Debugging in Eclipse

Finding bugs the easy way

EHSDI

eBuzima

What are bugs?

- It's an error, fault or mistake in a computer program
- Lots of different things cause bugs...
  - Syntax errors (e.g. using '=' instead of '==')
  - Logical errors (e.g. infinite loops)
  - Mathematical operations leading to arithmetic overflow
  - Uninitialized variables
  - Null pointers

Ways to find them

- We can fill our programs with `System.out.println` calls
- Even better we can put lots of DEBUG and TRACE level logging messages into our code

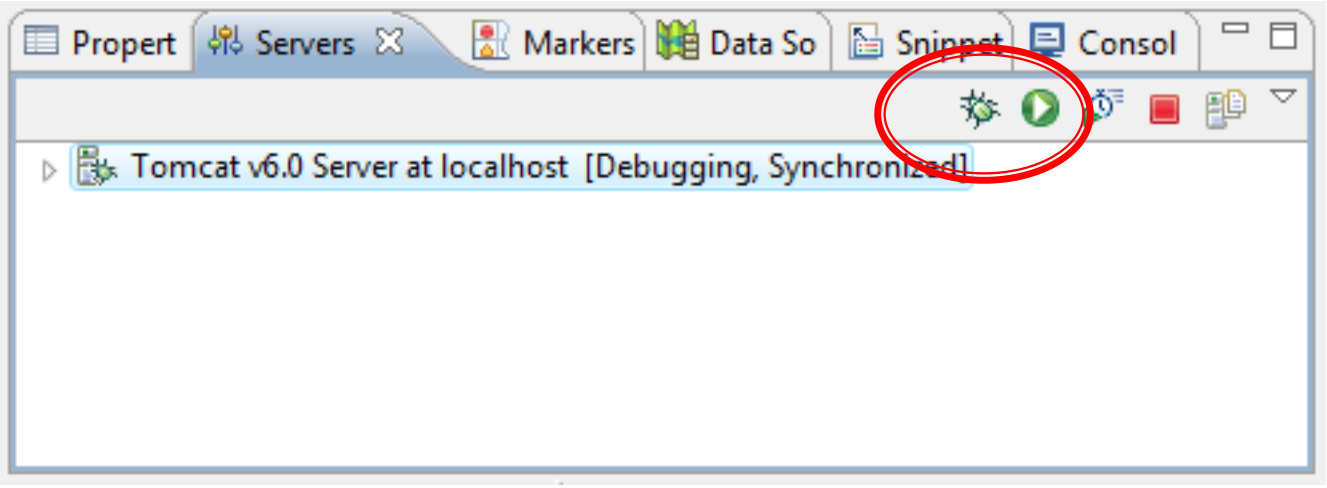
- Problem with debugging this way is that code becomes bloated with logging messages
- Lots of work to write logging messages that display the values of all relevant variables

Debugging

- Most modern languages such as Java allow applications to be run in a *debug* mode
- This means we can watch each statement being executed, and verify that the code is functioning as we intended

