

pREST

framework pre
webové aplikácie a služby

Peter Rybár
Centaur s.r.o.



Situácia v korporátnej sfére

- Dominuje **technológia** a nie **architektúra**



Situácia na Webe

- Dominuje architektúra **ROA**
- **REST** – štýl softvérovej architektúry pre distribuované hyper-mediálne systémy
- REST Webové aplikácie:
 - **Google**
 - <http://code.google.com/webtoolkit/>
 - **Amazon**
 - <http://developer.amazonwebservices.com/>
 - **Yahoo!**
 - <http://developer.yahoo.com/>

Ktorú cestu zvolit'?

- To je otázka
 - Funkcionality
 - Ceny
 - Výkonu
 - **SEO** – Search engine optimization
 - Otvorenosti kódu
 - Technickej podpory
 - Zložitosti vývoja
 - **Kvality vývojárov**

Výber riešenia

- Prečo je java web frameworkov viac ako realizovaných projektov?
 - Žiaden framework nevie všetko
 - Rôzna náročnosť vývoja
 - Rôzna väzba na iné technológie



pREST

Peter Rybár
Centaur s.r.o.



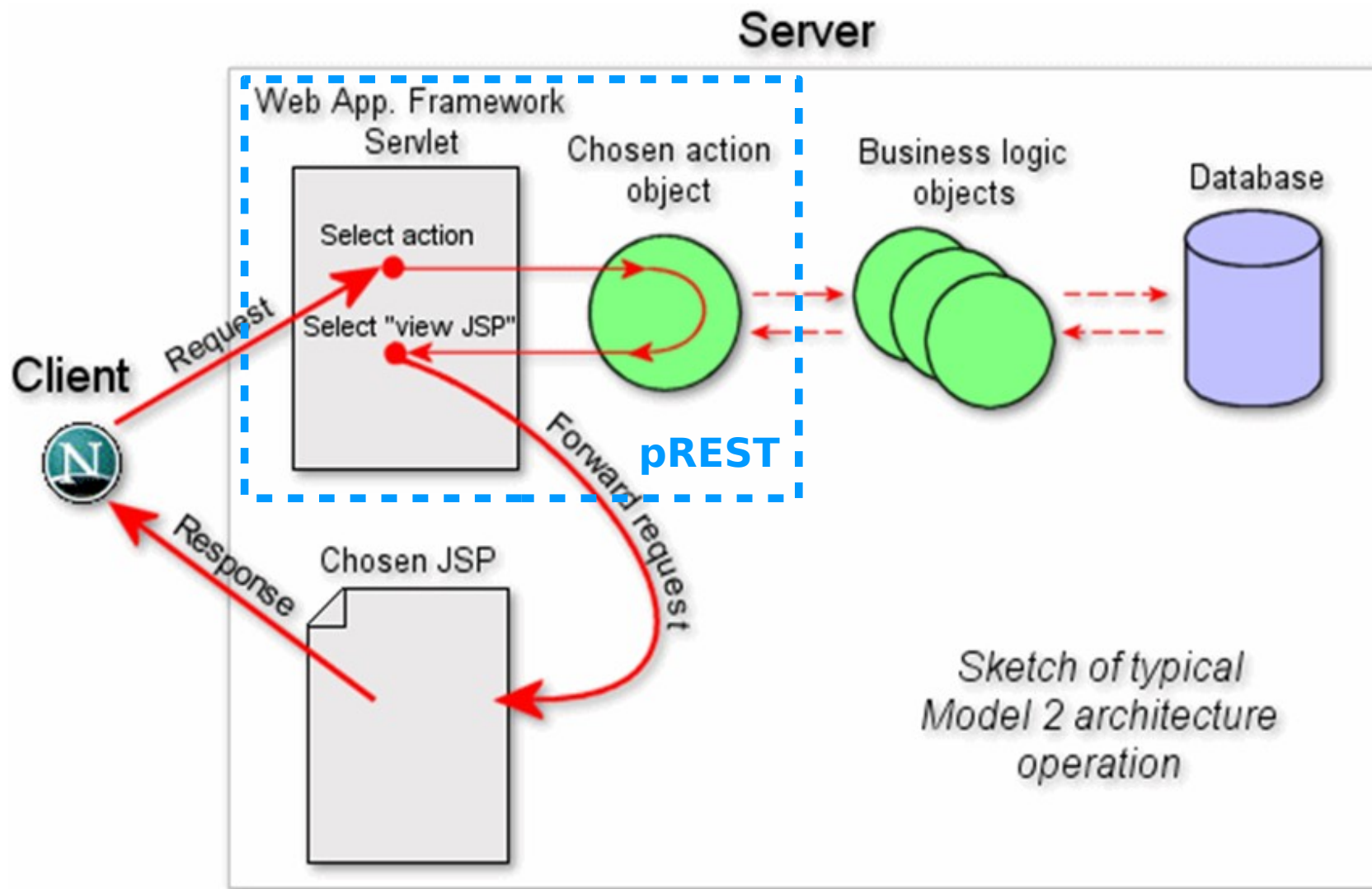
pREST – čo to je?

