

JSP

Technický úvod

- Co je JSP stránka? Jedná se o servlet, který nevytváříme přímo, ale je vytvořen kontejnerem z námi vytvořené šablony. JSP se zkompiluje na Servlet při prvním požadavku klienta.
- V Apache Tomcat se můžete podívat na vytvořený servlet ... přejděte na příslušnou JSP stránku a poté se podívejte do adresáře [apache-tomcat]/work, kde se nachází cache Tomcatu.
- JSP stránky jsou jednou z mnoha možností jak vytvořit UI (user interface) v Javě. Další populární technologie:
 - Apache Velocity
 - JSF
 - GWT

Scriptlety

- V JSP stránkách se kdykoli můžete „přepnout“ do Javy a obohatit HTML kód o dynamické prvky. Takovým částem kódu se říká scriptlety a v dnešní době se moc často nepoužívají, ale je možné pomocí nich udělat cokoli a občas je jejich použití vhodné.

`<%@page import="java.util.Date"%>`

page directive
pro import balíčku

`<%@page import="java.text.SimpleDateFormat"%>`

`<%! private int count = 0; %>`

Declaration pro deklarování
atributů anebo metod

Počet přístupů na obrazovku: `<% out.print(++count); %>`

scriptlet

(C) 2007 - `<%= new SimpleDateFormat("yyyy").format(new Date()) %>`

expression

Implicitní objekty

- Proměnné out se říká implicitní objekt. V JSP stránkách jsou implicitní objekty pro zjednodušení psaní kódu (toho samého bychom docílili pomocí `request.getWriter()`)
- Existují další implicitní objekty:

① <code>application : ServletContext</code>	←	<code>request.getServletContext()</code>
① <code>config : ServletConfig</code>	←	<code>getServletConfig()</code>
① <code>out : JspWriter</code>	←	<code>request.getWriter()</code>
① <code>page : Object</code>		
① <code>pageContext : PageContext</code>		
① <code>request : HttpServletRequest</code>		
① <code>response : HttpServletResponse</code>		
① <code>session : HttpSession</code>	←	<code>request.getSession()</code>

Page scope

- V JSP existuje oproti klasickým třem „nástěnkám“ (scope) – request, session, application ještě jeden: **page scope**.
- Atribut uložený v page scope má dobu platnosti po dobu vytvoření JSP stránky.

EL (Expression Language)

- Expression Language slouží k získání dynamické hodnoty proměnné / vlastnosti bez použití scriptletu.
- Níže je JSP expression a pod tím alternativní zápis pomocí EL:

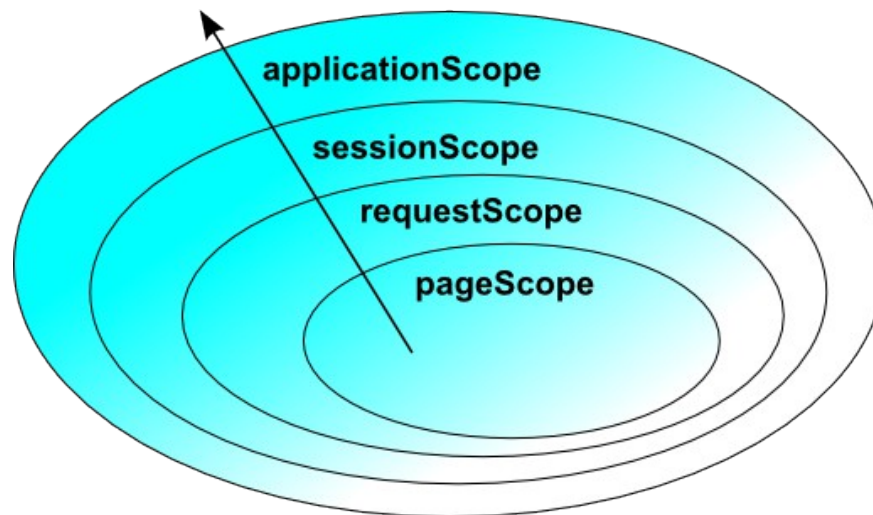
`<%= ((foo.Person) request.getAttribute("person")).getDog().getName() %>`

`${person.dog.name}`

- Syntaxe výrazu v EL je následující:
`${objekt.vlastnost1.vlastnost2}`
- EL slouží ke snadnému procházení objektů a jejich vlastností a k provádění jednoduchých operací nad nimi, ale není to programovací jazyk! Výrazy EL mohou být libovolně kombinovány se statickým textem.

Identifikátory v EL

- Výraz EL začíná identifikátorem – tj. buď implicitním objektem (pageScope, requestScope, sessionScope, applicationScope, param, paramValues, header, headerValues, cookie, initParam, pageContext – neplést s implicitními objekty v JSP), nebo názvem atributu (v oborech platnosti page, request, session nebo application).
- Pokud identifikátor není názvem implicitního objektu, považuje se za název atributu. Takový atribut se poté hledá nejdříve v oboru platnosti page, dále (není-li nalezena) postupně v request, v session a nakonec v application – je vrácena hodnota atributu nebo null.



