

JSP implicitní objekty

JSP struktura stránky

- <HTML elementy>
- Skriptovací elementy – tagy (`<%! %>`)
- **Implicitní objekty (request)**
- Direktivy JSP (*page*)
- Standardní instrukce – akce (*jsp:include*)
- *JSTL, Custom Tags, EL*

JSP implicitní objekty

- Jsou to speciální objekty – proměnné, které jsou poskytovány JSP stránkou (servletem)
- Jsou dostupné ve všech výrazech a skriptletech daného dokumentu (*ne v deklaracích*)
- JSP nám dává k dispozici **9** automaticky vytvořených implicitních objektů:

<http://download.oracle.com/javaee/7/api/>

Implicitní objekt	Typ (interfaces nebo classes)
request	javax.servlet. <i>ServletRequest</i>
response	javax.servlet. <i>ServletResponse</i>
out	javax.servlet.jsp. <i>JspWriter</i>
application	javax.servlet. <i>ServletContext</i>
session	javax.servlet.http. <i>HttpSession</i>
config	javax.servlet. <i>ServletConfig</i>
page	java.lang. <i>Object</i>
pageContext	javax.servlet.jsp. <i>PageContext</i>
exception	java.lang. <i>Throwable</i>

request

- Objekt *request* realizuje požadavek *protokolu HTTP* na URL.
- Každý požadavek (i odpověď) je doprovázen jedním nebo více záhlavími – lze zjistit metodou:
 - *request.getHeader("hlavicka")*
- Hlavičky dotazů:
 - ↗ *User-Agent*
 - ↗ *Accept, Accept-Charset(Encoding, Language)*
 - ↗ *Cookie*
 - ↗ *Host*
 - ↗ *Connection*
- Některé další metody:
 - *getMethod()*
 - *getRemoteAddr()*
 - *getRemoteHost()*
 - *getLocale()*
 - *getSession()*
 - *getParameter()*
 - *getAttribute(), setAttribute()*
 - *setCharacterEncoding(), atd.*



response

- Objekt *response* (odpověď) reaguje na klientský požadavek.
- V JSP se píše obvykle kód, který pouze nastavuje vlastnosti objektu *response*.
 - *sendError()*
 - *sendRedirect()*
 - *addCookie()*
 - *setHeader()*
 - *setContentType()* atd.
- Málokdy se *response* využívá přímo na stránkách JSP, více se používá při programování *servletů*.



out

- Jedná se o objekt typu `JspWriter`, pomocí něhož přidáváme text do HTML stránky.
- Nejdůležitějšími metodami objektu *out* jsou:
 - `print()`
 - `println()`



Pozn.: jelikož prohlížeč převádí znak nového řádku (`\n`) na prázdný znak, rozdíl mezi metodami není žádný a k odřádkování nedojde!

Řešení: do kódu JSP nebo Javy vložit značku `
`

application

- Tento objekt slouží k získávání a ukládání informací o Vašem projektu, který je tvořen JSP stránkami a dalšími dokumenty. Mluvíme o tzv. JSP aplikaci, která je na serveru uspořádána ve vlastní adresářové struktuře.
- Je to objekt, který má k dispozici každý JSP dokument. Mohou se do něj data ukládat nebo z něj získávat.
- *Příklad info-metod:*
 - `application.getServerInfo()`
 - `application.getRealPath(request.getServletPath())`
- Tak jako objekt *request* nebo *session* (viz dále) má aplikace své atributy:
 - `getAttribute()`
 - `setAttribute()`



(atributy *versus* parametry)

- Atributy nejsou parametry!
 - Atributy jsou dodatečné informace, které si pomocí implicitního objektu můžeme posílat mezi serverem a klientem a které také poskytujeme ostatním uživatelům
 - Atribut je typu *Object*, parametr je vždy typu *String*!
 - Neexistuje metoda *setParameter()*; parametry se načítají – z formulářů
- Existují 4 rozsahy platnosti (*scope*) atributů i proměnných na JSP stránkách:
 1. *application-scope* - atributy platí pro celou aplikaci
 2. *session-scope* - platí pouze v rámci jedné session
 3. *request-scope* - pouze v rámci jednoho dotazu (např. servlet nastaví hodnotu *request-atributu* a HTML stránka tento atribut zobrazí)
 4. *page-scope* - platnost na stránce, na které byl atribut deklarován



session

- Webové servery a protokol HTTP pojem *session* neznají. Uživatel odešle požadavek na server, ten potom odešle uživateli zpět odpověď a tím celá relace (transakce) končí. Další požadavek na stejný sever již protokol HTTP považuje za zcela novou transakci.
- JSP a tedy i *servlety* tento problém řeší tak, že umožňují udržování tzv. *session* (sezení, relace) uživatele – klienta. Session si udržuje stav při více požadavcích od jednoho klienta a můžeme ji tedy použít pro uchovávání informací mezi klientem a serverem během všech dotazů – po celou dobu trvání session.
- Session je objekt na serveru, pomocí kterého mohou všechna spojení od jednoho uživatele sdílet data.