- **Web framework**
- Softvérový framework navrhnutý na vývoj
 - Webových aplikácií
 - Webových služieb
- Pre vývoj architektonickým štýlom **REST**
- Kladie dôraz na:
 - Jednoduchosť vývoja
 - Efektivitu vývoja
 - Výkon

pREST – zameranie

- **Modularita** – modulárna architektúra
- **Extenzibilita** – jednoduchá možnosť integrácie iných technológií
- **Vývoj REST Web aplikácií a služieb**
 - AJAX, RIA (**R**ich **I**nternet **A**pplications)
- **Oddelenie vývoja**
 - Prezentačnej vrstvy
 - Serverového backendu
 - Aplikačnej logiky

pREST – schéma architektúry



pREST – spíňa požiadavky

- Minimálna doba nábehu vývojára do vývoja
 - čas rádovo v hodinách
- Horizontálny vývoj aplikácií
 - vývojár ovláda iba svoju doménu
 - vyššia kvalita kódu, efektivita
- Voľná väzba
 - nezávislosť od veľkého počtu technológií
 - stabilita
- Platformová nezávislosť – Java 5

pREST – dva komponenty

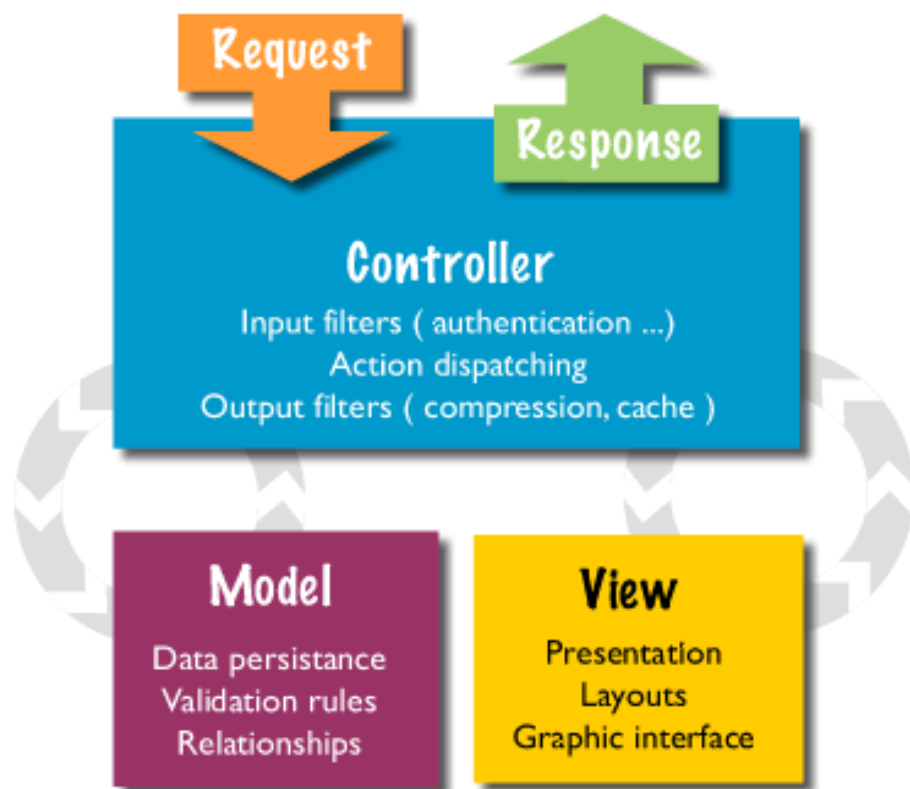
- Framework **pozostáva z dvoch komponentov**
 - **Serverový komponent**
 - **Kontróler** pre servletový kontajner so sadou rozšírení
 - **Klientský komponent**
 - **Javascript knižnica** – je možné ju použiť v kombinácii s ľubovoľnou technológiou na strane servera.

