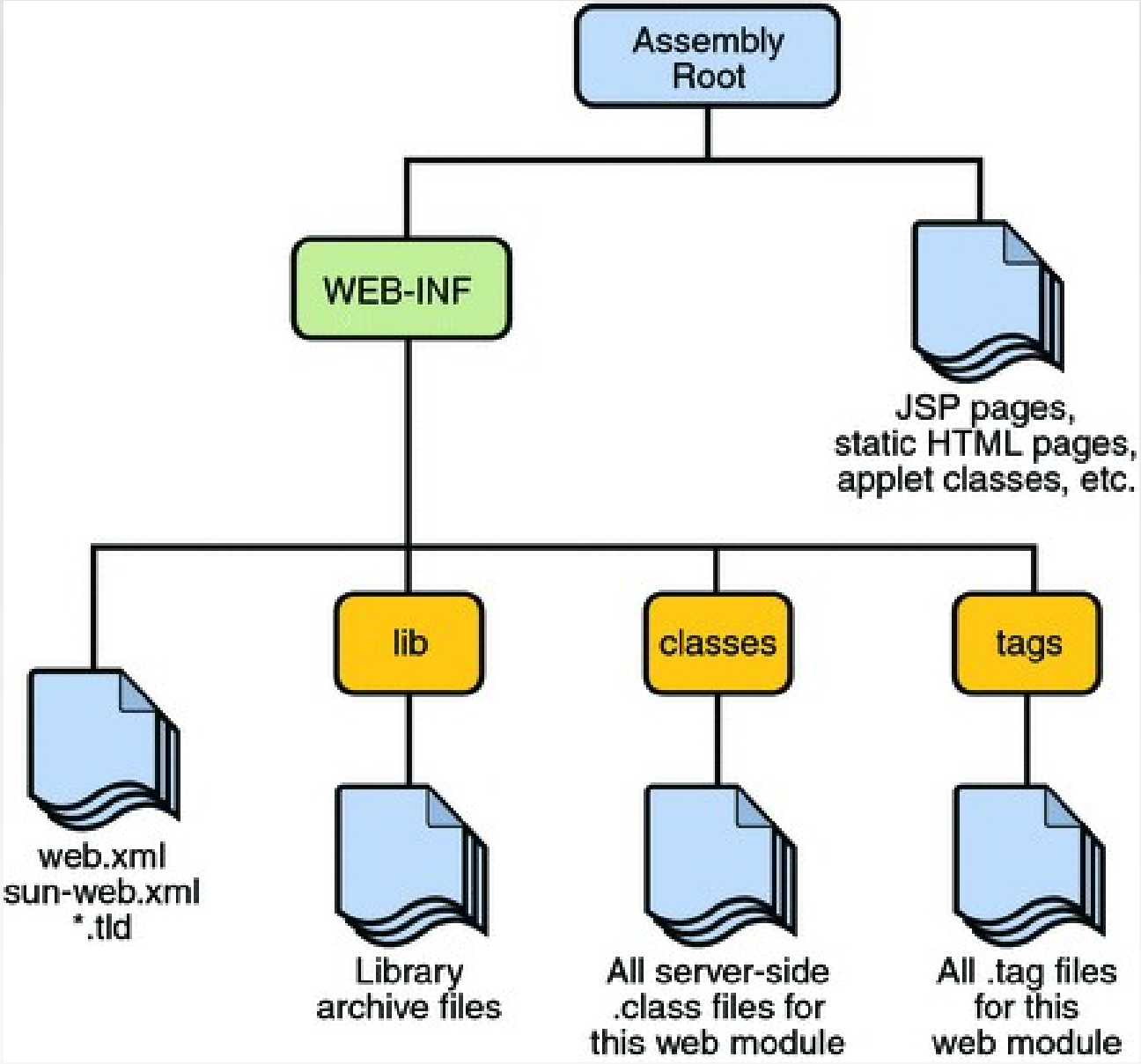
J2EE - webové aplikace



J2EE - webové aplikace

- nasazení probíhá pomocí WAR archivu
- jedná se o ZIP soubor s pevně danou předepsanou strukturou a příponou
- adresář WEB-INF obsahuje zkompilované třídy, knihovny, rozšiřující moduly a konfiguraci
- nejdůležitější je soubor **web.xml** - bez něj webová aplikace nemůže v J2EE fungovat
- nasazení: přes ant, pomocí IDE, přes webovou konzoli serveru nebo nástrojem (např. asadmin deploy aplikace.war)

