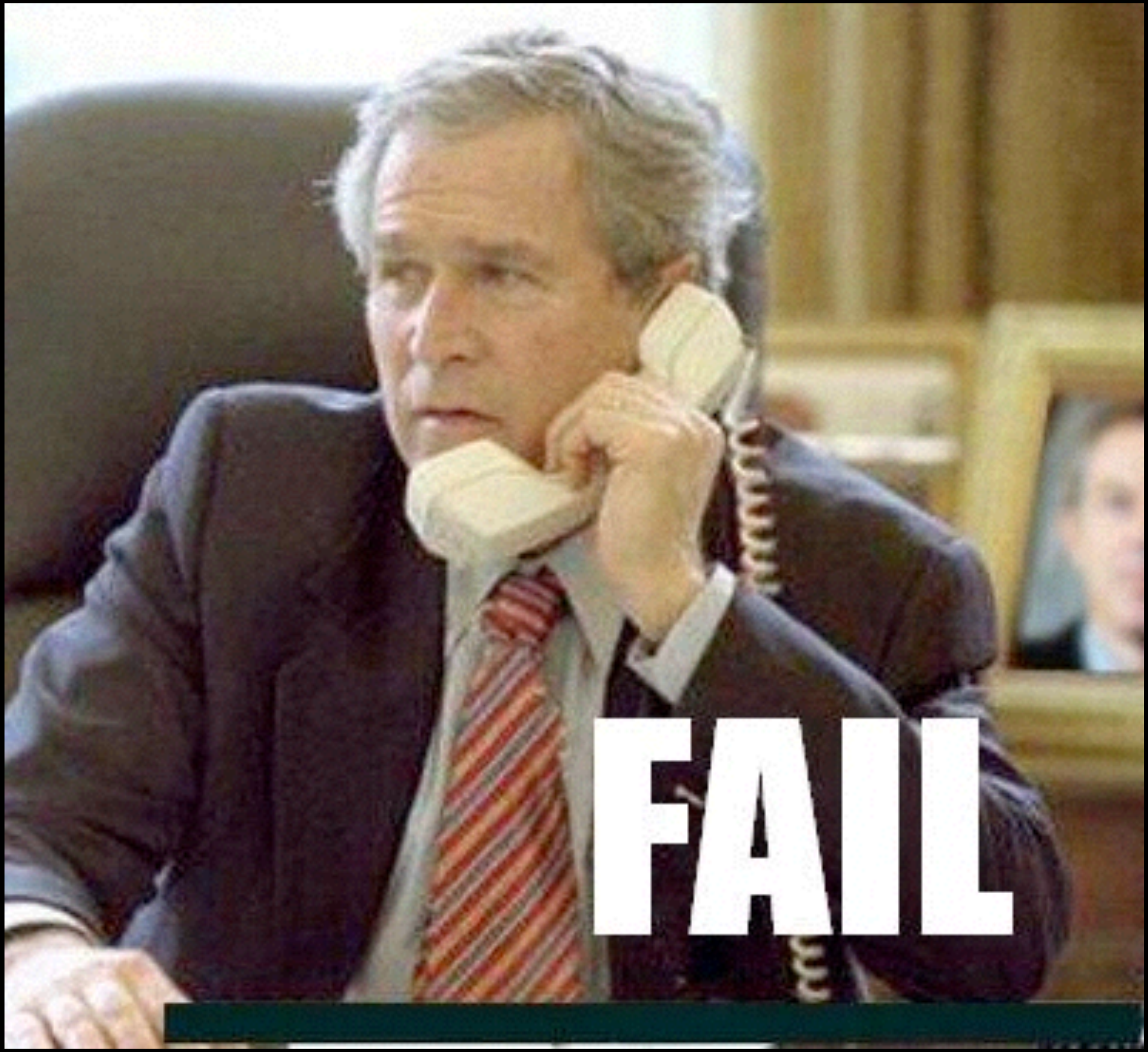
Martin Lippert
Principal Software Engineer - Pivotal
mlippert@gopivotal.com
@martinlippert

Optimizing Performance

how to make your Eclipse-based tools run faster

Every developer benefits from better performance

Find out where the problem is...



failblog.com

Measure !!!

