Technologie Java Enterprise
Servlets

Łukasz Rybka · Gdańsk 2015

O mnie

Z WYKSZTAŁCENIA

Z WYKSZTAŁCENIA

fizyk

O mnie

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

O mnie

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

O mnie

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

Z ZAMIŁOWANIA

O mnie

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

Z ZAMIŁOWANIA

wykładowca / prelegent / szkoleniowiec

Wykład - materiały

Slajdy:

http://smoczysko.github.io/lectures/jet

Bazowa aplikacja:

https://github.com/Smoczysko/introduction-to-jee-ug

Przykłady z wykładów (i nie tylko):

https://github.com/Smoczysko/introduction-to-jee-examples

Ustalenia

Pytania mile widziane!

Pytania mile widziane!

Konsultacje po zajęciach - informacja odpowiednio przed (np. mailowo)

Ustalenia

Pytania mile widziane!
Konsultacje po zajęciach - informacja odpowiednio przed (np. mailowo)
Wykład nie jest obowiązkowy . . .

Dlaczego warto?

Java Portal Developer

Jeśli:

- *lubisz się rozwijać, czytać, i pogłębiać swoją wiedzę,*
- *nie lubisz bezczynności, ograniczeń i korporacyjnej inercji,*
- *nie cierpisz mechanicznej pracy, wykonywania tych samych czynności i popełniania tych samych błędów dwa razy,*
- *nie przyjmujesz niczego na wiarę, tylko sprawdzasz i drążysz aż do zrozumienia,*
- *chcesz być wsród najlepszych w tym co robisz i być dumnym ze swoich osiągnięć,*
- *nie boisz się wyzwań,*
- *(opcjonalne) lubisz grać w piłkarzyki lub ping-ponga*

przy czym

- *programujesz w Java Enterprise co najmniej pół roku (również hobbystycznie) ,*
- *lubisz poznawac zakamarki różnych technologii, a co njamniej trzy spośród **JPA, EJB, JMS, JBoss, JDBC, Hibernate, PostgreSQL, JSP, Struts** w*

co więcej, interesuje Cię

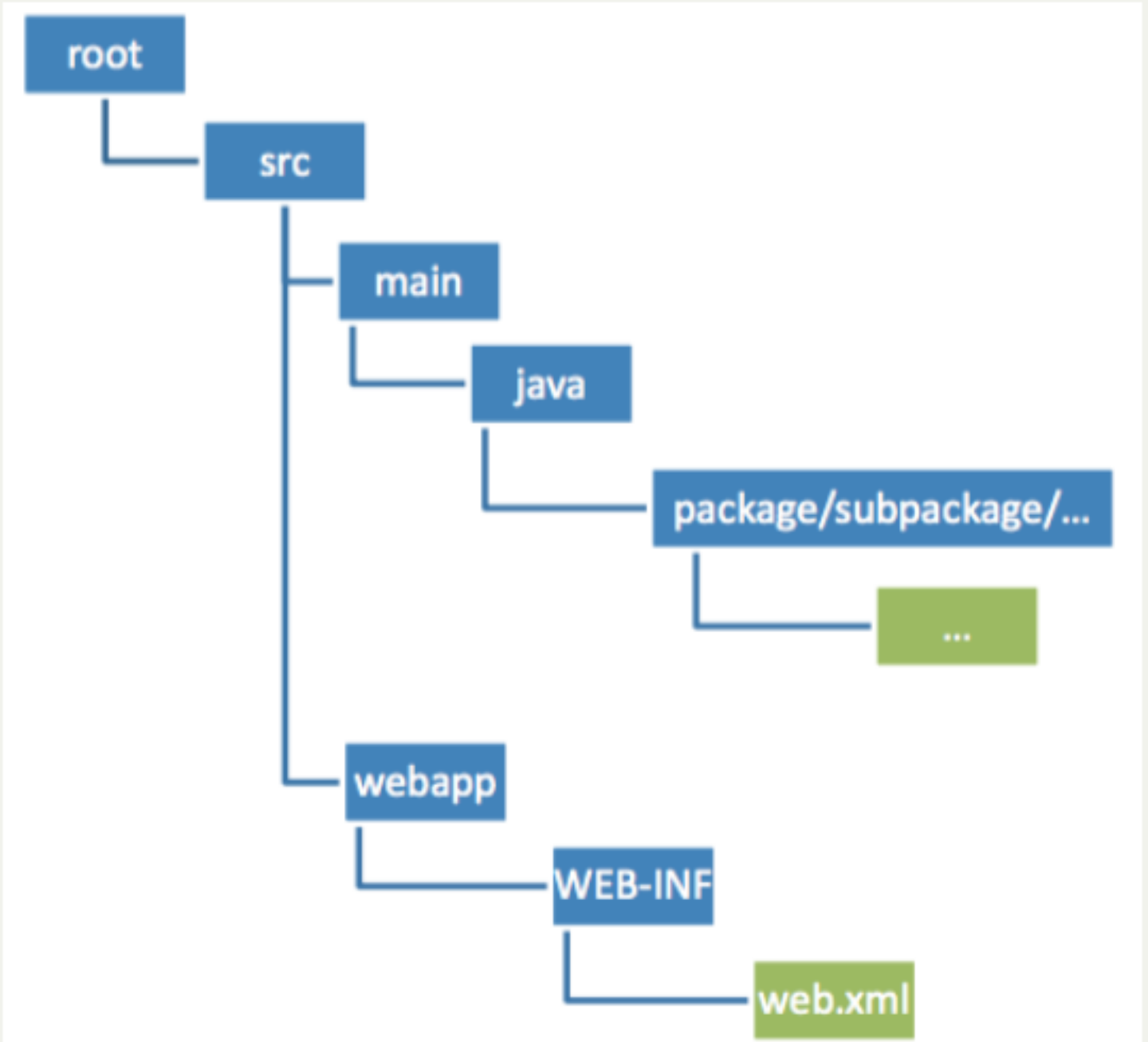
- *nie tylko czy coś działa - ale czy działa wydajnie i przewidywalnie*
- *jak w szczegółach funkcjonują poważne serwisy internetowe,*
- *w których kierunkach rozwija się dzisiejszy Internet*

