# EXPRESSION LANGUAGE

# Expression Language

- JSR 245
- Скриптовый язык
  - доступ к Java компонентам
    - более наглядный, чем с помощью действий
  - вычисления выражений
  - имеет свой синтаксис
- Составная часть JSP (с JSP 2.1)

# Синтаксис EL

Общий вид: ${EL expression}

Выражение может включать:

- операторы EL / литералы
  - арифметические выражения
- конструкции доступа к полям атрибутов
- конструкции доступа к элементам контейнеров
  - массивы / списки / карты
- неявные объекты
- вызов функций (стандартные/user-defined)

# Литералы

Литералы - это константы.

В выражениях EL могут быть использованы следующие литералы:

- Boolean: true / false
- Integer: 43 / 0
- Double: 443.11E3 / 443.11
- String: 'str' / "str"
  - экранирование в строках: \' \" \\
- Нул-тип: null

# Доступ к полям бинов

user.getLogin()

${user.login} ~ ${user["login"]} ~ ${user['login']}

Поиск атрибута с именем user:

page  request  session  application

- атрибут не найден - вывода нет
- атрибут найден
  - вызов метода getLogin и приведение к String
  - вывод результата

# Доступ к элементам

Массив: ${ar['1']} ~ ${ar["1"]
${ar[index-as-attribute]}

Список: ${list['1']} ~ ${list["1"]
${list[name-of-index-attribute]}

Карта: ${map['key']} ~ ${map["key"]} ~ ${map.key}
${map[name-of-key-as-attribute]}

key – строка.

# Операторы [] .

Выражение expr-a.identifier-b

эквивалентно expr-a["identifier-b"]

При вызове функций (*см. последний слайд*)

expr-a.identifier-b(params)

expra["identifier-b"](params)

# Контейнеры атрибутов

Неявные объекты, тип Map<String, Object>

pageScope requestScope

sessionScope applicationScope

Определены для использования внутри выражений.

Поиск атрибута user только в области session: ${sessionScope.user.login}

# Неявные объекты

Контейнеры, имеют тип Map<String, Value>

Контейнер  элементы контейнера

paramValues  <имя парам., массив значений>
header  <имя заголовка, значение>
headerValues <имя заг., массив значений>
cookie  <имя cookie, объект Cookie>
initParam  <имя парам. контекста, значение>

массив значений – массив строк

# Логические операции

&& and ===> И по краткой схеме
|| or ===> ИЛИ по краткой схеме
! not ===> ОТРИЦАНИЕ

Оба операнда приводятся к Boolean.

# Оператор empty

Проверяет на пустоту объект.

Пример:
${**empty** x}

Возвращает true если x:
1) null
2) строка нулевой длины
3) массив длиной 0
4) пустые Map или Collection

Во всех других случаях возвращает false

# Операции сравнения

== eq ===> равно
!= ne ===> не равно
Для сравнения используется метод equals.

< lt ===> меньше
> gt ===> больше
<= le ===> меньше или равно
>= ge ===> больше или равно
Для сравнения используется метод compareTo.

Если один из операндов или оба равны null, то результат операций false, кроме ==/eq:
null == null ==> true  null eq null ==> true

# Унарный минус

Меняет знак числа на противоположный.

Пример:
${-4}

Если операнд null, результат **0**

Замечание: операция унарный **+** в EL не определена.

# Условный оператор выбора

Вид:
A ? B : C

Значение A ===> Результат

true ===> B
false ===> C

# Арифметические операции

+ ===> сложение
- ===> вычитание
* ===> умножение
/ div ===> деление
% mod ===> остаток от деления

Если один из операндов null, вместо него будет подставлен **0**.

Пример: ${2+3*4}

# Определение функций

- Определить публичный статический метод в некотором классе.
- В TLD библиотеки определить имя функции и ее сигнатуру по сигнатуре метода.
- Связать функцию с классом, который ее реализует.
- На JSP странице подключить библиотеку, вызывать функцию.

# Пример определения функции

JAVA CLASS

```
public class A {
    public static String fullName(User user) {...}
}
```

TLD

```
<short-name>mylib</short-name>
<uri>uri string</uri>
<function>
    <name>funcName</name>
    <function-class>com.my.A</function-class>
    <function-signature>
        java.lang.String fullName(com.my.User)
    </function-signature>
</function>
```

JSP

```
<%@ taglib uri="uri string" prefix="mylib" %>
${mylib:funcName(user)}
```