200
180
160
140
120
40
60
80
100
AIRSPEED
KNOTS
PRESS ALT
TEMP
VACUUM
9
8
6
D.C. ELEC

VisualVM

Free
Easy to use
comes as part of the JDK
extremely useful to capture data remotely

YourKit

used for comprehensive analysis

various options and ways to track down issues

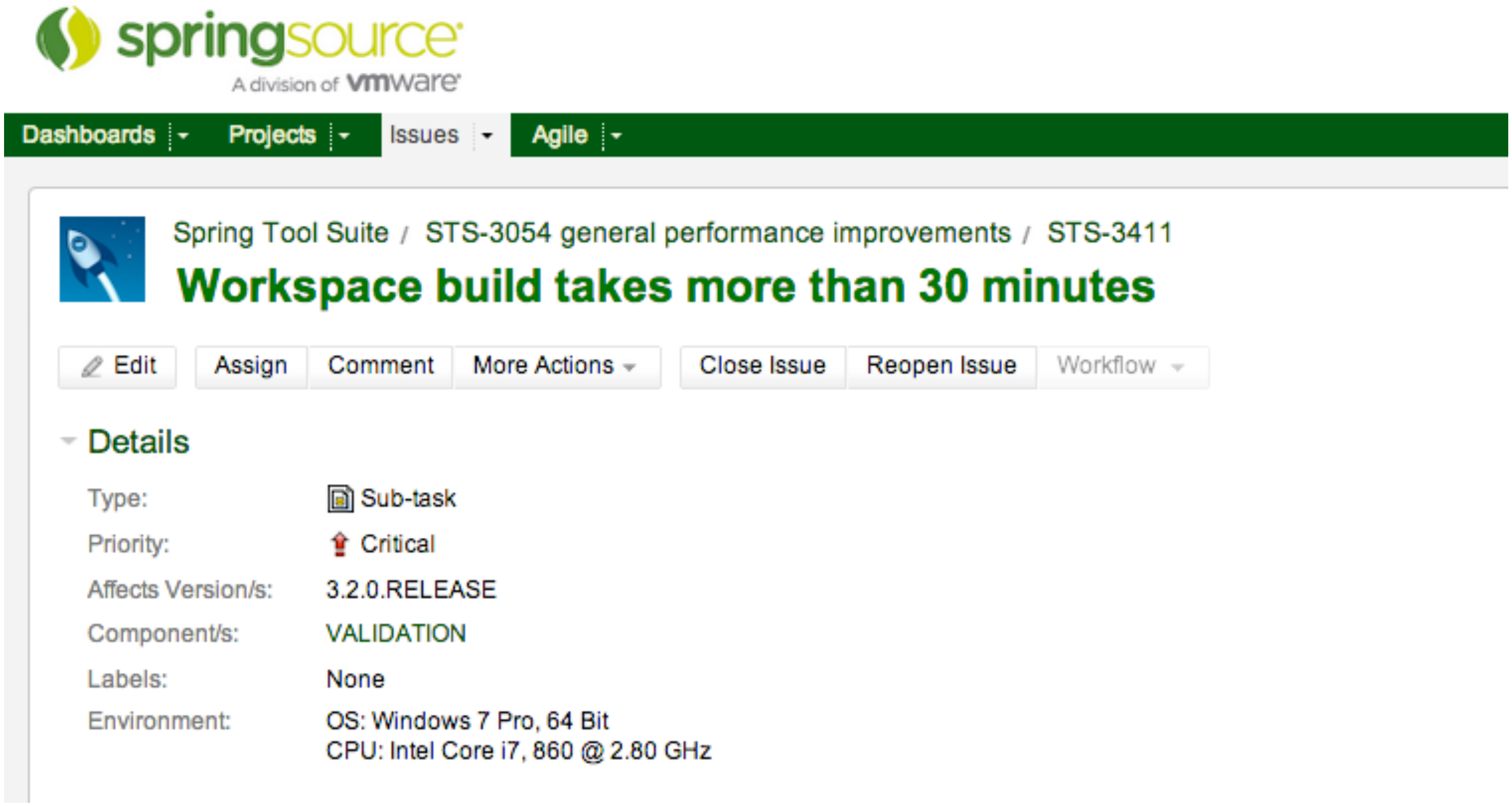
$$$

alternative:

JProfiler

$$$

the case



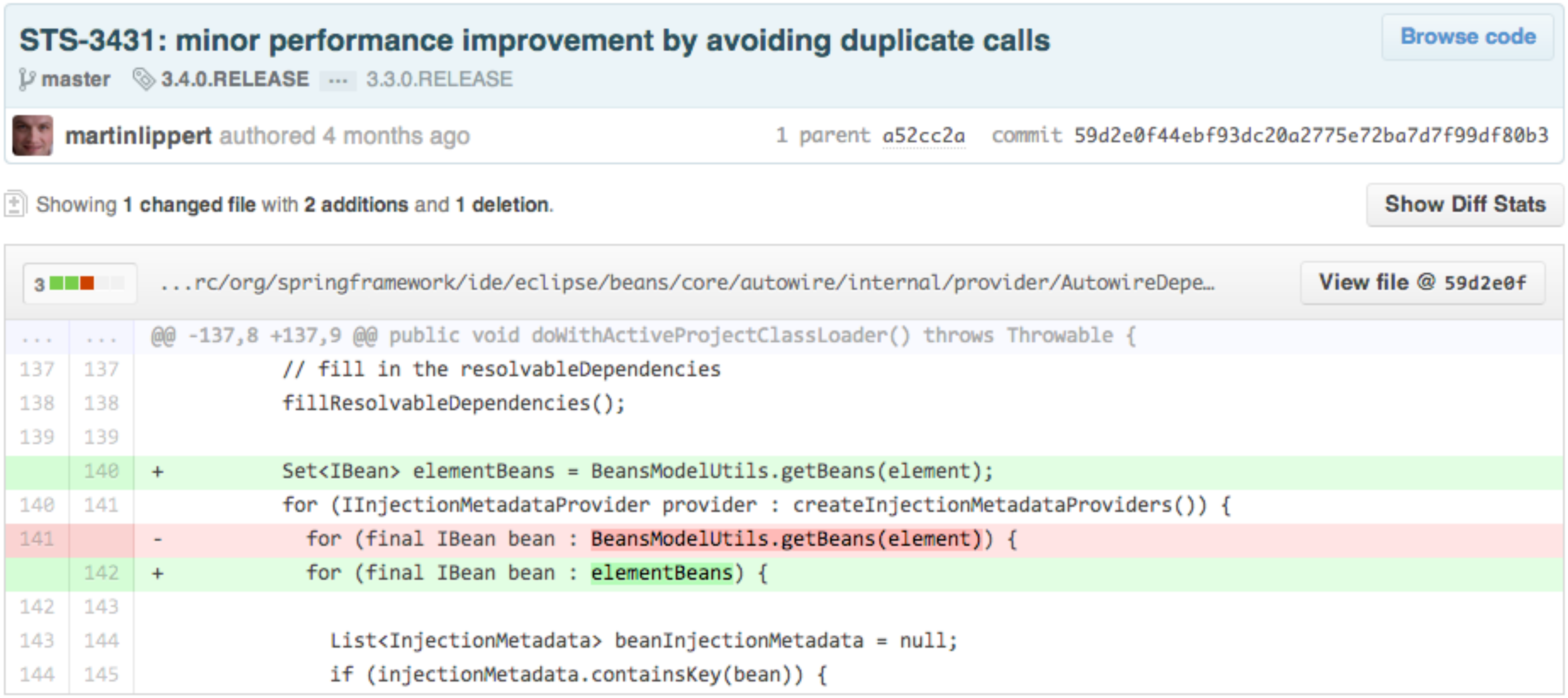
Spring Tool Suite / STS-3054 general performance improvements / STS-3411