J2EE - webové aplikace



J2EE - webové aplikace

- v konfiguraci (web.xml) se nastavuje:
  - kořenová adresa, např: http://host:port/context-root
  - welcome files (index.html, index.jsp)
  - konfigurační hodnoty aplikačního kontextu
  - chybové stránky
  - zdroje (resources)
    - servlety, filtry, naslouchače (kontext, session)
    - knihovny značek (taglibs) a naslouchače
    - managed beans

J2EE - Servlet

- servlet je komponenta uzpůsobená pro psaní serveru u request-response komunikaci
- HttpServlet je implementací pro HTTP
- baličky javax.servlet a javax.servlet.http
- každý servlet je namapován na určitou adresu či skupinu adres URL
- servlety se obvykle používají (používaly) pro psaní logiky aplikace, po provedené akci přesměřovaly tok na JSP stránku, která zobrazila výsledek

J2EE - Servlet

- pro vytvoření servletu přetížíme třídu HttpServlet a implementujeme jednu nebo více metod doAkce; kde Akce je:
  - get
  - delete
  - options
  - post
  - put
  - trace

J2EE - Servlet

```
public class BookDetailsServlet extends HttpServlet {
    ...
    public void doGet (HttpServletRequest request,
          HttpServletResponse response)
          throws ServletException, IOException {
      ...
      // set headers before accessing the Writer
      response.setContentType("text/html");
      response.setBufferSize(8192);
      PrintWriter out = response.getWriter();

      // then write the response
      out.println("<html>" +
          "<head><title>+
          messages.getString("TitleBookDescription")
          +</title></head>");

      /
      out.close();
   }
}
```

J2EE - Servlet

```
public class Dispatcher extends HttpServlet {

    public void doGet(HttpServletRequest request,
         HttpServletResponse response) {

        RequestDispatcher dispatcher = request.
            getRequestDispatcher("/template.jsp");
        if (dispatcher != null)
            dispatcher.forward(request, response);
    }

}
```

J2EE - Servlet

- životní cyklus servletu
  - metoda init(Config config)
  - zpracovávání požadavků
  - metoda destroy()
- instancí jednoho servletu může být několik

J2EE - Servlet

- webová aplikace má několik kontextů
- kontext = místo (jmenný prostor) pro objekty
- kontexty:
  - ServletContext - pro různé konexe do databáze
  - SessionContext - proměnné sezení

```
public class CashierServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {

      HttpSession session = request.getSession();
      ShoppingCart cart =
        (ShoppingCart)session.
          getAttribute("cart");

      double total = cart.getTotal();
```

J2EE - Servlet

- Servlet musí být napsán vícevláknově
- jeden servlet může zpracovávat více požadavků!
- to platí i pro objekty, které se ukládají do kontextů (včetně session kontextu!)
- pokud to není možné zajistit, je nutné implementovat rozhraní SingleThreadModel a kontejner zajistí řazení požadavků do fronty

J2EE - Filter

```
public final class HitCounterFilter implements Filter {
    private FilterConfig filterConfig = null;

    public void init(FilterConfig filterConfig)
        throws ServletException {
        this.filterConfig = filterConfig;
    }
    public void destroy() {
        this.filterConfig = null;
    }
    public void doFilter(ServletRequest request,
        ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        if (filterConfig == null)
            return;
        StringWriter sw = new StringWriter();
        PrintWriter writer = new PrintWriter(sw);
        Counter counter = (Counter)filterConfig.
            getServletContext().
            getAttribute("hitCounter");
        writer.println("The number of hits is: " +
            counter.incCounter());
        // Log the resulting string
        writer.flush();
        System.out.println(sw.getBuffer().toString());
        ...
        chain.doFilter(request, wrapper);
        ...
    }
}
```