Dlaczego warto?

Czego oczekujemy

- znajomości Java, SQL, Spring Framework 4, JPA;
- doświadczenia w modelowaniu danych z JPA i optymalizacji operacji na danych w bazie SQL;
- umiejętności tworzenia interfejsów Web API bazujących na JSON, REST;
- umiejętności podstawowej konfiguracji i zarządzania serwerem w środowisku Linux

Podstawowa struktura projektu



Deskryptor aplikacji (web.xml)

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
    <display-name>Introduction to JEE</display-name>
    <description>Some description</description>

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-
class>pl.edu.ug.introductiontojee.web.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Budowa servletu

```
@WebServlet(urlPatterns = "/hello")(1)
public class HelloServlet extends HttpServlet {(2)

    @Override
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");

        PrintWriter writer = response.getWriter();
        writer.println("<html><body><h2>Hello!</h2></body></html>");
        writer.close();
    }
}
```

1. Zastępuje servlet-mapping w web.xml
2. Każdy servlet musi rozszerzać klasę javax.servlet.http.HttpServlet

Uruchomienie aplikacji

```
<plugin>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>8.0.3.v20111011</version>
    <configuration>
        <scanIntervalSeconds>3</scanIntervalSeconds>(2)
        <webAppConfig>
            <contextPath>/helloapp</contextPath>(1)
        </webAppConfig>
    </configuration>
</plugin>
```

1. Ustalamy ścieżkę **względną** do naszej aplikacji
2. Wskazujemy co ile sekund Jetty ma sprawdzać czy zaszły zmiany i przeładowywać aplikację

```
mvn jetty:run
```

Servlet URL

scheme://domain(:port)/path(?query_string)(#fragment_id)

http://localhost:8080/helloapp/hello

HyperText Transfer Protocol (HTTP) methods

HyperText Transfer Protocol (HTTP) methods

GET - prosi o reprezentację danego zasobu

HyperText Transfer Protocol (HTTP) methods

GET - prosi o reprezentację danego zasobu

HEAD - prosi o metainformacje (nagłówki) zasobu

HyperText Transfer Protocol (HTTP) methods

GET - prosi o reprezentację danego zasobu

HEAD - prosi o metainformacje (nagłówki) zasobu

POST - dodaje nowy zasób podrzędnie do wskazanego przez URI

HyperText Transfer Protocol (HTTP) methods

GET - prosi o reprezentację danego zasobu
HEAD - prosi o metainformacje (nagłówki) zasobu
POST - dodaje nowy zasób podrzędnie do wskazanego przez URI
PUT - aktualizacja zasobu

