JSP & Servlets

Expression Language

Expression Language

Een request attribuut met een object

zie vorige keer:

De waarde van een request attribuut kan een object zijn.
(vb. instantie van de class Begroeting)

in de servlet:

request.setAttribute("boodschap", new Begroeting( ));

in de jsp:

${boodschap} → het resultaat van de toString( ) method

Hardcoded values in EL

Ook mogelijk in EL:

Hardgecodeerde waarden:

gehele getallen	${ 7 }
getallen met decimalen	${ 40.3399 }
strings of ${ ‘james’ }	${ “james” }
booleans	${ false }

Wiskundige operatoren in EL

EL kent de volgende wiskundige operatoren, die je toepast op getalwaarden

- `+` `-` `*` `/` en `%`
- `div` is een synoniem voor `/` (delen)
- `mod` is een synoniem voor `%` (rest bepalen bij delen)

Als een request attribuut `getal` de waarde `9` bevat,
geven volgende EL expressies bijbehorende resultaten

EL expressie	resultaat	EL expressie met zelfde resultaat
`${getal + 1}`	`10`	
`${getal - 1}`	`8`	
`${getal * 2}`	`18`	
`${getal / 2}`	`4.5`	`${getal div 2}`
`${getal % 2}`	`1`	`${getal mod 2}`

Vergelijkingsoperatoren in EL

EL kent de volgende vergelijkingsoperatoren,
die je toepast op getalwaarden of string waarden

- `==` `!=` `>` `<` `>=` `<=`
- `eq` is een synoniem voor `==`
- `ne` is een synoniem voor `!=`
- `gt` is een synoniem voor `>`
- `ge` is een synoniem voor `>=`
- `lt` is een synoniem voor `<`
- `le` is een synoniem voor `<=`

Als een request attribuut `getal` de waarde `9` bevat,
geven volgende EL expressies bijbehorende resultaten

Vergelijkingsoperatoren in EL

EL expressie	resultaat	EL expressie met zelfde resultaat
${getal == 9}	true	${getal **eq** 9}
${getal != 9}	false	${getal **ne** 9}
${getal > 9}	false	${getal **gt** 9}
${getal >= 9}	true	${getal **ge** 9}
${getal < 9}	false	${getal **lt** 9}
${getal <= 9}	true	${getal **le** 9}

Logische operatoren in EL

EL kent de klassieke logische operatoren, die je toepast op boolean waarden

- `!` `&&` `||`
- `not` is een synoniem voor `!`
- `and` is een synoniem voor `&&`
- `or` is een synoniem voor `||`

Als een request attribuut `getal` de waarde `9` bevat,
geven volgende EL expressies bijbehorende resultaten

EL expressie	**resultaat**	**EL expressie met zelfde resultaat**
`${! (getal == 9)}`	`false`	`${not (getal == 9)}`
`${getal > 8 && getal < 10}`	`true`	`${getal > 8 and getal < 10}`
`${getal > 8 || getal < 10}`	`true`	`${getal > 8 or getal < 10}`

De conditionele operator ?

Syntax

```
voorwaarde ? waardeAlsVoorwaardeTrue : waardeAlsVoorwaardeFalse
```

Als de voorwaarde vóór `?` gelijk is aan `true`, geeft deze operator de waarde tussen `?` en `:` terug. Anders geeft deze operator de waarde na `:` terug.

Als een request attribuut `getal` de waarde `7` bevat, geeft de EL expressie `${getal == 7 ? "geluk" : "geen geluk"}` de waarde `geluk` terug.

De operator empty

Je vermeldt na `empty` een expressie. De empty operator geeft `true` terug als

- de expressie gelijk is aan `null`
- de expressie een lege `String` is
- de expressie een lege verzameling (array, `List`, `Set`, `Map`) is
- de expressie de naam is van een onbestaand request attribuut

Als een request attribuut `klanten` ingevuld is met een lege List

```
request.setAttribute("klanten", new ArrayList());
```

geeft de EL expressie `${empty klanten}` de waarde `true` terug.

Een element uit een array halen

Als een request attribuut een array bevat, lees je met EL één element uit de array met volgende syntax: `${naamVanHetRequestAttribuut[indexVanHetElement]}`

Als je in een servlet een request attribuut `namen` maakt met een `String` array

```
request.setAttribute("namen", new String[] {"Joe","William","Jack","Averell"});
```

lees je in de bijbehorende JSP het eerste element van de array als

```
${namen[0]}
```

Een element uit een List halen

Je gebruikt dezelfde EL syntax om één element uit een `List` te lezen.

Als je in een servlet een request attribuut `namen` maakt met een `List`

```
List<String> namen = Arrays.asList("Joe", "William", "Jack", "Averell");
request.setAttribute("namen", namen);
```

lees je in de bijbehorende JSP het tweede element van die `List` als

```
${namen[1]}
```

Een element uit een Map halen

Je gebruikt dezelfde EL syntax om de waarde van één entry uit een `Map` te lezen. Je geeft tussen de vierkante haakjes de key mee van de entry die je wilt lezen.

Als je in een servlet een request attribuut `namen` maakt met een `Map`

```
Map<String, String> eigenschappen = new HashMap<String, String>();
eigenschappen.put("Joe", "driftig");
eigenschappen.put("William", "kleurloos");
eigenschappen.put("Jack", "kleurloos");
eigenschappen.put("Averell", "hongerig");
request.setAttribute("eigenschappen", eigenschappen);
```

lees je in de bijbehorende JSP de waarde van de entry met de key `Averell` als

```
${eigenschappen["Averell"]}
```

Als de keys van de `Map` van het type `String` zijn, kan je op een kortere manier de waarde van één entry uit de `Map` lezen

```
${eigenschappen.Averell}
```

Het resultaat van een method oproep

Als je website draait op een webserver die minstens servlets 3.0 ondersteunt (bvb. Tomcat 7), kan je met EL het resultaat van een method oproep lezen.

Als je in een servlet een request attribuut met een `String` object maakt

```
request.setAttribute("familienaam", "daltons");
```

lees je in de bijbehorende JSP de lengte van deze `String` als

```
${familienaam.length()}
```