Implicitní objekty I.

- **pageScope** – mapa obsahující názvy a hodnoty atributů v oboru platnosti page.
- **requestScope** – mapa obsahující názvy a hodnoty atributů v oboru platnosti request.

Scriptlet:

```
<%@page import="cz.java.skoleni.eshop.entity.Item"%>
<% Item item = (Item)request.getAttribute("item"); %>
<%= item.getName() %>
```

EL: `${requestScope.item.name}`

- **sessionScope** – mapa obsahující názvy a hodnoty atributů v oboru platnosti session.
- **applicationScope** – mapa obsahující názvy a hodnoty atributů v oboru platnosti application.

Implicitní objekty II.

- **header** – mapa obsahující názvy a hodnoty HTTP hlaviček předávaných v requestu.
- **headerValues** – mapa obsahující hodnoty HTTP hlaviček v requestu jako pole řetězců.

Scriptlet: `<%= request.getHeader("referer") %>`

EL: `${header.referer}`

- **param** – mapa obsahující názvy a hodnoty parametrů předávaných v requestu.
- **paramValues** – mapa obsahující hodnoty parametrů v requestu jako pole řetězců.

Scriptlet: `<%= request.getParameter("title") %>`

EL: `${param.title}`

Implicitní objekty III.

- **cookie** – mapa obsahující názvy a hodnoty cookies (instance třídy `Cookie`).
- **initParam** – mapa obsahující názvy a hodnoty inicializačních parametrů webové aplikace, které jsou nastaveny ve `/WEB-INF/web.xml`
- **pageContext** – instance třídy `PageContext` (kontext stránky), obsahuje vlastnosti pro přístup ke všem implicitním objektům JSP stránky.

Scriptlet: `<%= request.getLocale() %>`

EL: `${pageContext.request.locale}`

Zpřístupnění vlastností objektu

- Pro zpřístupnění vlastností objektu se používá tečková notace:
 - `${person.firstName}` (ve třídě musí být definována přístupová metoda `getFirstName()`)
- Pokud je atributem pole nebo kolekce, jednotlivé prvky lze zpřístupnit pomocí hranatých závorek (u map lze použít kromě `[]` i tečku a název klíče):
 - `${pole[3]}`
 - `${telefonniSeznam["774912047"]}`
- Accessory lze aplikovat rekurzivně:
 - `${persons["milan"].address.city}`
- Pokud některá část výrazu vrací null, celý výraz je vyhodnocen jako null, není vyvolávána žádná výjimka (i v případě, že jsou na nullovém výrazu volány jeho další možné vlastnosti).

Operátory

- **Aritmetické:** +, -, *, / (div), % (mod),
- **relační:** == (eq), != (ne), < (lt), > (gt), <= (le), >= (ge),
- **logické:** && (and), || (or), ! (not),
- **validační operátor** empty (dotaz na nullovou hodnotu nebo prázdnot pole, kolekce, řetězec nulové délky).
- **Příklady:**
 - `${polozka.cena * (1 + danovaSazba[uzivatel.adresa.psc])}`,
 - `${(x >= min) && (x <= max)}`

Podmínkový operátor ?:

- EL také zná podmínkový operátor ?:, který efektivně využijete například v následující situaci:

```
<select name="category">
  <option value="1" ${product.category == '1' ? 'selected' : ''}>Dogs</option>
  <option value="2" ${product.category == '2' ? 'selected' : ''}>Cats</option>
  <option value="5" ${product.category == '5' ? 'selected' : ''}>Others</option>
</select>
```

- Nebo ve spojení s JSTL (viz. dále):

```
<select name="category">
  <c:forEach items="${categories}" var="category">
    <option value="${category.id}" ${product.category == category.id ? 'selected' : ''}>
      ${category.name}
    </option>
  </c:forEach>
</select>
```

Knihovny tagů JSTL I.

- JSTL = JavaServer Pages Standard Tag Library
- Jedná se o soubor často používaných knihoven tagů. Tyto tagy jsou alternativou ke scriptletům.
- Do projektu je potřeba přidat knihovnu JSTL:

```
<dependency>  
  <groupId>jstl</groupId>  
  <artifactId>jstl</artifactId>  
  <version>1.2</version>  
</dependency>
```

- Knihovna tagů se musí importovat do JSP stránky, poté lze používat její tagy:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
  
<c:out value="${person.name}" />
```

Knihovny tagů JSTL II.

- Základní knihovny tagů:
 - Core library (c:),
 - Formatting library (fmt:),
 - SQL library (sql:),
 - XML library (x:),
 - JSTL functions (fn:),
 - ...
- Nejčastěji se používá Core knihovna.
- Knihovny s příponou "rt" používají namísto EL historicky starší JSP expression bloky (`<%=...%>`),
- Pro použití nejnovějších JSTL knihoven s podporou EL výrazů je doporučeno importovat knihovny:
 - `uri="http://java.sun.com/jsp/..."`

Nastavování atributů

- `<c:set>` - možnost (pře)nastavení hodnoty atributu:

```
<c:set var="nazev" scope="session" value="hodnota" />
```

```
<c:set var="nazev" scope="page">hodnota</c:set>
```

- `<c:remove>` - odebrání atributu:

```
<c:remove var="nazev" scope="session"/>
```


Výpis atributů

- `<c:out>` - výpis hodnoty expression výrazu:

```
<c:out value="expression" default="value" escapeXml="boolean" />
```

- Vypíše hodnotu EL. Pokud je výsledkem null, vypíše se hodnota volitelně uvedená v default. Atribut `escapeXml` při výchozím nastavení na `true` převádí znaky se speciálním významem v HTML/XML (např. `<`, `>`, `&`) na odpovídající XML entity (`<`, `>`, `&`).

Procházení polí / kolekcí I.

- `<c:foreach>` - slouží pro iterování polí a kolekcí:

```
<table>
  <c:forEach var="item" items="${items}">
    <tr>
      <td>${item.name}</td>
      <td>${item.price}</td>
    </tr>
  </c:forEach>
</table>
```

A grey arrow points from the variable `item` in the `var="item"` attribute of the `<c:forEach>` tag to the `item` property access in the `<td>${item.name}</td>` expression.

Procházení polí / kolekcí II.

- Konstrukci s atributem items lze použít i pro pole objektů nebo pole primitivních typů. Primitivní typy jsou automaticky zabaleny do obalovacích tříd.
- Pro iteraci přes index můžeme použít následující speciální verzi forEach pro pole:

```
<c:forEach var="i" begin="0" end="9" step="1">  
  <c:out value="${pole[i]}" />  
</c:forEach>
```

Procházení polí / kolekcí III.

- Volitelný atribut `varStatus` slouží pro vytvoření proměnné (instance třídy `LoopTagStatus`) s těmito dostupnými vlastnostmi:
 - **current** – aktuální položka z procházené kolekce (pole),
 - **index** – index aktuální iterace (od 0),
 - **count** – číslo aktuální iterace (od 1),
 - **first** – true při první iteraci,
 - **last** – true při poslední iteraci,
 - **begin, end, step** – hodnoty atributů `begin`, `end`, `step`.

Podmínkové větvení

- `<c:if>` - jednoduché podmínkové větvení (neobsahuje něco jako „else if“ nebo „else“):

```
<c:if test="expression vracejici boolean hodnotu">
```

```
...
```

```
</c:if>
```

- `<c:choose>` - výkonnější větvení:

```
<c:choose>
```

```
<c:when test="boolean hodnota">...</c:when>
```

```
<c:when test="boolean hodnota 2">...</c:when>
```

```
<c:otherwise>...</c:otherwise>
```

```
</c:choose>
```

- Bude provedena první vyhovující větev `when`, anebo `otherwise`.

Příklad na podmínkové větvení

```
<c:choose>
```

```
  <c:when test="${empty userOrders}">
```

```
    <p>V systému nejsou žádné objednávky!</p>
```

```
  </c:when>
```

```
  <c:otherwise>
```

```
    <c:forEach var="order" items="${userOrders}">
```

```
      <div>
```

```
        <c:out value="${order.name}" />
```

```
      </div>
```

```
    </c:forEach>
```

```
  </c:otherwise>
```

```
</c:choose>
```

Generování URL odkazu I.

- `<c:url>` - generování URL odkazu:

```
<c:url value="adresa" />
```

- Automaticky připojí jméno servlet kontextu, pokud zadaná URL adresa začíná "/". Pokud adresa nezačíná lomítkem, je považována za relativní, připojení jména kontextu pak není nutné.
- Vhodné použití `<c:url>` je pro tvorbu URL odkazů v menu, headeru a footeru, protože tyto části stránky se vyskytují na všech stránkách webové aplikace a tudíž musí být absolutní.
- Jedno z typů použití `<c:url>`:

```
<a href="<c:url value=' /items.html ' />">Items</a>
```

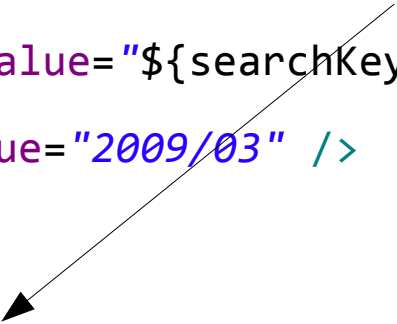


Poznámka: Na některých Java EE serverech nesmí být vnořené uvozovky, tak je možné použít kombinaci uvozovek a apostrofů jako v tomto příkladu

Generování URL odkazu II.

- Pokud je atributem var zadáno jméno proměnné, výsledné URL se nevypíše na výstup, ale uloží se do této proměnné:

```
<c:url value="/content/search.html" var="searchUrl">  
  <c:param name="keyword" value="${searchKeyword}" />  
  <c:param name="month" value="2009/03" />  
</c:url>  
  
<a href="<c:out value='${searchUrl}' />">Hledat</a>
```



- Vhodné při dynamičtější tvorbě URL. Explicitnější než:

```
<a href="<c:out  
  value='/content/search.html?keyword=${searchKeyword}&month=2009/03' />">  
  Hledat  
</a>
```


Knihovna fmt

- Obsahuje tagy, které podporují lokalizaci textů, zobrazování a parsování čísel, data a času.
- Lokalizace dat je ovlivňována národnostním nastavením (tzv. "locale" – kódem jazyka a země) a časovou zónou.
- Locale ovlivňuje výpis čísel, měny, data a času. Pokud je území specifikované pomocí locale příliš velké (např. australský kontinent), lze provádět další upřesnění zobrazování dat na základě časové zóny.
- Výchozí locale a časovou zónu získá JVM na základě interakce s operačním systémem, kde je nainstalovaná. Toto není ideální pro webové aplikace. Proto knihovna fmt umožňuje změnit locale.

Nastavení locale a časové zóny

- `<fmt:setLocale>` - nastavení locale zadáním řetězce s locale (např. "en", "cs", "en_US", "cs_CZ") do atributu `value`. Lze poskytnout instanci třídy `Locale`, která tento řetězec obsahuje. Platnost locale lze nastavit pro různé obory viditelnosti (`page`, `request`, `session`, `application`).
- `<fmt:setTimeZone>` - nastavení časové zóny zadáním řetězce (dostupné identifikátory zón lze získat např. pomocí `TimeZone.getAvailableIDs()`, neexistuje jednotný standard pro pojmenování časových zón), nebo poskytnutím instance třídy `TimeZone`, která tento řetězec obsahuje.

Formátování data a času

- `<fmt:formatDate value="instance Date" type="time|date|both" dateStyle="default|short|medium|long|full" timeStyle="default|short|medium|long|full" pattern="vlastní formátovací vzor"/>`

- Vypíše datum a čas na výstup, popř. jej uloží do specifikované proměnné. Pokud je uveden atribut `pattern`, ignoruje se výpis podle národnostního nastavení, výpis se pak provádí vždy v daném formátu (viz. dokumentace třídy `SimpleDateFormat`).

```
<fmt:formatDate value="${dateOfInsert}"  
  type="both" pattern="d.M.yyyy, HH:mm:ss" />
```

- `<fmt:parseDate ... />`
 - Generuje instanci třídy `Date`, která bude vytvořena ze zadaného řetězce.

Formátování čísel

- `<fmt:formatNumber value="číslo" type="number|currency|percentage" pattern="vlastní formátovací vzor" currencyCode="kód měny" currencySymbol="symbol měny" maxIntegerDigits="max. počet míst před desetinnou čárkou" maxFractionDigits="max. počet desetinných míst" groupingUsed="oddělovat po třech"/>`
 - Vypíše číslo na výstup, popř. jej uloží do specifikované proměnné. Pokud je uveden atribut `pattern`, ignoruje se výpis podle národnostního nastavení, výpis se pak provádí vždy v daném formátu (viz. dokumentace třídy `DecimalFormat`).

```
<fmt:formatNumber value="${product.price}"  
  type="currency" pattern="#,##0.00 Kč" />
```

- `<fmt:parseNumber ... />`
 - Generuje instanci třídy `Number`, která bude vytvořena ze zadaného řetězce.

Lokalizace textů

- 1) Je třeba vytvořit Resource Bundle – množinu překladových souborů s texty pro různé locales v některém balíčku aplikace. Jedná se o properties soubory.
- 2) `<fmt:setBundle basename="jméno bundlu bez lokalizačních přípon" scope="obor viditelnosti, pro který bude bundle nastaven|scope proměnné" var="název proměnné"/>`
 - Nastavuje resource bundle s překladovými texty, které lze zobrazovat pomocí tagu `<fmt:message/>`. Pokud je zadáno jméno proměnné, bundle se nenastaví jako výchozí, ale odkaz na něj se uloží do této proměnné. Na proměnnou bundlu se lze odkazovat v atributu "bundle" tagu "message".

```
<fmt:message key="klíč textu" bundle="${proměnná s bundlem}"/>
```

Příklad lokalizace textů

- Text v budlu messages_en.properties:

```
index.greetings=Hello, {0}
```

- Lokalizace v JSP stránce:

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<fmt:setBundle basename="messages"/>
<fmt:message key="index.greetings">
    <fmt:param value="${user.fullName}"/>
</fmt:message>
```

- Jak na dynamické přepínání jazyků? Vytvořte Servlet, do kterého se předá parametr „locale“ s hodnotou zvoleného jazyku. Poté proveďte přesměrování na stránku, která ho zavolala (jedná se o stránku, která je v header hlavičce s názvem „referer“).