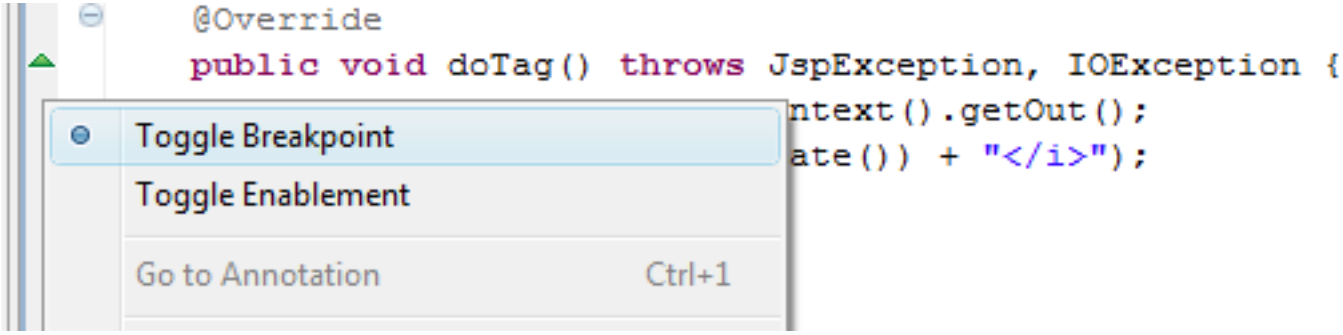
Debugging web applications

- To launch a web application or servlet in debug mode, right-click on it and choose *Debug As → Debug on Server*
- Or switch the server into debug mode…



Breakpoints

- These are markers which tell the JVM to pause the program's execution
- While the program is paused, we can inspect the values of variables to see if the code is working as we intended



Breakpoints

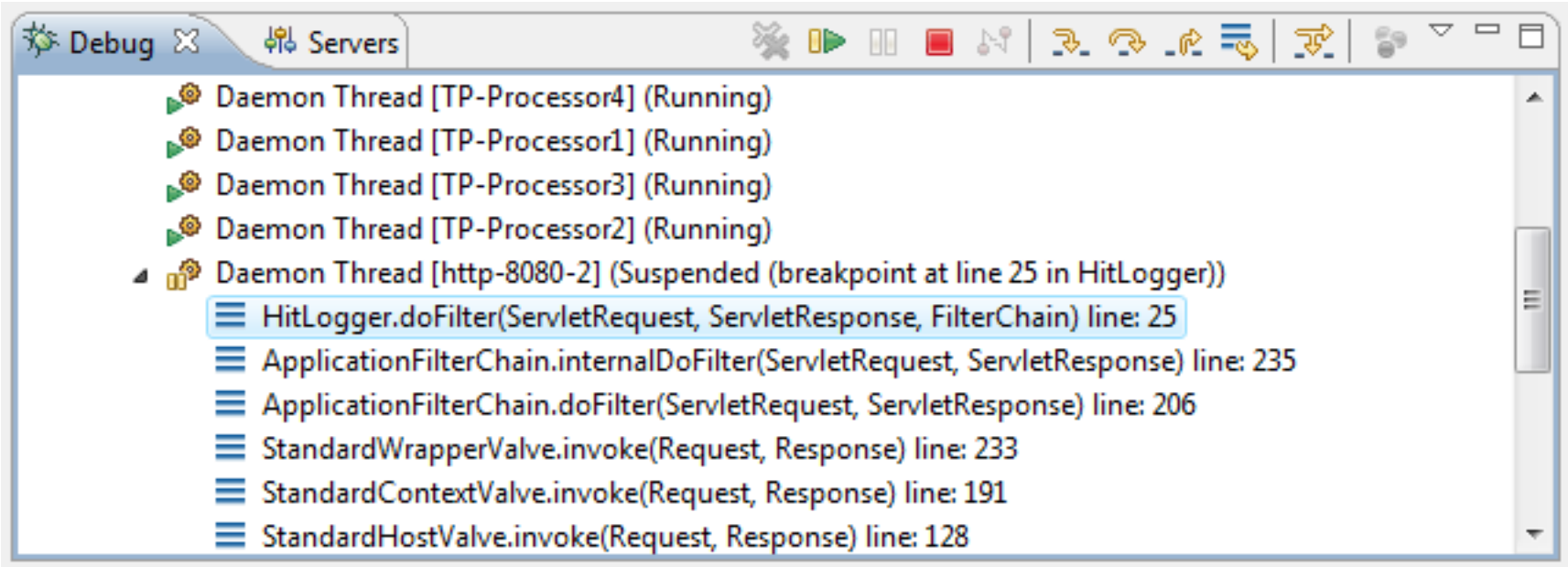
- When Eclipse hits a breakpoint, it will:
  - Open the Debug perspective (optional)
  - Highlight the current line

```
Visit visit = new Visit(new Date(), httpReq.getRequestURI(), request.
Visit.record(visit);

chain.doFilter(request, response);
```

