Working with Loops

Nigel Poulton
Author & Trainer

@nigelpoulton   nigelpoulton.com

Agenda

Syntax

Infinite & while loops

Range loops

Break & continue

Recap

Syntax

```
for <expression> {
}
```

```
for <expression> {
    <code>
}
```

```
for {
    <code>
}
```

`for` without an expression creates an infinite loop

```
for <expression> {
    <code>
}
```

`for` without an expression creates an infinite loop

```
for i < 10 {
    <code>
}
```

`for` without an expression creates an infinite loop

Expression can be any valid Boolean expression

```
courseList := string[]{"courseA", "courseB",
"courseC"}

for i := range courseList {
    <code>
}
```

`for` without an expression creates an infinite loop

Expression can be any valid Boolean expression

`for range` iterates over a list (one entry per iteration of the loop)

```
for i < 10 {
    <code>
}
```

`for` without an expression creates an infinite loop

Expression can be any valid Boolean expression

`for range` iterates over a list (one entry per iteration of the loop)

```
for i := 0; i < 10; i++ {
    <code>
}
```

`for` without an expression creates an infinite loop

Expression can be any valid Boolean expression

`for range` iterates over a list (one entry per iteration of the loop)

Accepts “pre” and “post” statements

i = Docker & Kubernetes: The Big Picture

```
i = Docker & Kubernetes: The Big Picture
j = Docker & Kubernetes: The Big Picture
```

i = Docker & Kubernetes: The Big Picture

j = Docker Deep Dive

i = Docker Networking

```
i = Docker Networking
j = Docker & Kubernetes: The Big Picture
```

```
i = Docker Networking
j = Docker Deep Dive
```

```
for <expression...> {
    <code>
    for <expression...> {
        <code>
        for <expression...> {
            <code>
        }
    }
}
```

```
for <expression...> {
    <code>
    for <expression...> {
        <code>
        for <expression...> {
            <code>
            break
        }
    }
}
```

```
breakPoint:
    for <expression...> {
        <code>
        for <expression...> {
            <code>
            for <expression...> {
                <code>
                break
            }
        }
    }
```

```
breakPoint:
    for <expression...> {
        <code>
        for <expression...> {
            <code>
            for <expression...> {
                <code>
                break breakPoint
            }
        }
    }
```

The “continue” statement

Recap

for

Infinite loops

```
for {
    <code>
}
```

While loops
(Boolean expr)

```
for pre; expr; post {
    <code>
}
```

Range loops

```
for i := range <list> {
    <code>
}
```

```go
for i := 0; <expr>; i++ {
    <code>
}
```

```
for i := 0; <expr>; i++ {
    <code>
    for <expression...> {
        <code>
        for <expression...> {
            <code>
        }
    }
}
```

```
for i := 0; <expr>; i++ {
    <code>
    for <expression...> {
        <code>
        for <expression...> {
            <code>
            break
        }
    }
}
```

```
breakpoint:
    for i := 0; <expr>; i++ {
        <code>
        for <expression...> {
            <code>
            for <expression...> {
                <code>
                break
            }
        }
    }
```

```
breakpoint:
    for i := 0; <expr>; i++ {
        <code>
        for <expression...> {
            <code>
            for <expression...> {
                <code>
                break breakPoint
            }
        }
    }
```

```
for i := 0; <expr>; i++ {
    <code>
    for <expression...> {
        <code>
        for <expression...> {
            <code>
            continue
        }
    }
}
```

Up Next:

Working with Arrays & Slices