JSTL functions

- Velice užitečnou knihovnou tagů je také knihovna JSTL functions:

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
```

- Má několik užitečných metod, jejichž použití je následující:

- Metoda length vrátí počet prvků v kolekci / poli:

```
${fn:length(pole)}
```

- Plný seznam funkcí včetně příkladů jejich použití:

- <http://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/fn/tld-summary.html>

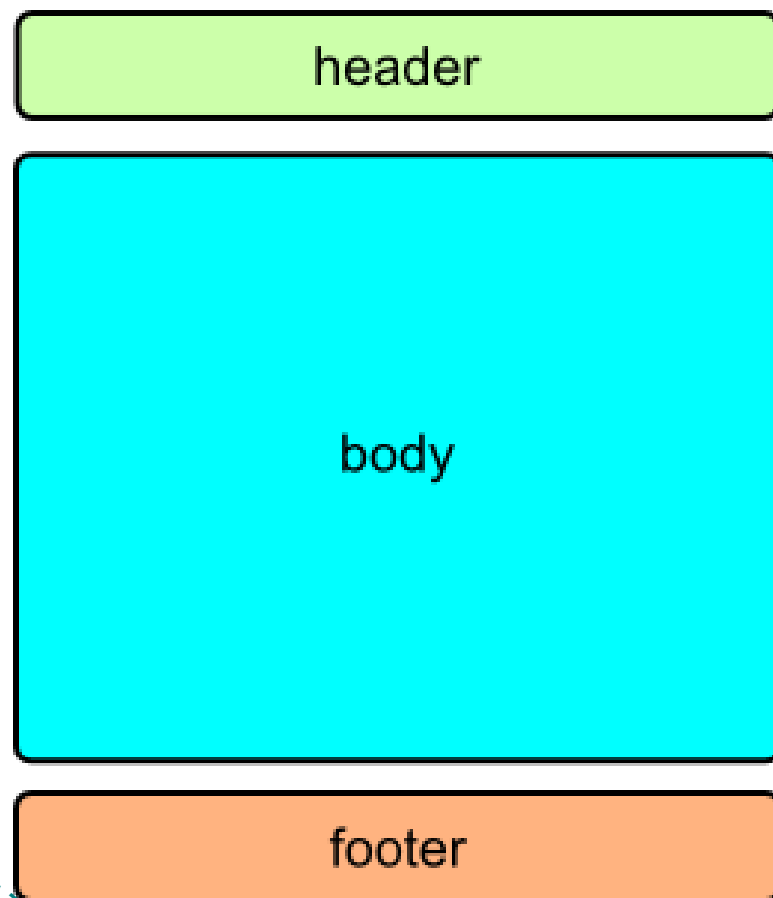
JSP include

- V každé JSP stránce je spousta informací (typicky v záhlaví a zápatí), které se v jednotlivých JSP stránkách opakují.
- Tyto informace je možné uložit do externích JSP souborů a ty includovat:

```
<jsp:include page="layout/footer.jsp" />
```

- Je možné také includované stránce poslat dynamické parametry:

```
<jsp:include page="layout/header.jsp">  
  <jsp:param name="title" value="items" />  
</jsp:include>
```



- V includované stránce se jedná o obyčejné parametry.

Directiva include

- Ještě je možné použít pro includování Standard action include:

```
<%@ include file="layout/taglib.jsp" %>
```

- Na první pohled JSP include i Standard action include vypadají stejně, ale rozdíl je v tom, kdy includování souboru probíhá!
- Directiva include vloží obsah souboru už při kompilaci kódu, zatímco Standard action include vloží obsah souboru za běhu aplikace.
- Používá se k definování taglib knihoven:

taglib.jsp:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
```

<jsp:useBean>

- Typicky pokud potřebujete získat aktuální datum a čas, tak použijete tag <jsp:useBean>:

```
<jsp:useBean id="today" class="java.util.Date" scope="page" />

<c:choose>

    <c:when test="${mydate > today}">moje datum je v budoucnosti</c:when>

    <c:otherwise>moje datum je v minulosti</c:otherwise>

</c:choose>
```

- Další typické použití ve footeru:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

<jsp:useBean id="today" class="java.util.Date" scope="page" />

&copy; Java školení 2007 - <fmt:formatDate value="${today}" pattern="yyyy"/>
```

- Pokud v příslušném scope bean neexistuje, pak ji vytvoří (zavolá její konstruktor bez parametrů).

Vlastní tagy

- JSTL tagy jsou standardizované tagy. Můžete si vytvořit i vlastní tagy. Používají se zejména k odstranění duplikování kódu v JSP stránkách. Existuje několik způsobů, jak vytvořit vlastní tag.
- Nejjednodušší způsob jak udělat tag – pomocí *.tag souboru, ve kterém lze používat JSP syntaxi:

1) Vytvořte soubor typu „tag“ v adresáři WEB-INF/tags (adresář se musí takto jmenovat).

2) V JSP, kde chcete používat tag, zapište tuto direktivu:

```
<%@ taglib prefix="myTags" tagdir="/WEB-INF/tags" %>
```

3) Použití tagu je následující:

```
<myTags:header/>
```

Další tagy: pack:tag

- Tento tag umožňuje transparentně komprimovat a spojovat zdroje jako jsou CSS a JavaScript a tím se zrychluje práce klienta s aplikací (mezi klientem a serverem se přenáší méně dat).
- <http://sourceforge.net/projects/packtag/>
- V současnosti není v Maven repozitáři.

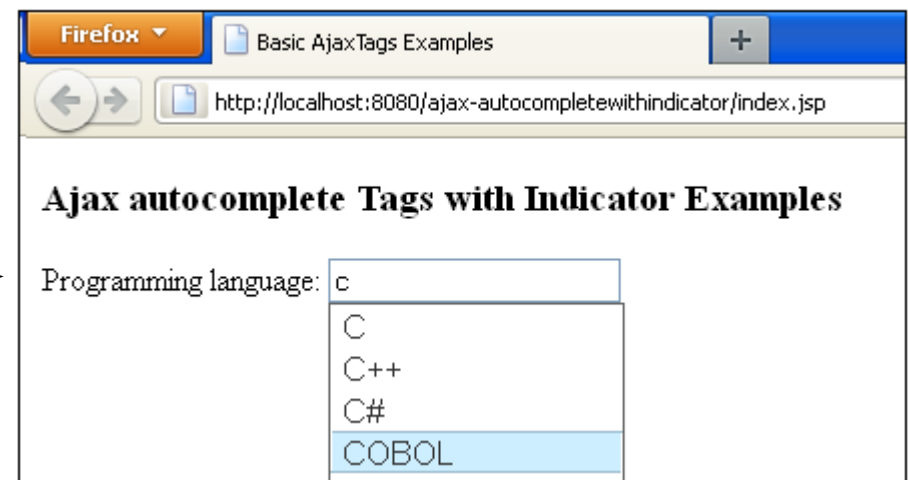
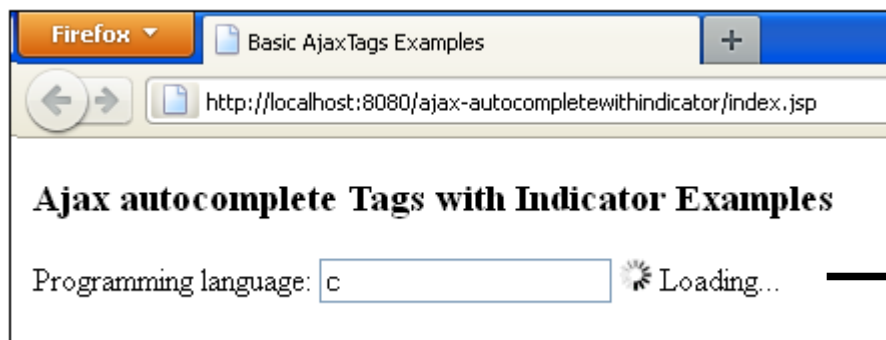
```
var xPackage = {  
  
    /** comments */  
    myMethod: function(aValue, bValue) {  
        var x = aValue * 2; // some stuff  
        return x + bValue;  
    },  
  
    // or not?  
    anotherMethod: function() {  
        var message = xPackage.myMethod(12, 11.3);  
        alert(message + " result");  
    }  
}
```

pack