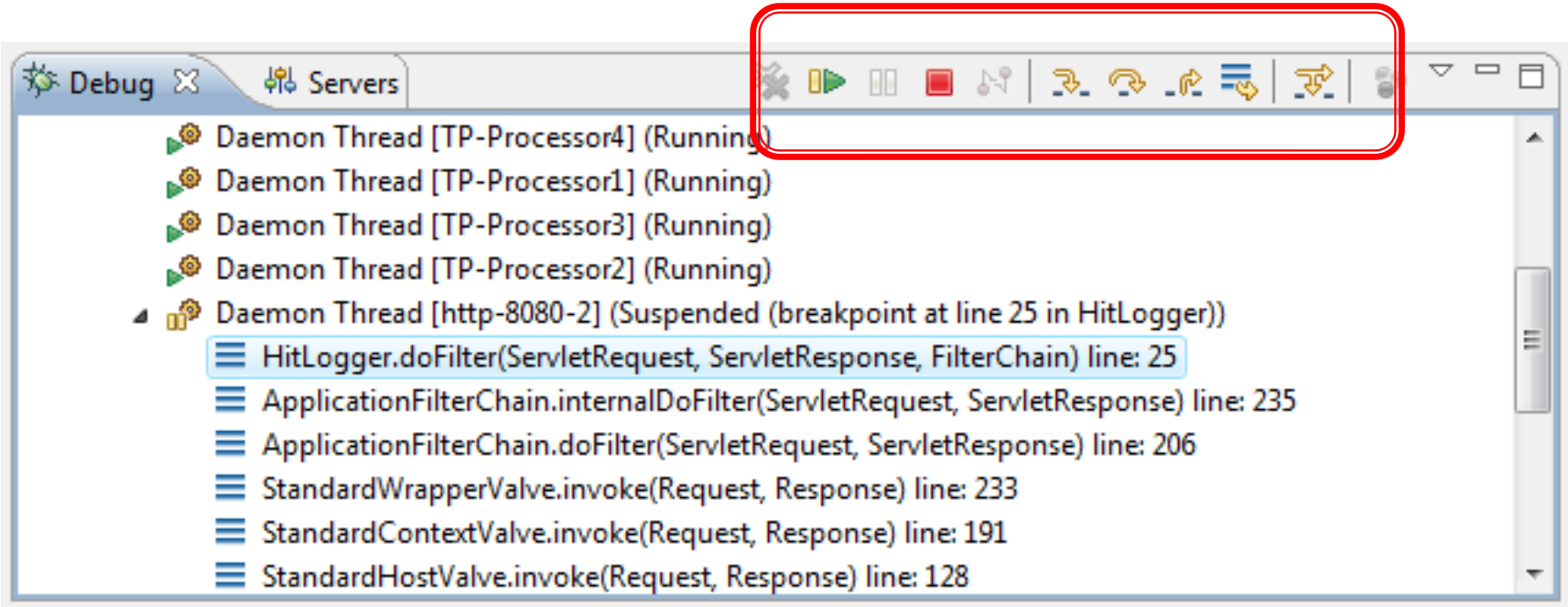
The debug view

- The debug view lists the threads being debugged. It tells us:
  - Which thread reached the breakpoint
  - The *call stack* of that thread – i.e. it's history of method calls



Controlling execution

- The debug view also provides controls to
  - Resume the program execution
  - Stop the program (i.e. end execution / debugging)
  - Step through the code

