Язык Expression Language

Язык Expression Language

- Основан на языке SPEL из JSTL 1.0
  - Simplest Possible Expression Language
- Разработан с учетом языков XPath и ECMAScript
- Входит в состав JSP с версии 2.0
- EL-элементы — замена скриптовых выражений
  - Более компактная форма, чем у Java-выражений
  - Синтаксис EL не противоречит спецификации XML
  - Основной принцип обработки ошибок:
    лучше ничего не вывести, чем выдать исключение

Вычисление EL-элементов

- **НЕ** преобразуются в соответствующий Java-код на 1ом этапе трансляции
- На этапе выполнения интерпретируются EL-вычислителем
  - Большие накладные расходы
  - Ошибки в EL-элементах могут быть обнаружены только при обработке запросов
- EL-элементы по времени вычисления бывают
  - Немедленные (immediate)
  - Отложенные (deferred)

Немедленные и отложенные EL-выражения

- Немедленные
  - `${текст_EL-выражения}`
  - Из JSP 2.0
  - Значение вычисляется сразу, как только EL-элемент встречается на JSP-странице
  - Используются в шаблоне и в тех атрибутах действий, которые допускают runtime-значения
- Отложенные
  - `#{текст_EL-выражения}`
  - Из JSF 1.1
  - Значение вычисляется тогда, когда это необходимо
    - В JSF — на шаге «Применить значения из запроса» и/или «Отобразить ответ»
  - Используются только в тех атрибутах действий, где допустимы отложенные EL-выражения
    - JSF Tag Libraries, JSTL

Типы EL-выражений

- Value Expression
  - rvalue — вычисляемое значение
    - `${2*2}`, `#{item.guessCount * 2}`
  - lvalue — ссылка на объект или его свойство
    - `${item.guessCount}`, `#{item.playerName}`
- Method Expression
  - Ссылка на метод объекта
  - Только для отложенных EL-выражений
  - В JSF используется для связи обработчика событий с компонентом UI
    - `<h:commandButton value="Refresh" actionListener="#{bean.refresh}"/>`

Синтаксис EL

Операции

1. `[ ]  .`
2. `( )`
3. – (унарный) `not ! empty`
4. `*  div  /  mod  %`
5. `+  -`
6. `lt  <  gt  >  le  <=  ge  >=`
7. `eq  ==  ne  !=`
8. `and  &&`
9. `or  ||`
10. `?:`

Нет присвоения!

Синтаксис EL
Операции [] и .

- **вырА.идБ** и **вырА["идБ"]** — эквивалентны!
- Вычисление выражения **вырА[вырБ]**

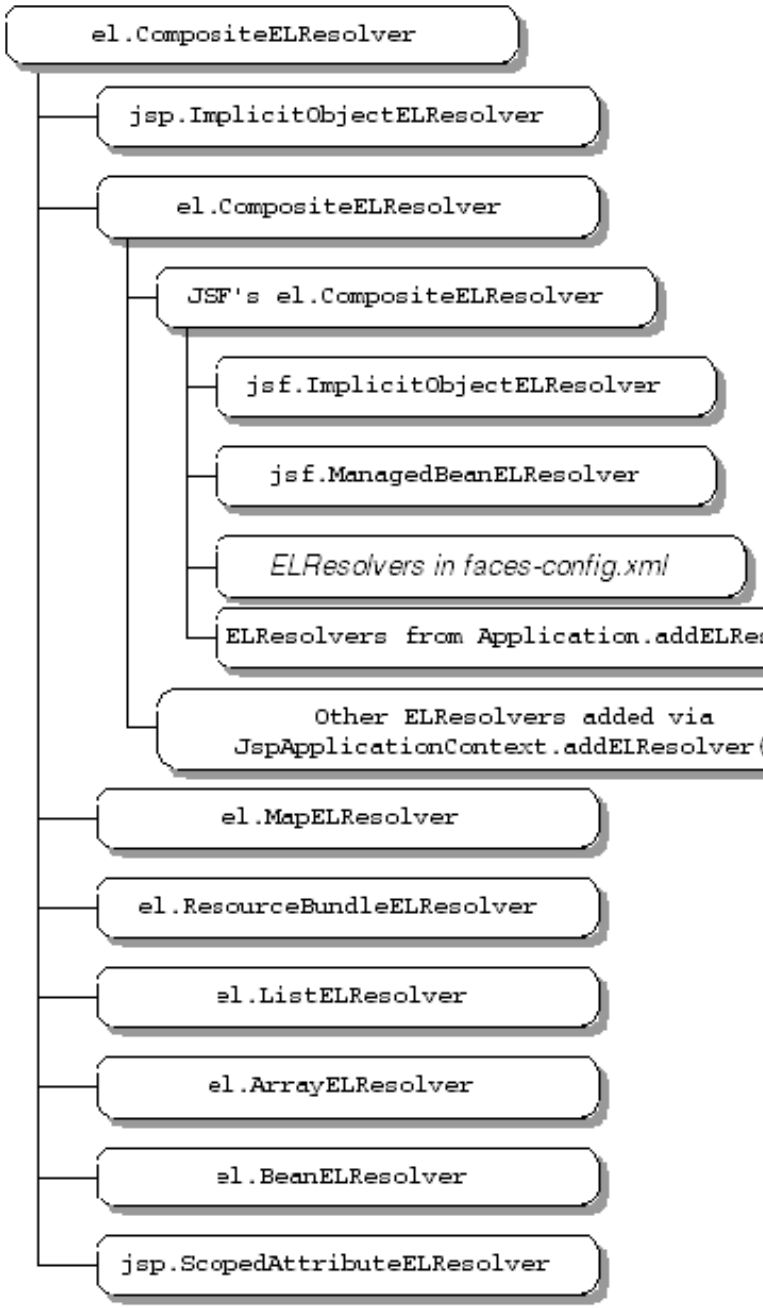
```
1) значА = eval (вырА);
2) значБ = eval (вырБ);
3) if (значА instanceof java.util.Map)
       return значА.get(значБ);
4) if (значА instanceof java.util.List)
       return значА.get((int) значБ);
5) if (значА.getClass().isArray())
       return значА[(int) значБ];
6) return значА.getЗначБ();
```