HyperText Transfer Protocol (HTTP) methods

GET - prosi o reprezentację danego zasobu
HEAD - prosi o metainformacje (nagłówki) zasobu
POST - dodaje nowy zasób podrzędnie do wskazanego przez URI
PUT - aktualizacja zasobu
DELETE - usunięcie zasobu

HyperText Transfer Protocol (HTTP) methods

HyperText Transfer Protocol (HTTP) methods

TRACE - zwraca Żadanie klienta (efekt echo)

HyperText Transfer Protocol (HTTP) methods

TRACE - zwraca Żadanie klienta (efekt echo)

OPTIONS - lista metod HTTP wspierana dla wskazanego zasobu

HyperText Transfer Protocol (HTTP) methods

TRACE - zwraca Żadanie klienta (efekt echo)

OPTIONS - lista metod HTTP wspierana dla wskazanego zasobu

CONNECT - przekształca zapytanie w tunel TCP/IP (np. szyfrowana komunikacja szyfrowanej (SSL) przez nieszyfrowane proxy HTTP)

HyperText Transfer Protocol (HTTP) methods

TRACE - zwraca Żadanie klienta (efekt echo)

OPTIONS - lista metod HTTP wspierana dla wskazanego zasobu

CONNECT - przekształca zapytanie w tunel TCP/IP (np. szyfrowana komunikacja szyfrowanej (SSL) przez nieszyfrowane proxy HTTP)

PATCH - modyfikuje jedynie fragment zasobu

Obsługa formularza - GET

Formularz wysłany metodą GET wszystkie pola przesyła za pomocą parametrów zapytania

Obsługa formularza - GET

Formularz wysłany metodą GET wszystkie pola przesyła za pomocą parametrów zapytania

Brak mapowania typów pól formularza na typy Javowe

Obsługa formularza - GET

Formularz wysłany metodą GET wszystkie pola przesyła za pomocą parametrów zapytania

Brak mapowania typów pól formularza na typy Javowe

NPE - konieczne jest ręczne sprawdzenie, czy dany parametr został przesłany

Obsługa formularza - GET

```
@WebServlet(urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) {

        String name = req.getParameter("name");(1)

        String[] hobbies = req.getParameterValues("hobbies");(2)

        boolean isUnderage = req.getParameter("underage") != null;(3)

        String comment = req.getParametr("comment");(4)
    }
}
```

1. Pobranie ostatniej (!) wartości parametru (wartość ostatniego pola o zadanej nazwie w HTML'u)
2. Pobranie wszystkich wartości parametru o zadanej nazwie
3. Pole <input> o typie "checkbox", które nie jest zaznaczone nie będzie przesłane w formularzu
4. Pole <textarea> przesyłane jest w zapytaniu jako zwykły String

Obsługa formularza - POST

Obsługa formularza - POST

Dane z formularza przesyłane są w ciele zapytania HTTP

Obsługa formularza - POST

Dane z formularza przesyłane są w ciele zapytania HTTP

"Brak" ograniczenia wielkości danych przesyłanych - w niektórych przeglądarkach długość URL jest limitowana

Obsługa formularza - POST

Dane z formularza przesyłane są w ciele zapytania HTTP

"Brak" ograniczenia wielkości danych przesyłanych - w niektórych przeglądarkach długość URL jest limitowana