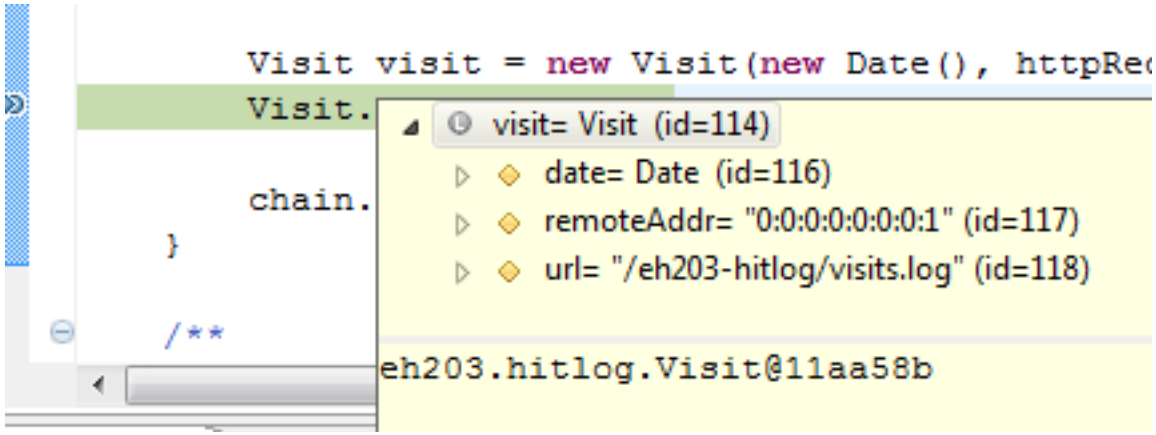


Stepping through code

Icon	Name	Key	Description
	Step Into	F5	If current line is a method call, debugger will move to the first line of that method
	Step Over	F6	If current line is a method call, debugger executes method and moves to next line
	Step Return	F7	Debugger finishes execution of current method, and moves to line of code that called the method
	Drop to Frame		'Rewinds' the program back to start of the current stack frame

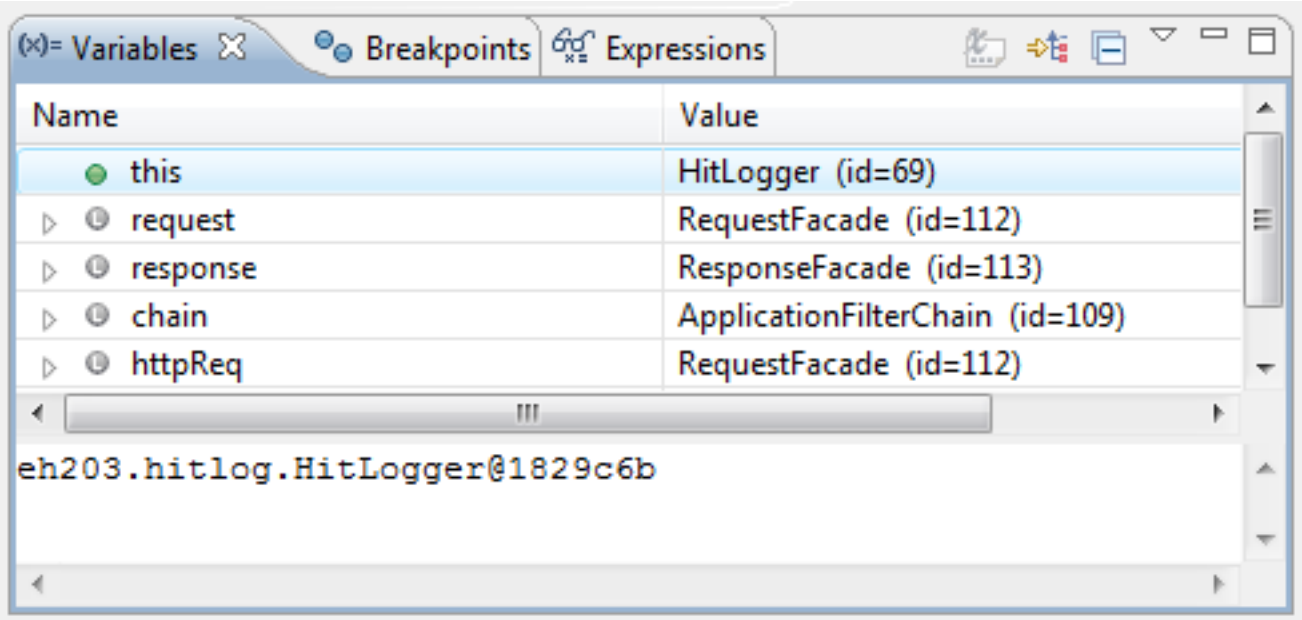
Inspecting variables

- We can inspect the variables within the code to see what values they have
- Hovering the mouse over a variable shows its value/members in a popup