J2EE - JSP

- JSP = Java Server Pages
- technologie pro vývoj JSP stránek - textově orientovaných dokumentů, kde lze mixovat Javu a text (HTML, XHTML, ...)
- kromě JSP značek obsahuje jazyk EL (Expression Language), pomocí kterého lze snadno přistupovat k objektům
- poskytuje mechanismy pro další rozšiřování

J2EE - JSP

- přípony:
- .jsp, .jspx - stránka JSP
- .jspf - fragment
- .tag - značka

J2EE - JSP

- java kód píšeme do <% ... %>, <%! ... <%= ...
- speciální direktivy do <%@ ... %>
- značky (tags) se vkládají jako normální XML, ale v odděleném jmenném prostoru
- abychom vůbec nemuseli psát java kód, je zde EL, který se zapisuje pomocí ${ ... }
- veškeré značky se nakonec přeloží do java třídy, která se zkompiluje na Servlet (JSP stránky tedy nejsou pomalejší než Servlety)

J2EE - JSP

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="/functions" prefix="f" %>
<html>
<head><title>Localized Dates</title></head>
<body bgcolor="white">
<jsp:useBean id="locales" scope="application"
   class="mypkg.MyLocales"/>

<form name="localeForm" action="index.jsp" method="post">
<c:set var="selectedLocaleString" value="${param.locale}" />
<c:set var="selectedFlag"
    value="${!empty selectedLocaleString}" />
<b>Locale:</b>
<select name=locale>
<c:forEach var="localeString" items="${locales.localeNames}" >
<c:choose>
   <c:when test="${selectedFlag}">
      <c:choose>
         <c:when
             test="${f:equals(selectedLocaleString, localeString)}" >
            <option selected>${localeString}</option>
         </c:when>
      </c:choose>
   </c:when>
...
```

J2EE - JSP