pREST – server

Peter Rybár
Centaur s.r.o.



pREST – kontroler



MVC Diagram

■ Kontroler

- Základná funkčná jednotka
- Extenduje vždy spoločného predka (*prest.core.Controller*)
- Alebo jeho potomka (napríklad *WebController*)
- **Multiaction** – nie komand
- **Akcie** sú mapované na verejné metódy kontrolera

pREST – mapovanie kontrolera

- Kontroler mapujeme na **URL** od rootu aplikácie

- `http://<server>/<app_root>/<controller_path>`

- Příklad:

- `http://server.net/company/depot`

```
mount("/depot", new DepotController());
```

pREST – mapovanie metód

- Metóda kontrolera je mapovaná na časť **URL** prislúchajúcu **akcii** a **HTTP metóde**
 - `http://<server>/<app_root>/<controller_path>/<action>`
- Příklad:
 - `http://server.net/company/depot/parts`

tREST – mapovanie vstupov

- **Vstup z webu len v textovej podobe**

- **URL parametre:** <uri>/param1/param2/.../paramN

- Príklad:

- http://server.net/company/depot/parts/p1/p2/p3

- **GET:** <uri>?key1=value1&key2=value2

- Príklad:

- http://server.net/company/depot/parts?k1=v1&k2=v2

- **POST:** uri encoded v HTTP body

- key1=value1&key2=value2

pREST – mapovanie vstupov

- HTTP parametre sú mapované na **natívne dátové typy** parametrov java metód:
 - Java typy so **stringovým konštruktorom** a ich polia
 - Java **primitívne typy** a ich polia
 - Špeciálne **agregované typy** implementujúce:
 - `trest.core.types.InputType`
 - `trest.core.types.UriParametersType`
 - `trest.core.types.RequestParametersType`
 - `trest.core.types.FileRequestParametersType`

pREST – mapovanie vstupov

- Mapovanie URL a HTTP parametrov

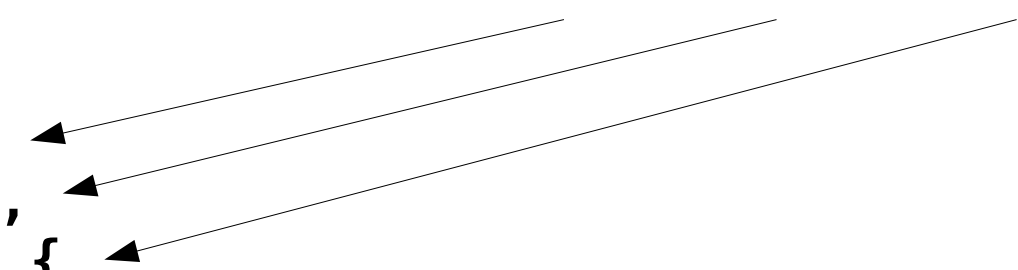
http://server.net/tutorial/input/url_params/str/30.126/xyz

```
public void url_params(String s, Double d, MyType m) {  
}
```



http://server.net/tutorial/input/req_params?k1=str&k2=37&k3=xyz

```
public void req_params(  
    @Key("k1") String s,  
    @Key("k2") Integer i,  
    @Key("k3") MyType m) {  
}
```



pREST – spracovanie výstupov

- Telo HTTP response môže obsahovať buď **textový, alebo binárny stream reprezentovaný**
 - `java.lang.String`
 - `java.io.Reader`
 - `byte[]`
 - `java.io.InputStream`
- **Metóda kontrolera vracia ľubovoľný natívny Java objekt** – do tela **HTTP response** zapíše **stringová reprezentácia objektu** získaná volaním metódy **`toString()`**