```
var xPackage={myMethod:function(  
aValue,bValue){var x=aValue*2;  
return x+bValue;},anotherMethod:  
function(){var message=xPackage  
.myMethod(12,11.3);alert(message  
+" result");}}
```

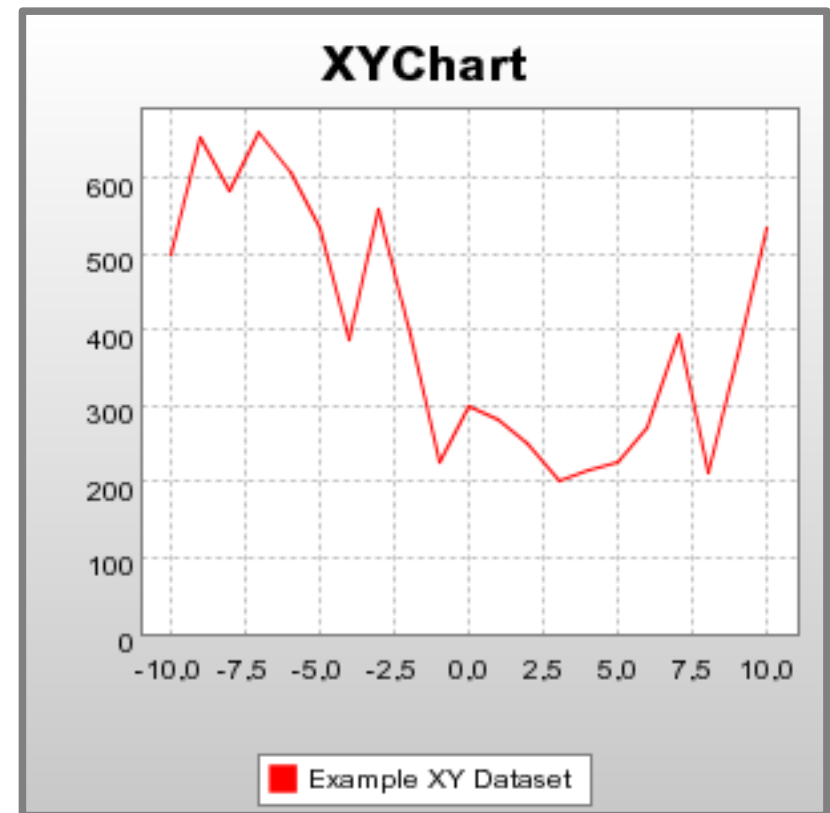
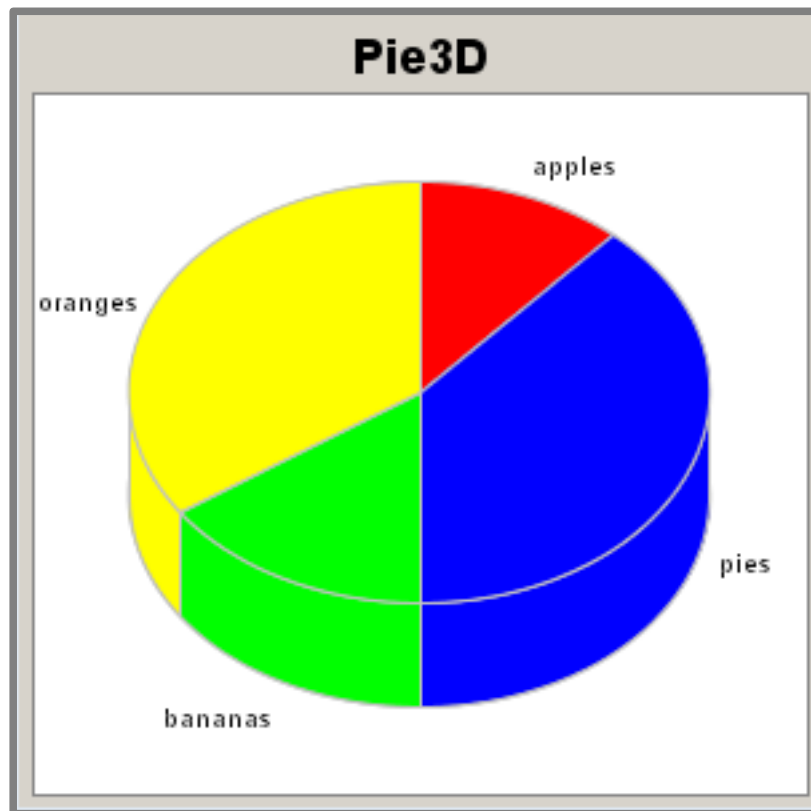
Další tagy: AjaxTags

- Při práci s AJAXem můžete využít několik přístupů, které je možné kombinovat:
 - Napsat si vše v čistém JavaScriptu.
 - Použít JavaScript knihovny jako JQuery, Echo Web Framework, YUI apod.
 - Nepracovat vůbec s JSP soubory, vše napsat v GWT / Vaadin.
 - Použít v některých případech knihovnu AjaxTags:
 - <http://ajaxtags.sourceforge.net/>
 - Obsahuje komponenty jako Autocomplete, Select / dropdown, Tabs, ...
 - V Maven repozitáři je aktuálně starší verze, nejnovější je na webu.



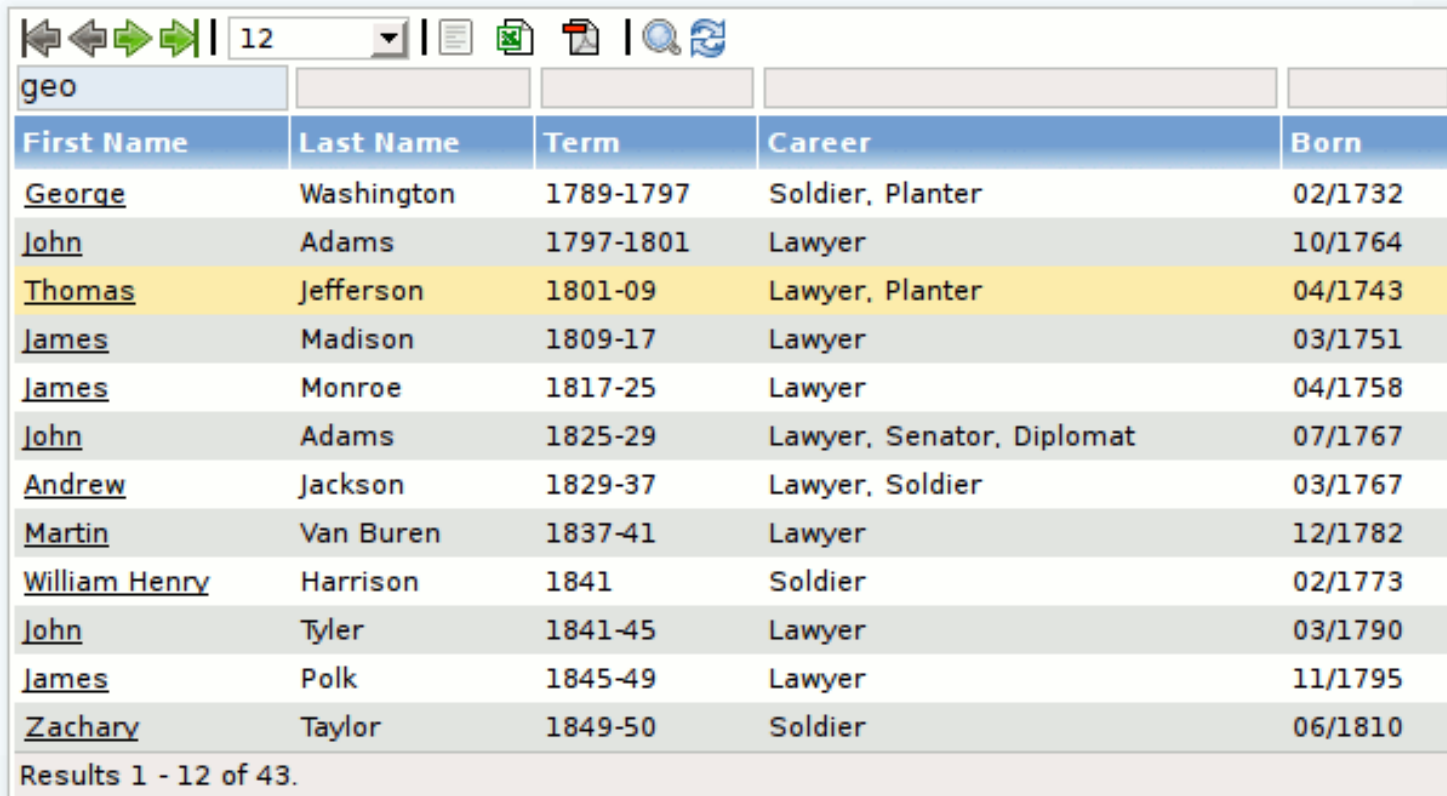
Další tagy: Cewolf

- CeWolf knihovna se používá pro generování grafů, vnitřně používá JFreeChart.
 - <http://cewolf.sourceforge.net/>
 - V Maven repozitáři je aktuálně starší verze, nejnovější je na webu.



Další tagy: JMesa

- JMesa je dynamická tabulka, která podporuje filtrování, řazení, export a editaci dat.
 - <http://code.google.com/p/jmesa/>
 - Je také v Maven repozitáři.