Workspace build takes more than 30 minutes

Details

Type:	Sub-task
Priority:	Critical
Affects Version/s:	3.2.0.RELEASE
Component/s:	VALIDATION
Labels:	None
Environment:	OS: Windows 7 Pro, 64 Bit CPU: Intel Core i7, 860 @ 2.80 GHz

trivial: expensive calls inside loops



STS-3431: minor performance improvement by avoiding duplicate calls

Browse code

master 3.4.0.RELEASE ... 3.3.0.RELEASE

martinlippert authored 4 months ago

1 parent a52cc2a commit 59d2e0f44ebf93dc20a2775e72ba7d7f99df80b3

Showing 1 changed file with 2 additions and 1 deletion.

Show Diff Stats

3 ...rc/org/springframework/ide/eclipse/beans/core/autowire/internal/provider/AutowireDepe...

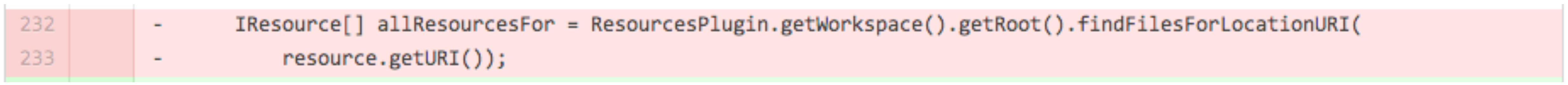
View file @ 59d2e0f

```
@@ -137,8 +137,9 @@ public void doWithActiveProjectClassLoader() throws Throwable {
            // fill in the resolvableDependencies
            fillResolvableDependencies();

+           Set<IBean> elementBeans = BeansModelUtils.getBeans(element);
            for (IInjectionMetadataProvider provider : createInjectionMetadataProviders()) {
-             for (final IBean bean : BeansModelUtils.getBeans(element)) {
+             for (final IBean bean : elementBeans) {

                List<InjectionMetadata> beanInjectionMetadata = null;
                if (injectionMetadata.containsKey(bean)) {
```

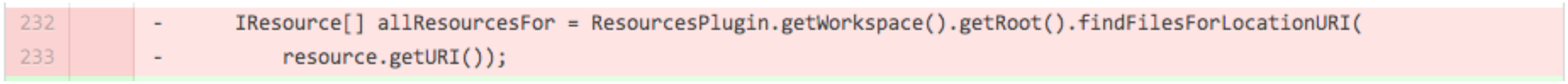
`findFilesForLocationURI(..)` is slow



```
-       IResource[] allResourcesFor = ResourcesPlugin.getWorkspace().getRoot().findFilesForLocationURI(
-           resource.getURI());
```

step 1: fix this in the Eclipse platform

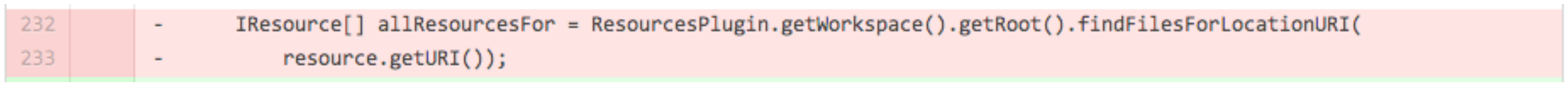
`findFilesForLocationURI(..)` is slow



```
-       IResource[] allResourcesFor = ResourcesPlugin.getWorkspace().getRoot().findFilesForLocationURI(
-           resource.getURI());
```