Синтаксис EL
Литералы

- `null`
- Булевские литералы - `true` и `false`
- Числовые значения (целочисленные и дробные) - как в языке Java
- Строковые литералы - в двойных или одинарных кавычках
  - Распознаются esc-последовательности `\"` `\'` `\\`

Синтаксис EL
Переменные

- Имена переменных преобразуются в объекты с помощью «резольверов»
- Резольверы выстраиваются в цепочку, их порядок определяет:
  - последовательность поиска объектов
  - алгоритм вычисления операции []



Синтаксис EL
Переменные

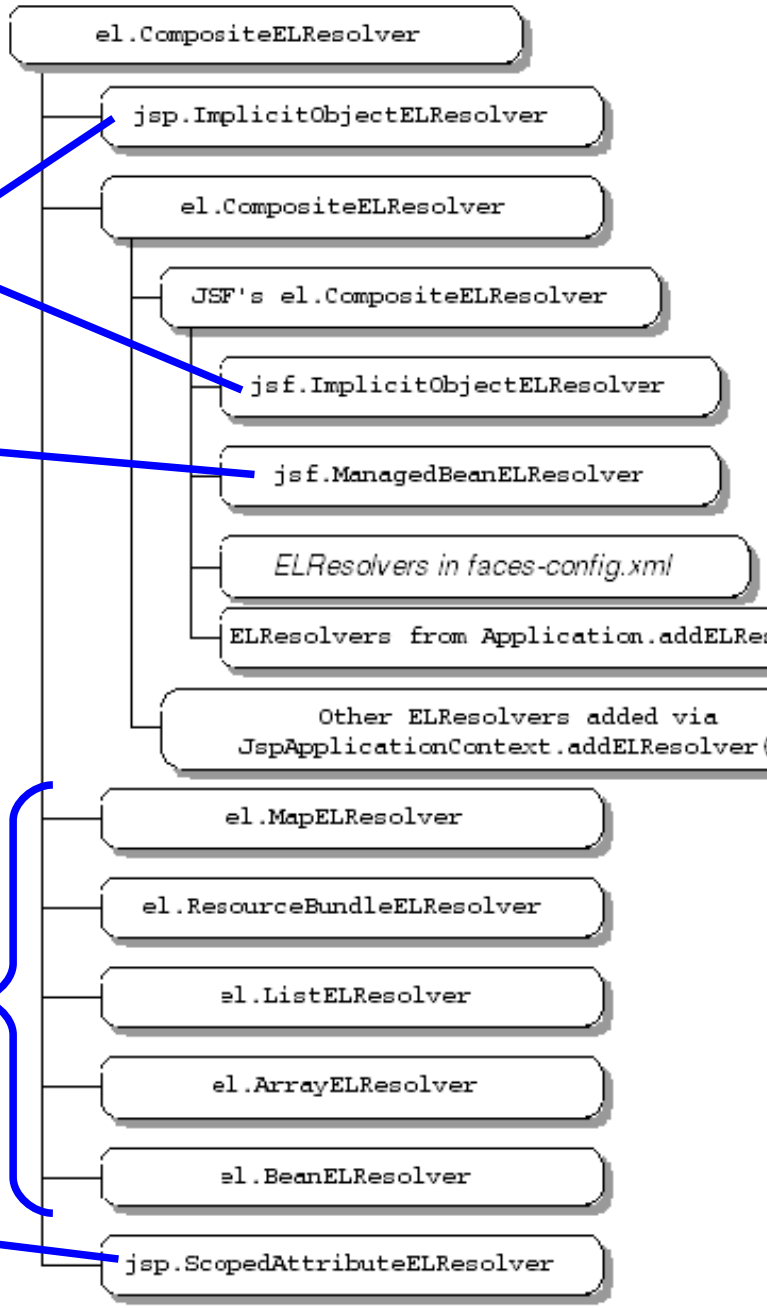
- Имена пере… …в
  объекты с помощью «резольверов»
- Резольверы … …нку,
  их порядок …
  - > последова… …ов
  - > алгоритм вычисления операции []