pREST – anotácie

- @Doc, @Action, @Key

```
@Doc("Documentation for class Annotations")
public class MyController extends WebController {

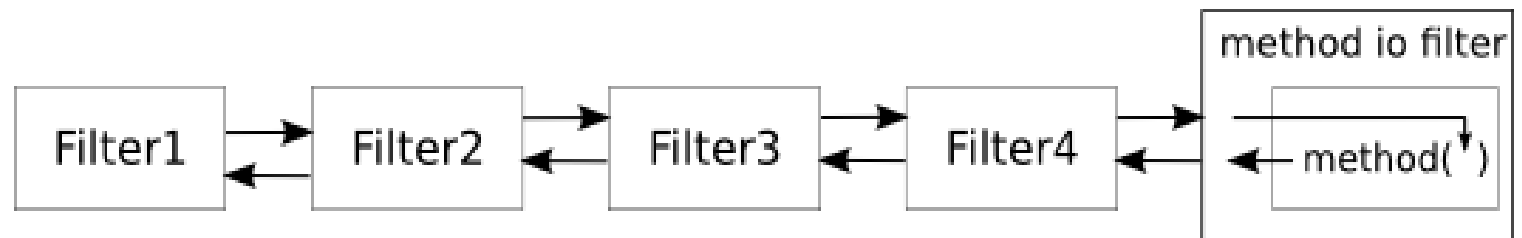
    @Doc("Method with method documentation")
    public void methodDoc(@Doc("Attribute") String s) {}

    @Action(name = "product", httpMethod = "DELETE")
    public void deleteProduct(int id) {}

    public void params_annotacion(@Key("age") int age) {}
}
```

pREST – I/O filtre

filter chain



```
@Filter1
@Filter2("attribute")
public class FiltersController {

    @Filter3(param = "value")
    @Filter4
    public String method() {
        return "xyz";
    }
}
```

pREST

demo aplikácia

Peter Rybár
Centaur s.r.o.



pREST – demo aplikácia

- Validácia vstupných dát
- JSP ako prezentačná vrstva
- RBAC – riadenie prístupu na základe rolí

The image displays two screenshots of the pREST demo application. The left screenshot shows the Login page, and the right screenshot shows the Profile page.

Login Page:

- Form fields: Name (Peter Rybar), Password (masked with dots).
- Button: login

Profile Page:

- Form fields: Name (Peter Rybar), E-mail (peter.rybar@centaur.sk), Age (empty).
- Button: save

pREST – štruktúra projektu

```
demo
|-- docs
|-- lib
|   |-- prest.jar
|-- src
|   |-- demo
|   |   |-- DemoApplication.java
|   |   |-- controllers
|   |       |-- ProfileController.java
|   |       |-- LoginController.java
|-- web
|   |-- META-INF
|   |-- WEB-INF
|   |   |-- web.xml
|   |-- index.html
|-- build.properties
|-- build.xml
```


pREST – klient

Peter Rybár
Centaur s.r.o.



pREST klient

- **JavaScript knižnica** pre tvorbu **RIA**
- **Ciel':**
 - **Jednoduchosť**
 - **Efektivita**
 - **Abstrakcia DOM**
 - **Reusability** a konektivita s okolím
 - **Nezávislosť na serverovej platforme**

Využiť silu JavaScript-u ako prototypovacieho objektového jazyka

pREST klient – Signal-Slot vzor

- **Signal-Slot** návrhový vzor
 - Implementácia **Observer pattern**
 - Ponúka **väčší potenciál ako „callback“**.
- **Koncept**
 - **Objekty** („**widgets**“) emitujú **signály**
 - Signály spájame s **funkciami** – **slotmi**.
- **Koncepcia** pochádza z GUI knižnice QT
 - **Pre realizáciu logiky „front-end“ aplikácií**.

pREST klient – Signal-Slot vzor

- **Signal objekt** sa stará o pripojenie, odpojenie a notifikáciu **slotov**

```
var input = new trest.widgets.forms.TextInput("input_id");

var object = {
    slot_function: function (value) {
        alert("value: " + value);
    }
};

// slot je metoda objektu
input.signal_change.connect(object, object.slot_function);

// slot je funkcia
input.signal_change.connect(slot_function);
```

pREST – klient

ukážka

Peter Rybár
Centaur s.r.o.