Dostęp do danych z poziomu servletu identyczny jak przy metodzie GET

HTTP Session

HTTP Session

Zbiór informacji o komunikacji użytkownika (przeglądarki) z serwerem

HTTP Session

Zbiór informacji o komunikacji użytkownika (przeglądarki) z serwerem

Dostępna (i modyfikowalna) z poziomu servletu

HTTP Session

Zbiór informacji o komunikacji użytkownika (przeglądarki) z serwerem

Dostępna (i modyfikowalna) z poziomu servletu

Istnieje możliwość nasłuchiwania na zdarzenia utworzenia i zniszczenia sesji

HTTP Session

Zbiór informacji o komunikacji użytkownika (przeglądarki) z serwerem

Dostępna (i modyfikowalna) z poziomu servletu

Istnieje możliwość nasłuchiwania na zdarzenia utworzenia i zniszczenia sesji

Może wygasnąć lub zostać usunięta (np. wyczyszczenie cookies przeglądarki)

HTTP Session

```
@WebServlet(urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) {

        HttpSession session = request.getSession();(1)

        Object counter = session.getAttribute("counter");(2)
        session.setAttribute("counter", counter);(3)

        int creationTime = session.getCreationTime();>4>
        int lastAccessedTime = session.getLastAccessedTime();(4)
    }
}
```

1. Pobranie obiektu sesji
2. Pobranie obiektu (serializowalnego) z sesji
3. Zapisanie (lub usunięcie i zapisanie !) obiektu do sesji
4. Pobranie metainformacji o sesji

HTTP Session listeners

HTTP Session listeners

Możliwość nasłuchiwania na utworzenie i usunięcie obiektu sesji

HTTP Session listeners

Możliwość nasłuchiwania na utworzenie i usunięcie obiektu sesji

Nasłuchiwanie na zdarzenia (un)bound obiektu w sesji

HTTP Session listeners

Możliwość nasłuchiwania na utworzenie i usunięcie obiektu sesji

Nasłuchiwanie na zdarzenia (un)bound obiektu w sesji

Nasłuchiwanie na dodanie, usunięcie i zmianę dowolnego obiektu (atrybutu) w sesji

HTTPSessionListener

```
@WebListener
public class RequestSessionListener implements HttpSessionListener {
    @Override
    public void sessionCreated(HttpSessionEvent event) {

    }

    @Override
    public void sessionDestroyed(HttpSessionEvent event) {

    }
}
```

HttpSessionBindingListener

```
public class Counter implements HttpSessionBindingListener {
    // Standard POJO definition

    @Override
    public void valueBound(HttpSessionBindingEvent event) {

    }

    @Override
    public void valueUnbound(HttpSessionBindingEvent event) {

    }
}
```

HttpSessionAttributeListener

```
@WebListener
public class CounterSessionAttributeListener implements
HttpSessionAttributeListener {
    @Override
    public void attributeAdded(HttpSessionBindingEvent event) {
        if (event.getValue() instanceof Counter) {1

        }
    }

    @Override
    public void attributeRemoved(HttpSessionBindingEvent event) {
        if (event.getName().equals("counter")) {2

        }
    }

    @Override
    public void attributeReplaced(HttpSessionBindingEvent event) {
        if (event.getValue() instanceof Counter) {1

        }
    }
}
```

Servlet Context

http://jee-context-example.herokuapp.com

Servlet Context

Zawiera zbiór informacja i metod używanych do komunikacji między servletem a kontenerem

http://jee-context-example.herokuapp.com

Servlet Context

Zawiera zbiór informacja i metod używanych do komunikacji między servletem a kontenerem

Informacje zapisane w kontekście są dzielone między zapytaniami i sesjami

http://jee-context-example.herokuapp.com

Servlet Context

Zawiera zbiór informacja i metod używanych do komunikacji między servletem a kontenerem

Informacje zapisane w kontekście są dzielone między zapytaniami i sesjami

API analogiczne do tego w sesji zapytania

http://jee-context-example.herokuapp.com

Servlet Context

```
@WebServlet(urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) {

        ServletContext servletContext = getServletContext();

        Object counter = servletContext.getAttribute("counter");2
        servletContext.setAttribute("counter", counter);3

        String contextPath = servletContext.getContextPath();>4>
        String serverInfo = servletContext.getServerInfo();4
    }
}
```

1. Pobranie obiektu kontekstu
2. Pobranie obiektu (serializowalnego) z kontekstu
3. Zapisanie (lub usunięcie i zapisanie !) obiektu do kontekstu
4. Pobranie metainformacji o kontekście, servlecie i samym serwerze