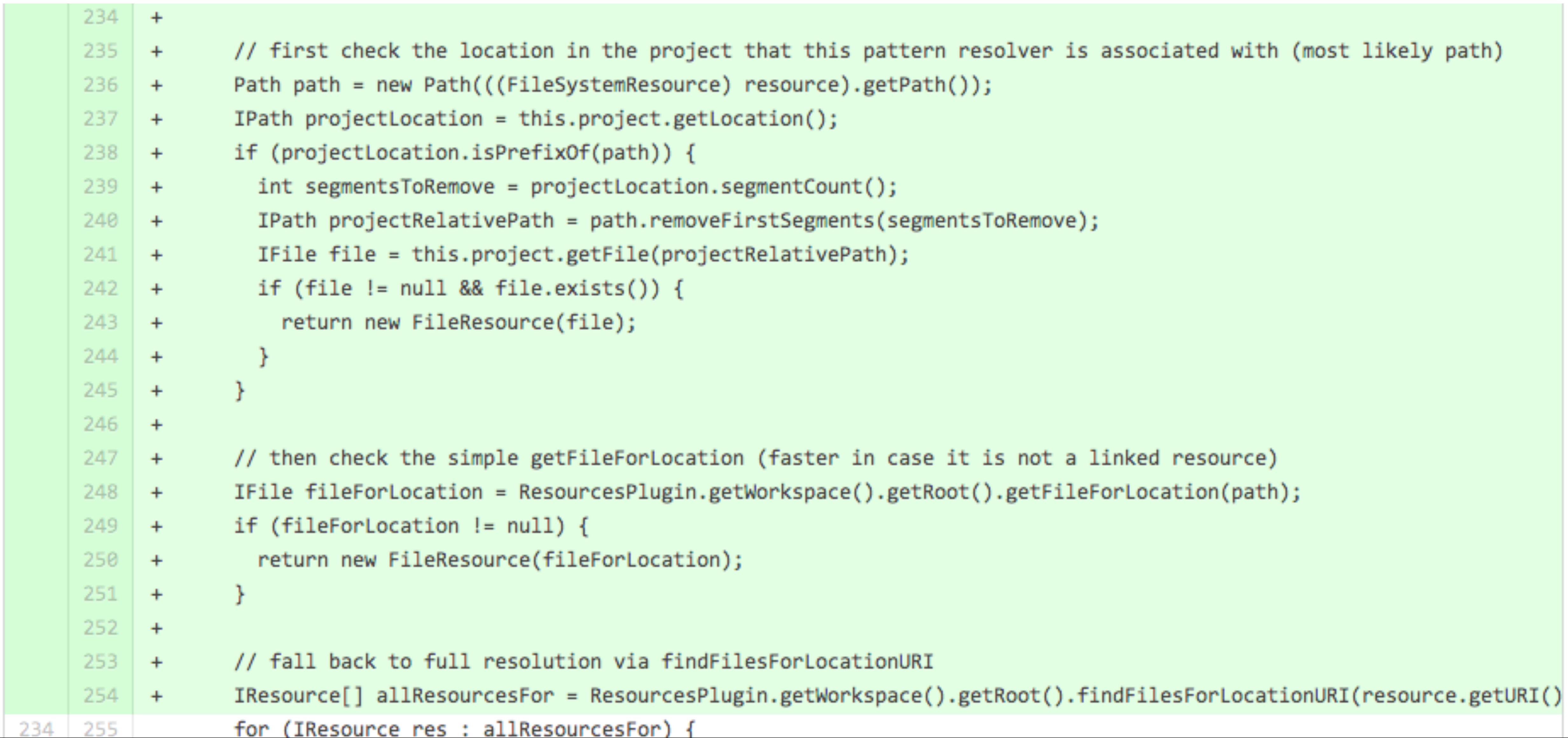
step 2: cache results if that makes sense

`findFilesForLocationURI(..)` is slow



```
-      IResource[] allResourcesFor = ResourcesPlugin.getWorkspace().getRoot().findFilesForLocationURI(
-          resource.getURI());
```