pREST

Výkon a efektivita

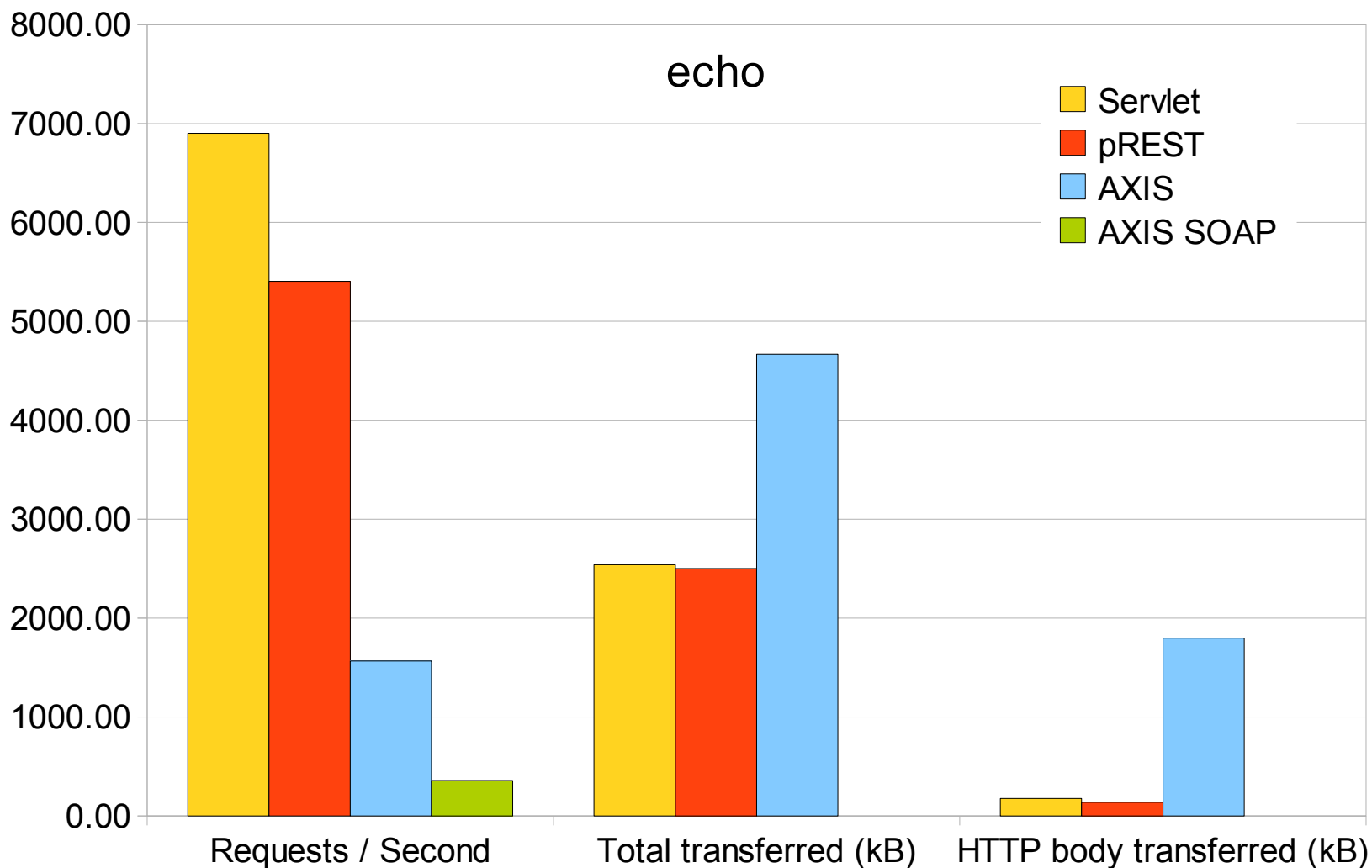
Peter Rybár
Centaur s.r.o.