Встроенные объекты

Управляемые бины
(lazy initialization)

Отображение, набор ресурсов,
список, массив, JavaBean

Поиск атрибутов в контекстах



Сравнение скриптовых и EL-выражений

- Скриптовое выражение:

```
<center>
   <jsp:useBean id="foo"
          class="FooBean" />
   <%= foo.getBar() %>
</center>
```

- EL-выражение:

```
<center>
   ${foo.bar}
</center>
```

- Скриптовые элементы:

```
<%  Map m = (Map)pageContext
                .getAttribute("states" );
   State s = ((State)m.get( "NY" ));
   if( s != null ) {
%>
      <%= s.getCapitol() %>
<%  } %>
```

- EL-выражение:

```
${states["NY"].capitol}
```

Встроенные объекты

Имя объекта	Тип	Описание	$	# (JSF)
pageContext	PageContext	контекст JSP-страницы	+	
pageScope	Map	атрибуты JSP-страницы	+	
request	HttpServletRequest	запрос		+
requestScope	Map	атрибуты запроса	+	+
session	HttpSession	HTTP-сессия		+
sessionScope	Map	атрибуты сессии	+	+
application	ServletContext	веб-приложение		+
applicationScope	Map	атрибуты веб-приложения	+	+
param	Map	параметры запроса	+	+
paramValues	Map	параметры запроса со всеми значениями	+	+
header	Map	заголовки запроса	+	+
headerValues	Map	заголовки запроса со всеми значениями	+	+
cookie	Map	куки запроса	+	+
initParam	Map	параметры инициализации	+	+
facesContext	FacesContext	контекст запроса в каркасе JSF		+
view	UIViewRoot	корень дерева компонентов в текущем JSF-представлении		+

Синтаксис EL
Функции

- Позволяют расширить набор операций, доступных в EL-выражениях
  - В JSTL 1.2 определено 16 функций

1) Функция описывается в TLD-файле

```
<function>
  <name>nickname</name>
  <function-class>Mypackage.MyFunctions</function-class>
  <function-signature>
    java.lang.String nickname(java.lang.String)
  </function-signature>
</function>
```

2) Реализуется в Java-классе статическим методом

3) Используется в JSP-странице

```
<%@taglib prefix="my" uri="http://my.com/functions" %>
<h1>Hello, dear ${my:nickname(user)}</h1>
```

Синтаксис EL
Правила преобразования типов

- Отличаются от принятых в языке Java
- Сформулированы в терминах объектных типов
  - Автоматический boxing / unboxing
- Определены правила преобразования значения:
  - В строку (класс String)
  - В числовой тип (подкласс Number)
  - В символ (класс Character)
  - К логическому типу (класс Boolean)
  - К перечислимому типу
  - К произвольному объектному типу

Примеры использования EL-выражений

- Для вычисления значения атрибута действия

```
<some:tag value="${expr}"/>
<some:tag value="some${expr}${expr}text${expr}"/>
```

- В тексте шаблона

```
<tr>
    <td>${client.lastName} ${client.firstName}</td>
    <td>${client.place}</td>
</tr>
```

  - Внутри действий

```
<r:report>
  <r:title>
    ${client.lastName} ${client.firstName}'s Orders
  </r:title>
</r:report>
```