session ID

- Aby server mohl přiřadit jednotlivé požadavky k dané **session**, má session své unikátní **id**.
- Při prvním **requestu** JSP konteiner vygeneruje unikátní **session ID** a vrátí ho klientovi v **response**. Klient pak posílá toto ID s každým dalším požadavkem (jinak by se vytvořila jiná session).
- Na to jak přenést *session ID* existují 3 způsoby:
 1. pomocí souboru **cookie** (přenesení se jako parametr v *header* požadavku, viz dříve)
 2. pomocí mechanismu – **URL rewriting**
 3. jako skryté pole v HTML formuláři (ID potom získáme metodou `session.getID()`)

cookies (1)

- **Cookies** jsou malé soubory, které webové servery při první návštěvě vytvoří a uloží na pevný disk vašeho počítače.
- Kromě *session ID* soubor *cookie* dále obsahuje (v podobě řetězců) informace o stavu při procházení různých stránek na webu nebo při pozdějším návratu na web, informace o uložení např. vašich osobních údajů, objednávkách, adresách pro doručení nebo fakturaci atd., které jsou také uloženy na serveru.
- **Důležité:** cookies soubory nelze použít ke spuštění kódu (spuštění programů) nebo k doručení virů do vašeho počítače.



cookies (2)

- Soubor cookie nelze odeslat na žádný jiný server než na ten, z něhož byl do vašeho prohlížeče odeslán.
- Cookie by měly zmizet jakmile zavřete prohlížeč. Je ale možné nastavit cookies na delší dobu života.
- Ve svém prohlížeči můžete příjem a zpracování cookies zakázat.

- Vytvoření instance třídy Cookie:

```
Cookie cookie = new Cookie(String name, String value);
```

- Nastavení doby života:

```
cookie.setMaxAge(30*60);           //... je to na 30 minut
```

- Poslání cookie klientovi:

```
response.addCookie(cookie);
```

URL rewriting

- *Přepisování URL adres* - jedná se o manuální přidání parametru do URL.
- *URL rewriting* musíte provést u každého linku ve vaší aplikaci! To uděláte např. takto:

```
Přejděte na <a href= <%= response.encodeURL("URL") %>další stránku</a>
```

Není to tudíž nejlepší možné řešení.

Jestliže má uživatel zapnuté *cookies*, neudělá tato metoda nic.

- Pokud si tedy chcete být jisti že sessions budou fungovat všude, použijte současně URL rewriting i cookies.

práce se session

- V rámci **session** je opět možné vytvářet a manipulovat s vlastními atributy:
 - `setAttribute();` `getAttribute();` `removeAttribute();`
 - `getAttributeNames();`
- Ukončení **session** - existují 3 způsoby:
 1. session se se ukončí automaticky po určité době neaktivity klienta (*timeout*) – nastaveno ve *web.xml*

```
<web-app ...>
    <servlet> ... </servlet>
    <session-config>
        <session-timeout>35 </session-timeout>
    </session-config>
</web-app>
```

2. zavoláním metody `invalidate()` nad session objektem
3. zhroucením aplikace

config a page

■ *config*

- Každý servlet (.class) obsahuje tzv. *konfiguraci*. Informace o konfiguraci servletu (aktuální stránky) jsou dostupné pomocí metod nad objektem *config*:
 - *getServletName()*
 - *getInitParameter()*
 - *getInitParameterNames()*

■ *page*

- Objekt odkazuje na aktuální stránku. Vlastně jednoduše zastupuje *this*. Pomocí *page* lze nastavit typ obsahu a kódování stránky.

pageContext a exception

■ *pageContext*

- Tento objekt popisuje prostředí, v němž jsou všechny JSP stránky spouštěny. Do této množiny spadá jak stránka, tak i aplikace a vše mezi nimi.
- Z toho plyne, že objekt *pageContext* je nejvšestrannějším ze všech objektů JSP. Cokoliv chcete udělat pomocí jiných implicitních objektů, můžete udělat i pomocí objektu *pageContext*.

■ *exception*

- Objekt (typu *Throwable* - JSE) poskytuje metody pro práci s výjimkami, které mohou vzniknout na stránce JSP.