The screenshot shows the JMesa web application interface. At the top, there is a navigation bar with icons for back, forward, search, and other functions. Below the navigation bar is a search input field containing the text "geo". The main content area displays a table with five columns: First Name, Last Name, Term, Career, and Born. The table contains 12 rows of data, with the third row (Thomas Jefferson) highlighted in yellow. The bottom of the table shows the text "Results 1 - 12 of 43."

First Name	Last Name	Term	Career	Born
<u>George</u>	Washington	1789-1797	Soldier, Planter	02/1732
<u>John</u>	Adams	1797-1801	Lawyer	10/1764
<u>Thomas</u>	Jefferson	1801-09	Lawyer, Planter	04/1743
<u>James</u>	Madison	1809-17	Lawyer	03/1751
<u>James</u>	Monroe	1817-25	Lawyer	04/1758
<u>John</u>	Adams	1825-29	Lawyer, Senator, Diplomat	07/1767
<u>Andrew</u>	Jackson	1829-37	Lawyer, Soldier	03/1767
<u>Martin</u>	Van Buren	1837-41	Lawyer	12/1782
<u>William Henry</u>	Harrison	1841	Soldier	02/1773
<u>John</u>	Tyler	1841-45	Lawyer	03/1790
<u>James</u>	Polk	1845-49	Lawyer	11/1795
<u>Zachary</u>	Taylor	1849-50	Soldier	06/1810

Results 1 - 12 of 43.

Další tagy: Display tag

- Další používaná tag knihovna pro tvorbu tabulek.
 - <http://www.displaytag.org/>
 - Je také v Maven repozitáři.

Amount	Project	Task	City
587.0	Arts	et gubergren ut et	Olympia
72.0	Gladiators	duo sit erat justo	Neapolis
277.0	Gladiators	justo tempor consetetur consetetur	Roma
598.0	Gladiators	magna erat tempor justo	Roma
953.0	Gladiators	voluptua nonumy et sadipscing	Olympia
7.0	Taxes	et est tempor eirmod	Roma