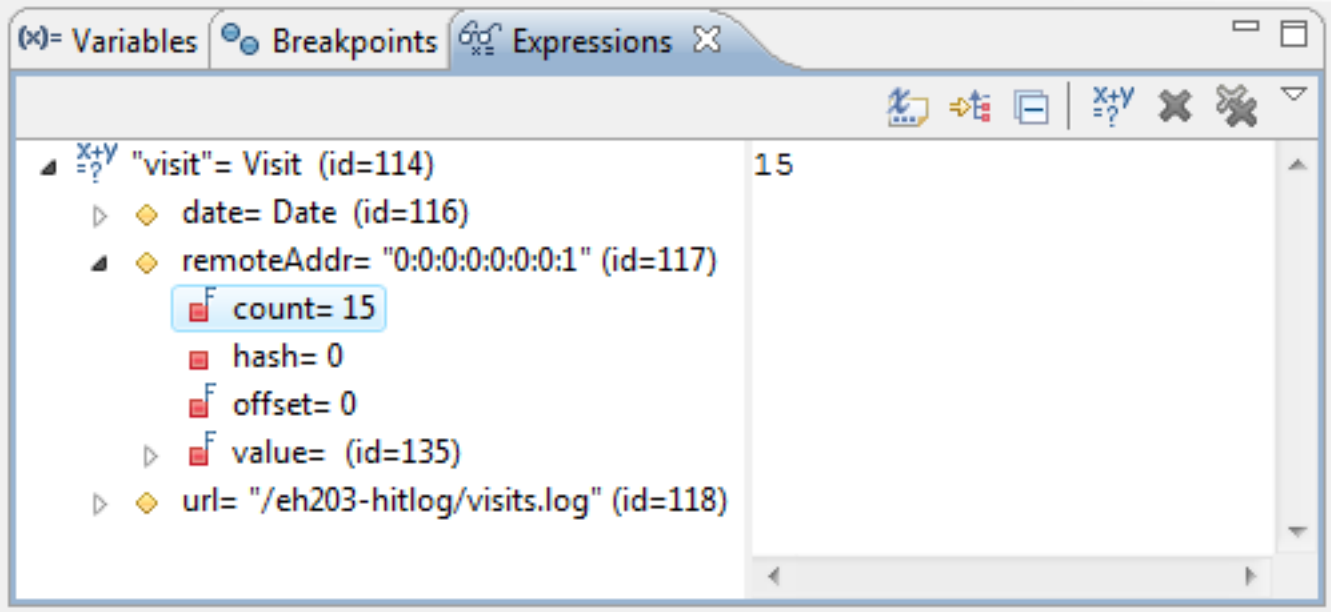
The variables view

- This lists all of the variables in the current scope
- We can even change values of variables