step 3: if you can‘t avoid massive use of this, optimize for the most likely case



```
+
+      // first check the location in the project that this pattern resolver is associated with (most likely path)
+      Path path = new Path(((FileSystemResource) resource).getPath());
+      IPath projectLocation = this.project.getLocation();
+      if (projectLocation.isPrefixOf(path)) {
+        int segmentsToRemove = projectLocation.segmentCount();
+        IPath projectRelativePath = path.removeFirstSegments(segmentsToRemove);
+        IFile file = this.project.getFile(projectRelativePath);
+        if (file != null && file.exists()) {
+          return new FileResource(file);
+        }
+      }
+
+      // then check the simple getFileForLocation (faster in case it is not a linked resource)
+      IFile fileForLocation = ResourcesPlugin.getWorkspace().getRoot().getFileForLocation(path);
+      if (fileForLocation != null) {
+        return new FileResource(fileForLocation);
+      }
+
+      // fall back to full resolution via findFilesForLocationURI
+      IResource[] allResourcesFor = ResourcesPlugin.getWorkspace().getRoot().findFilesForLocationURI(resource.getURI()
       for (IResource res : allResourcesFor) {
```

Build workspace (16%)...

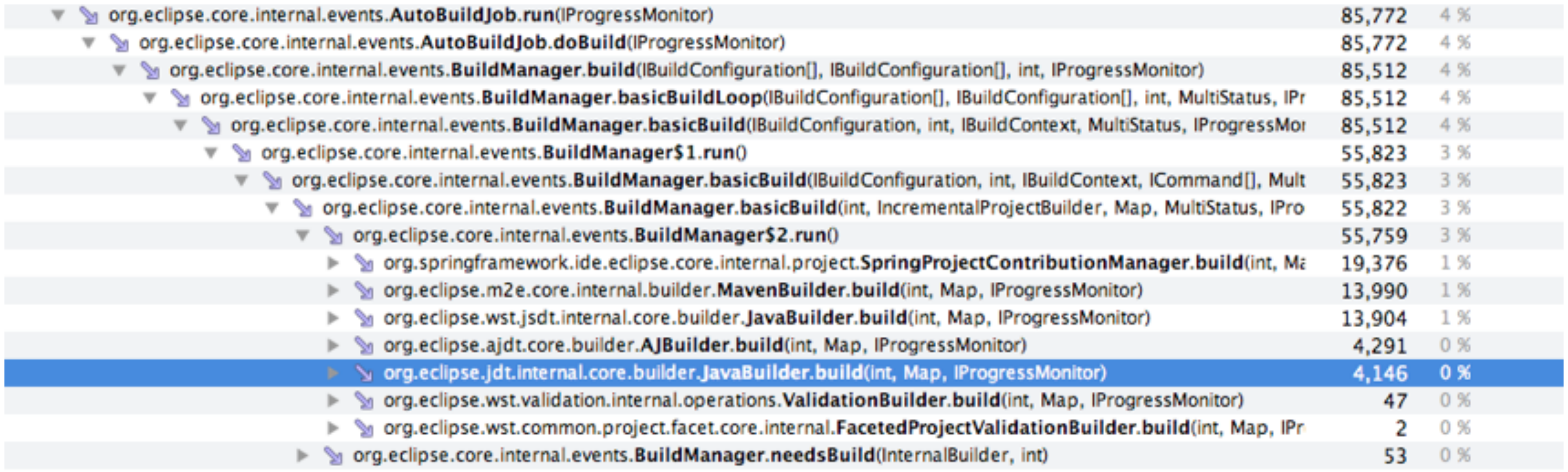
why is the build taking
sooooo long... ???

Build workspace (16%)...

Build workspace (16%)...

taken from a different case

what is exactly going on under the hood?