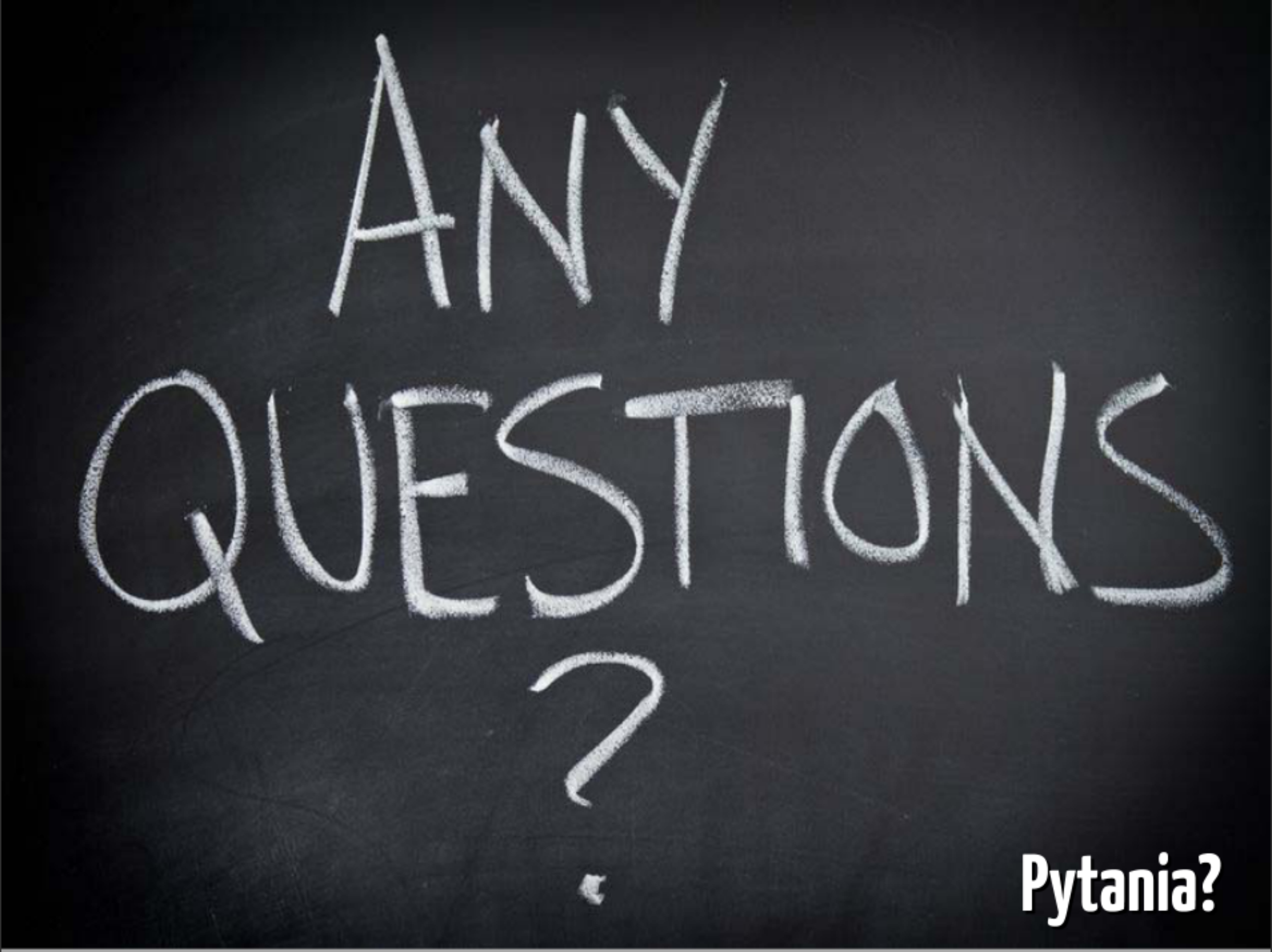
ServletContextListener

```
@WebListener
public class CounterServletContextListener implements
ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent event) {

    }

    @Override
    public void contextDestroyed(ServletContextEvent event) {

    }
}
```



Pytania?