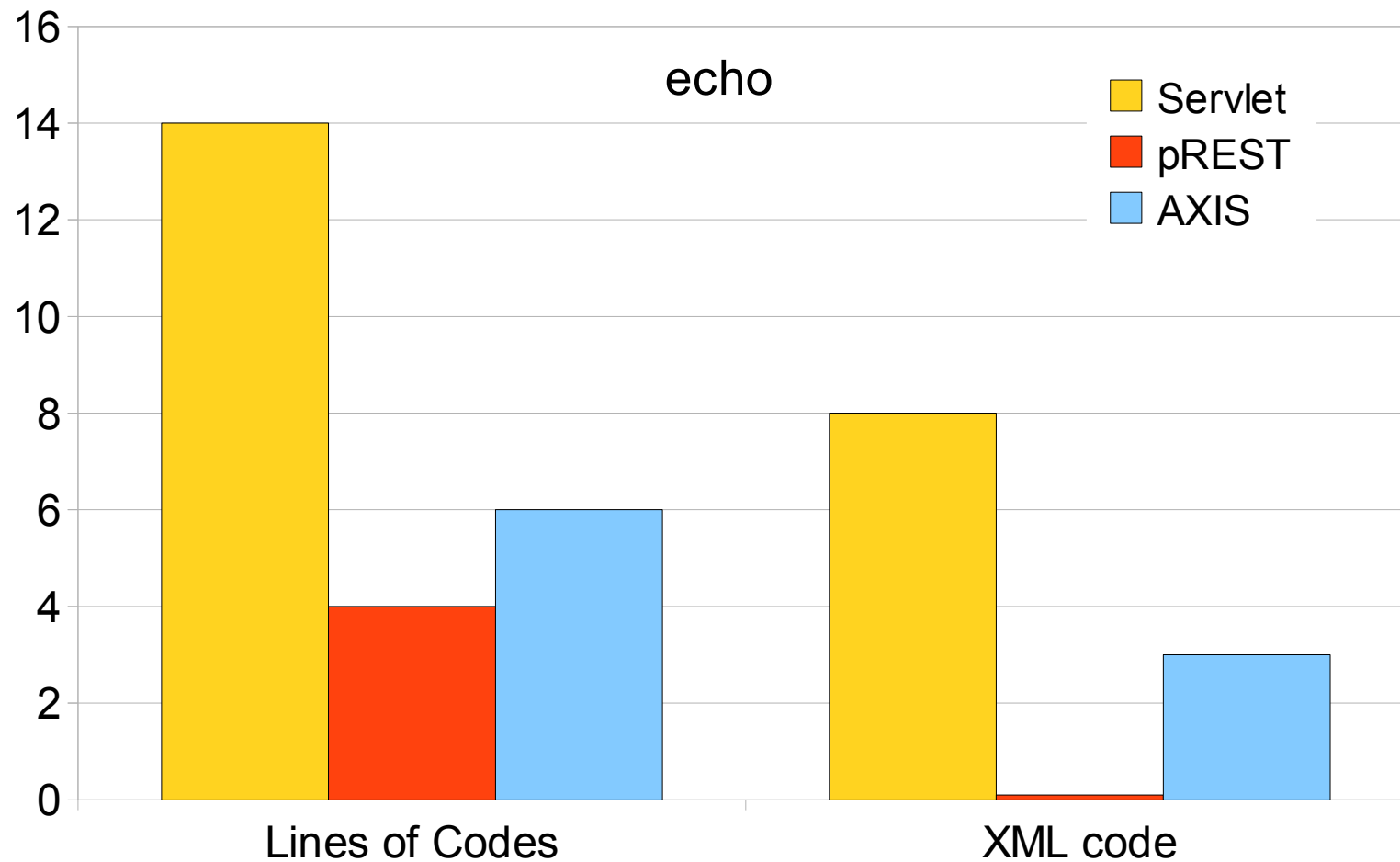
pREST – výkon, efektívita

- Programátor **najlepšie** rozumie kódu
- Kód píšeme v **programovacom jazyku**
- **XML nie je** programovací jazyk
- Generovaný kód **nie je** optimálny
- V generovanom kóde sa **t'ážko** hľadajú chyby a **ešte t'ážšie** opravujú
- Znovupoužitie kódu je najlepšie vo forme **knižníc**
- Špecifickosť kódu je nepriamo úmerná jeho opätovnému použitiu

pREST – výkon, efektivita, testy



pREST – výkon, efektivita, testy



Ďakujem

Otázky

peter.rybar@centaur.sk
dbuchta@centaur.sk
jsivek@centaur.sk

Peter Rybár
Centaur s.r.o.

