# Java a webové aplikácie

# Základná filozofia

- Java umožňuje plnú podporu webových aplikácii s dodržaním zásad OOP
- na uľahčenie je k dispozícii je množstvo rôznych pomocných aplikačných rámcov (cca 50, dominantných 5-6).
- základná filozofia: čo najviac odtieniť používateľa od nástrah a obmedzení HTTP protokolu

# HTTP protokol

- HTTP protokol je primárnym protokolom na prenos dát na webe
- bezstavový protokol – funguje systémom "požiadavka-odpoveď"
  - v HTTP protokole si server nepamätá stav o pripájajúcich sa klientoch
  - voľná analógia: SMS
  - výhoda – nízke požiadavky na server
  - nevýhoda – bezstavovosť treba obchádzať

"Programovanie webových aplikácii je snaha napchať štvorcový kolík do okrúhlej dierky"

# HTTP protokol

- server je jednoznačne identifikovaný URL adresou
- pod ňou je niekoľko zdrojov (stránky, obrázky, PDF...), každý je jednoznačne identifikovaný URL adresou

  http://ics.upjs.sk/~novotnyr/js
- niekoľko základných príkazov:
  - GET *adresa* – získaj dáta zo servera
  - POST *adresa* – pošli dáta na server

# Základné stavebné kamene

- servletový kontajner – Java server, v ktorom bežia webové aplikácie
  - na výber je viacero serverov: Tomcat, Jetty, Glassfish...
- webová aplikácia – tvoria ju statické stránky, obrázky, dynamické stránky, servlety...
- servlet – dedič triedy javax.servlet.http.HttpServlet
  - obsahuje metódy na spracovanie HTTP požiadavky

# Ukážka servletu

```
import javax.servlet.*;
import javax.servlet.http.*;

import java.io.PrintWriter;

public class FirstServlet extends HttpServlet {
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
  throws ServletException, java.io.IOException
  {

    PrintWriter out = response.getWriter();
    out.println("Ahoj, svet!");
  }
}
```

# Štruktúra webovej aplikácie

- webová aplikácia musí mať predpísanú adresárovú štruktúru

```
web
  |–WEB-INF
      |-web.xml
      |–lib
      |-classes
```

- lib – adresár pre JARy knižníc, ktoré webaplikácia potrebuje (napr. spring.jar, mysql-connector.jar, ...)
- classes – adresár s triedami webovej aplikácie (tu sa začína hierarchia balíčkov). Analógia `bin` adresára z klasického projektu.
- web.xml – konfiguračný súbor webovej aplikácie

# Ako nakonfigurujeme webovú aplikáciu

1. vytvoríme požadovanú adresárovú štruktúru
2. nastavíme web.xml
3. nainštalujeme webovú aplikáciu do servletového kontajnera

# Ako nakonfigurujeme webovú aplikáciu

1. nainštalujeme Tomcat
2. nakonfigurujeme obľúbené IDE
3. nasadíme

**Návody:**

http://ics.upjs.sk/~novotnyr/js/tomcat/tomcat.html

http://ics.upjs.sk/~novotnyr/js/web-tomcat/web-tomcat.html

# Alternatívna možnosť

- Jetty je jednoduchý servletový kontajner
- Na rozdiel od Tomcatu je možné ho ľahko embednúť a konfigurovať v kóde
- Stiahneme archív
- Do projektu pridáme
  - jetty-XXX.jar (jadro)
  - jetty-util-XXX.jar (pomocné knižnice)
  - servlet-api-XXX.jar (interfejsy pre servlety)

# Ukážka servletu

```
public class JettyRunner {
  public static void main(String[] args) throws Exception {
    Server server = new Server(8080);

    Context context = new Context(server,
      "/", Context.SESSIONS);

    context.addServlet(
      new ServletHolder(FirstServlet.class), "/*");

    server.start();
    server.join();
  }
}
```

- spustí kontajner
- nasadí servlet
- a sprístupní ho na http://localhost:8080

# Výhody Jetty pre primitívne aplikácie

- netreba adresárovú štruktúru
- netreba web.xml
- netreba zložité konfigurovanie

- samozrejme pri nasadení do ostrej prevádzky sa tomu nevyhneme

# HTTP, príkaz GET a odosielanie dát na server

- v URL adrese sa môžu nachádzať parametre
- v nich možno posielať dáta servletom

```
http://www.google.sk/search?hl=sk&q=java
```

- uvedené za otáznikom v URL
- usporiadané dvojice *kľúč* = *hodnota*
- oddelené ampersandom

# HTTP, príkaz GET a odosielanie dát na server

```
http://www.google.sk/search?hl=sk&q=java
```

| Parameter | Hodnota |
|---|---|
| hl | sk |
| q | java |

```
String parameterHl = request.getParameter("hl");
String parameterQ = request.getParameter("q");
```

**HttpServletRequest**, dostaneme ho v **doGet()**