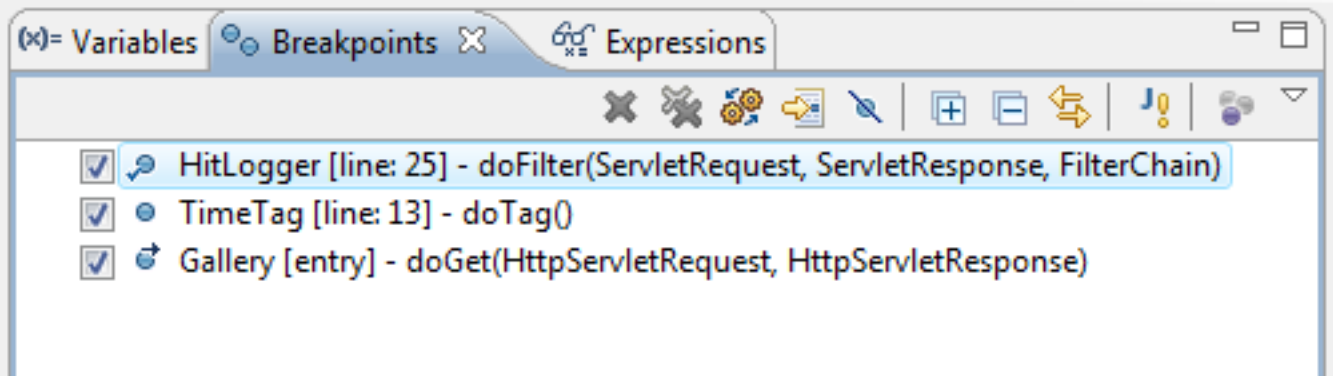
The expressions view

- Variables in the variables view can be 'watched' which means they are added to the expressions view



The breakpoints view

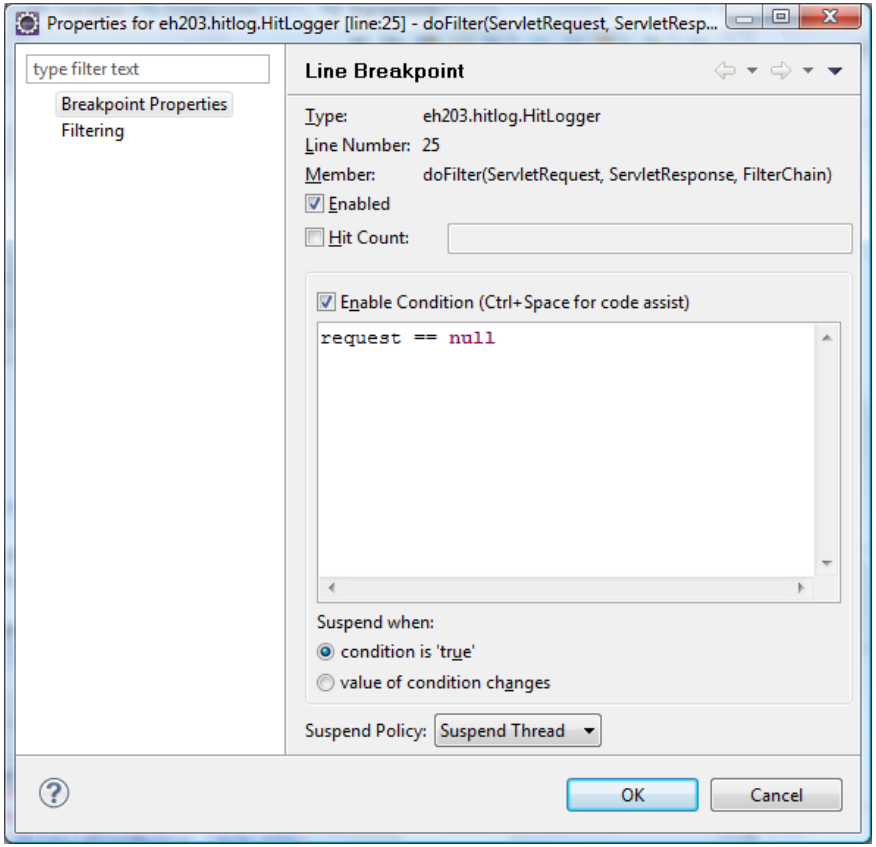
- This lists all the breakpoints in the workspace
- Makes it easy to enable and disable them



- And edit a breakpoint's properties...

Breakpoint properties

- This dialog lets us specify a condition for a breakpoint
- We can tell eclipse to only suspend execution when the condition is true



References

- Websites
  - http://www.ibm.com/developerworks/library/os-ecbug/