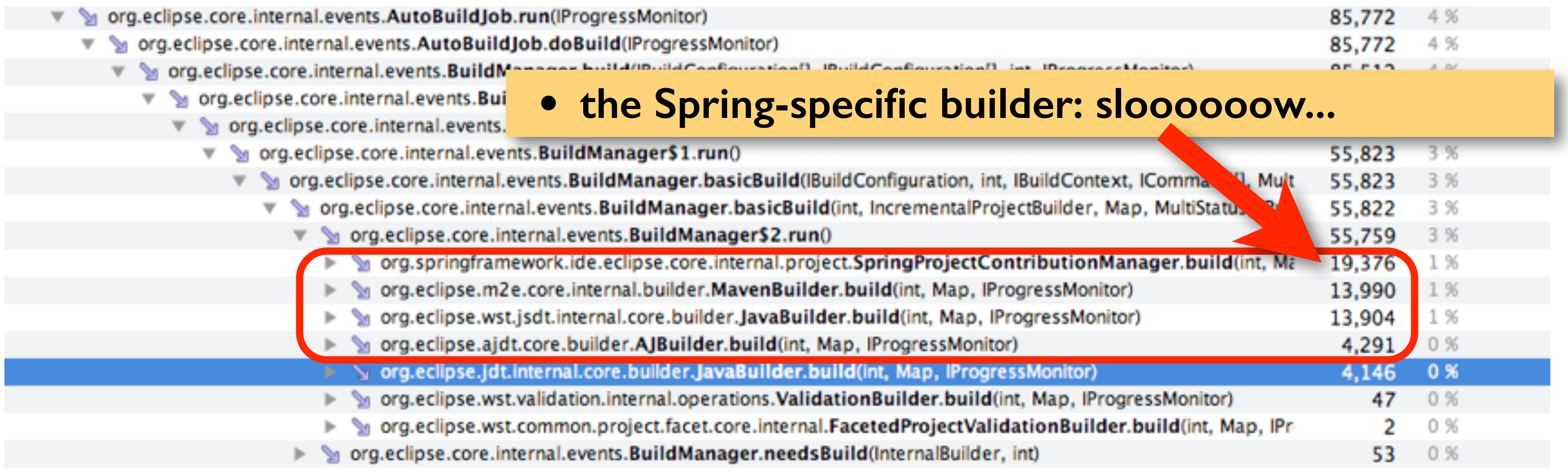


```
▼ org.eclipse.core.internal.events.AutoBuildJob.run(IProgressMonitor)                                                85,772   4 %
  ▼ org.eclipse.core.internal.events.AutoBuildJob.doBuild(IProgressMonitor)                                          85,772   4 %
    ▼ org.eclipse.core.internal.events.BuildManager.build(IBuildConfiguration[], IBuildConfiguration[], int, IProgressMonitor)   85,512   4 %
      ▼ org.eclipse.core.internal.events.BuildManager.basicBuildLoop(IBuildConfiguration[], IBuildConfiguration[], int, MultiStatus, IPr   85,512   4 %
        ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, MultiStatus, IProgressMo   85,512   4 %
          ▼ org.eclipse.core.internal.events.BuildManager$1.run()                                                    55,823   3 %
            ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, ICommand[], Mult   55,823   3 %
              ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(int, IncrementalProjectBuilder, Map, MultiStatus, IPro   55,822   3 %
                ▼ org.eclipse.core.internal.events.BuildManager$2.run()                                              55,759   3 %
                  ▶ org.springframework.ide.eclipse.core.internal.project.SpringProjectContributionManager.build(int, Ma   19,376   1 %
                  ▶ org.eclipse.m2e.core.internal.builder.MavenBuilder.build(int, Map, IProgressMonitor)             13,990   1 %
                  ▶ org.eclipse.wst.jsdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)         13,904   1 %
                  ▶ org.eclipse.ajdt.core.builder.AJBuilder.build(int, Map, IProgressMonitor)                         4,291   0 %
                  ▶ org.eclipse.jdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)               4,146   0 %
                  ▶ org.eclipse.wst.validation.internal.operations.ValidationBuilder.build(int, Map, IProgressMonitor)   47   0 %
                  ▶ org.eclipse.wst.common.project.facet.core.internal.FacetedProjectValidationBuilder.build(int, Map, IPr   2   0 %
                ▶ org.eclipse.core.internal.events.BuildManager.needsBuild(InternalBuilder, int)                         53   0 %
```

Build workspace (16%)...

taken from a different case

what is exactly going on under the hood?