- proměnné, které používáme na stránce JSP, mají několik rozsahů
  - page
  - request
  - session
  - application

J2EE - JSP (příklady EL)

- ${sessionScope.cart.numberOfItems}
- ${param['mycom.productId']}
- ${customer.age + 20}
- ${customer.orders[1]}
- technologie JSF používá stejný jazyk, který má ale několik odlišností
- proto je u JSF EL jiná syntaxe: #{ ... }
- JSP EL a JSF EL lze kombinovat

J2EE - JSP

- znovupoužívání obsahu:
  - značka <%@ include file=”header.jspf” %> - vkládání probíhá při převodu na Servlet
  - značka <jsp:include page=”response.jsp”/> - vkládání probíhá za běhu
- přesměřování běhu:
  <jsp:forward page="/main.jsp" />

J2EE - JSP

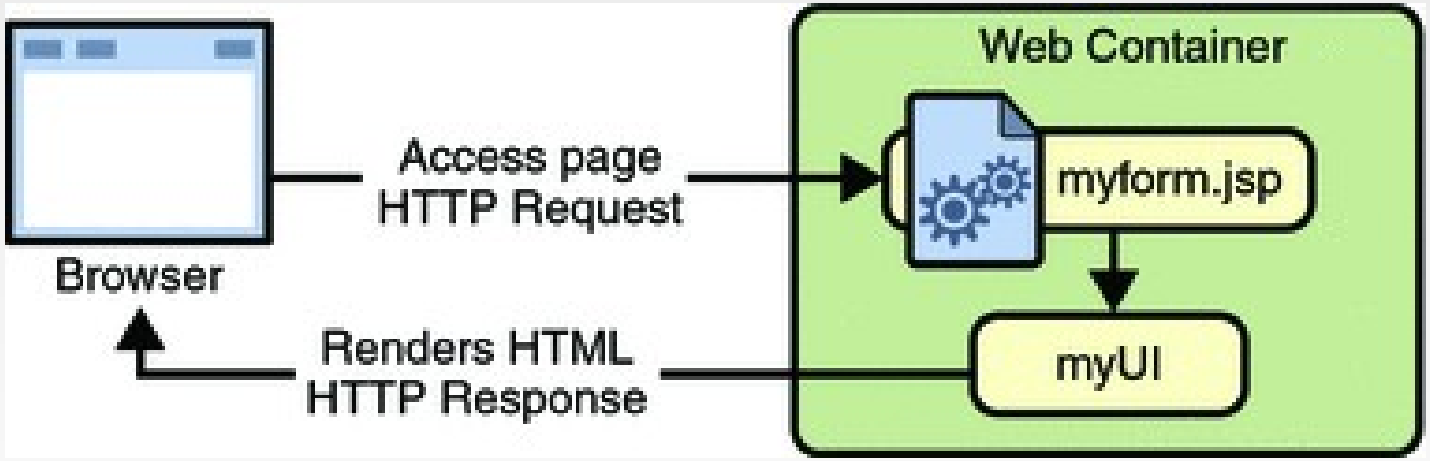
- JSTL - rozšiřující značky, které zpřehledňují JSP kód (od J2EE 5.0 přímou součástí)
  - core - proměnné, tok, URL pomůcky a utility
  - XML - transformace
  - I18n - locales, message formating, čísla
  - Database - značky pro SQL
  - Pomocné funkce

J2EE - JSF

- nejnovější technologie Java Server Faces
- komponentově orientovaný vývoj webových aplikací
- už nepracujeme s request-response, ale s komponentami
- JSF má modulární architekturu a je nezávislá na prezentační vrstvě

J2EE - JSF

- JSF se skládá z
  - rozhraní (API) pro tvorbu a používání UI komponent, posílání zpráv, validace, konverze, navigace, i18n
  - JSP speciální značky pro tvorbu prezentační vrstvy



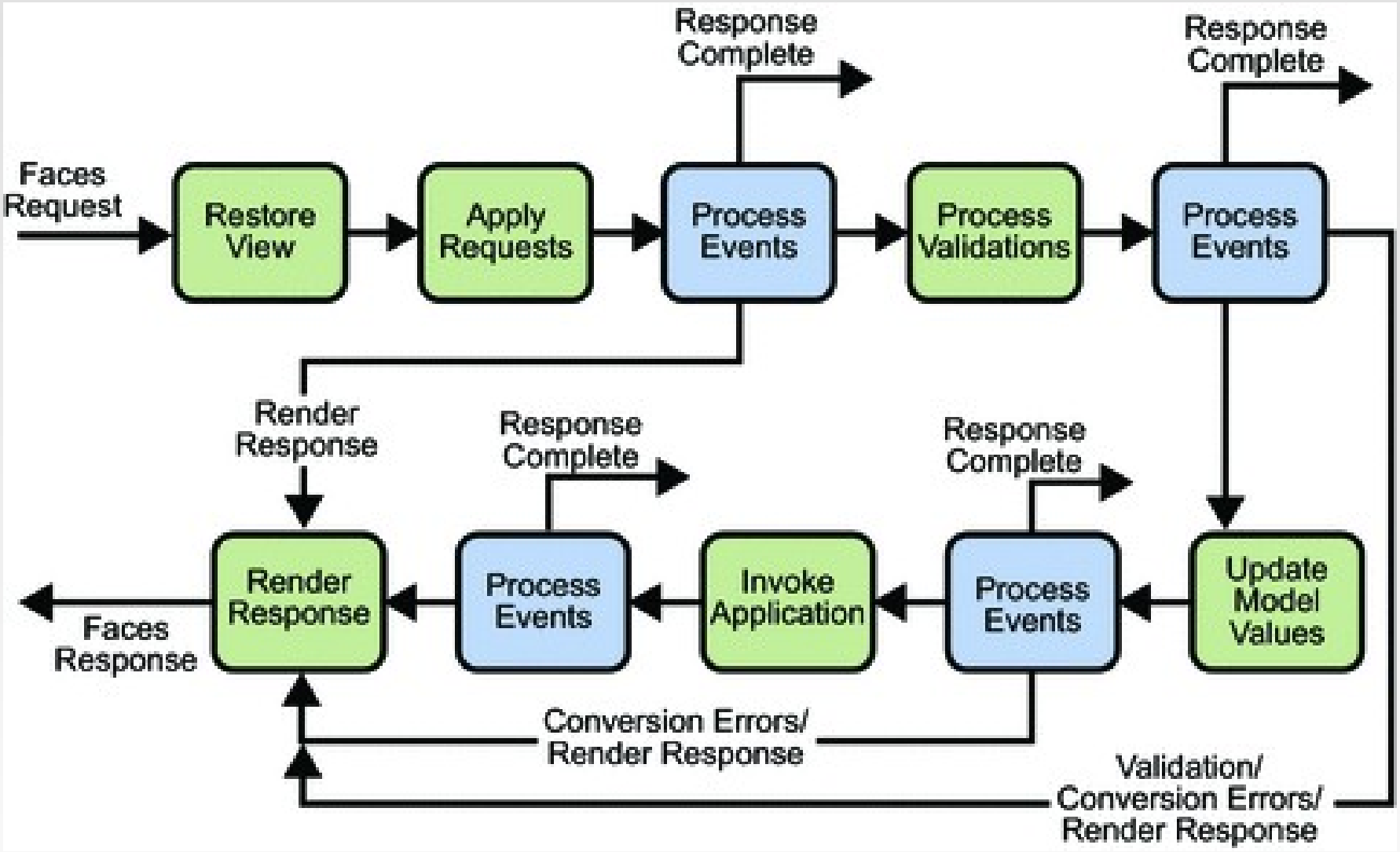
J2EE - JSF

- základ tvoří UI komponenty, které zapouzřují data a chování prezentačních prvků
- UIIInput, UIPanel, UICommand, UISelectItem
- jejich vykreslování (rendering) je nezávislé na samotných komponentách (rendering do HTML, XUL a podobně)
- například HTML rendering kit nabízí vykreslování HTML komponent (inputText, panelGroup, form, commandButton...)

J2EE - JSF

- JSF poskytuje architekturu pro
- zprávy a posílání zpráv (akce)
- konverzi a validaci dat (validators)
- navigaci (navigařní pravidla - v XML)
- vlastní datové prvky (datové beany, tzv. managed beans)
- třídy stojící za stránkami (backing beans - obsahují logiku prezentační vrstvy)

J2EE - JSF


Response Complete
Response Complete
Faces Request
Restore View
Apply Requests
Process Events
Process Validations
Process Events
Render Response
Response Complete
Response Complete
Render Response
Process Events
Invoke Application
Process Events
Update Model Values
Faces Response
Conversion Errors/ Render Response
Validation/ Conversion Errors/ Render Response

J2EE - JSF

```
<h:outputText value="#{cashier.shipDate}">
    <f:convertDateTime dateStyle="full" />
</h:outputText>

...

protected Date shipDate;

public Date getShipDate() {
    return this.shipDate;
}
public void setShipDate(Date shipDate) {
    this.shipDate = shipDate;
}
```

J2EE - JSF

- JSF aplikace jsou obyčejné J2EE webové aplikace
- k aplikaci je nutné přibalit pouze několik knihoven (JSF API, implementace)
- konfigurace (faces-config.xml)
- nastavení webové aplikace (web.xml)

J2EE - další webové frameworky

- Facelets - view technologie pro JSF
- Struts - praotec JSF
- Tapestry - komplexní komponentový framework
- Stripes - jednoduchý RAD
- Velocity, Freemaker, Barracuda - templates
- Cocoon - robustní XML framework
- GWT, DWR - AJAX
- Turbine, Tiles, Shale a další a další...