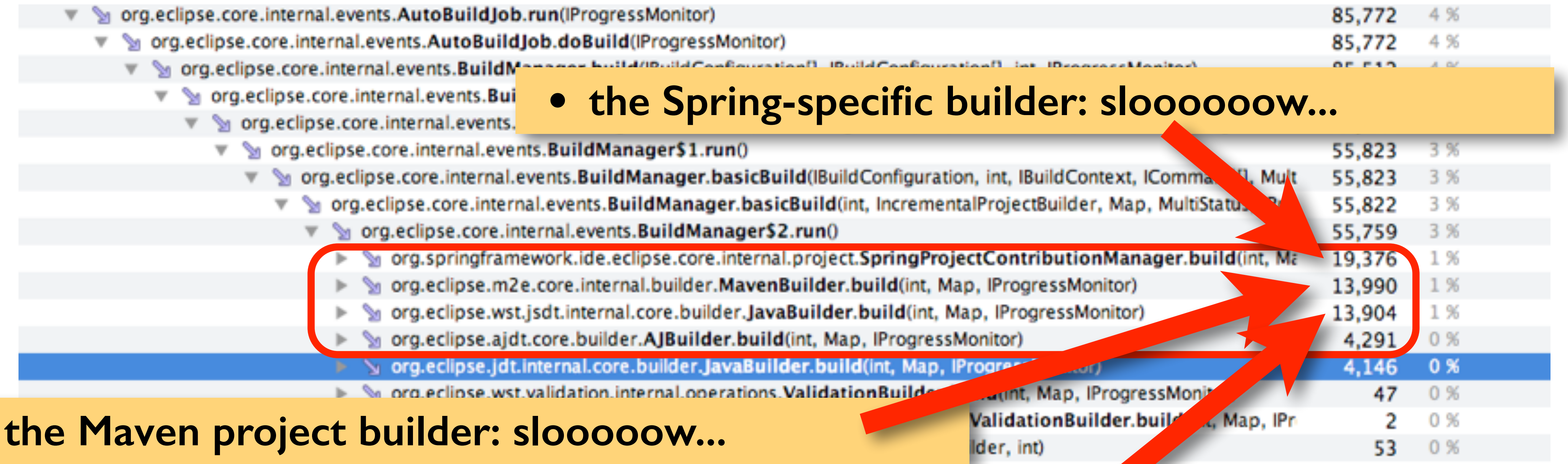
- the Spring-specific builder: sloooooow...

```
▼ org.eclipse.core.internal.events.AutoBuildJob.run(IProgressMonitor)                                              85,772  4 %
  ▼ org.eclipse.core.internal.events.AutoBuildJob.doBuild(IProgressMonitor)                                        85,772  4 %
    ▼ org.eclipse.core.internal.events.BuildM
      ▼ org.eclipse.core.internal.events.Bui
        ▼ org.eclipse.core.internal.events.
          ▼ org.eclipse.core.internal.events.BuildManager$1.run()                                                  55,823  3 %
            ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, IComma    Mult  55,823  3 %
              ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(int, IncrementalProjectBuilder, Map, MultiStatu   55,822  3 %
                ▼ org.eclipse.core.internal.events.BuildManager$2.run()                                            55,759  3 %
                  ▶ org.springframework.ide.eclipse.core.internal.project.SpringProjectContributionManager.build(int, Ma  19,376  1 %
                  ▶ org.eclipse.m2e.core.internal.builder.MavenBuilder.build(int, Map, IProgressMonitor)           13,990  1 %
                  ▶ org.eclipse.wst.jsdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)       13,904  1 %
                  ▶ org.eclipse.ajdt.core.builder.AJBuilder.build(int, Map, IProgressMonitor)                       4,291  0 %
                  ▶ org.eclipse.jdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)             4,146  0 %
                  ▶ org.eclipse.wst.validation.internal.operations.ValidationBuilder.build(int, Map, IProgressMonitor)    47  0 %
                  ▶ org.eclipse.wst.common.project.facet.core.internal.FacetedProjectValidationBuilder.build(int, Map, IPr    2  0 %
                ▶ org.eclipse.core.internal.events.BuildManager.needsBuild(InternalBuilder, int)                       53  0 %
```

Build workspace (16%)...

taken from a different case

what is exactly going on under the hood?



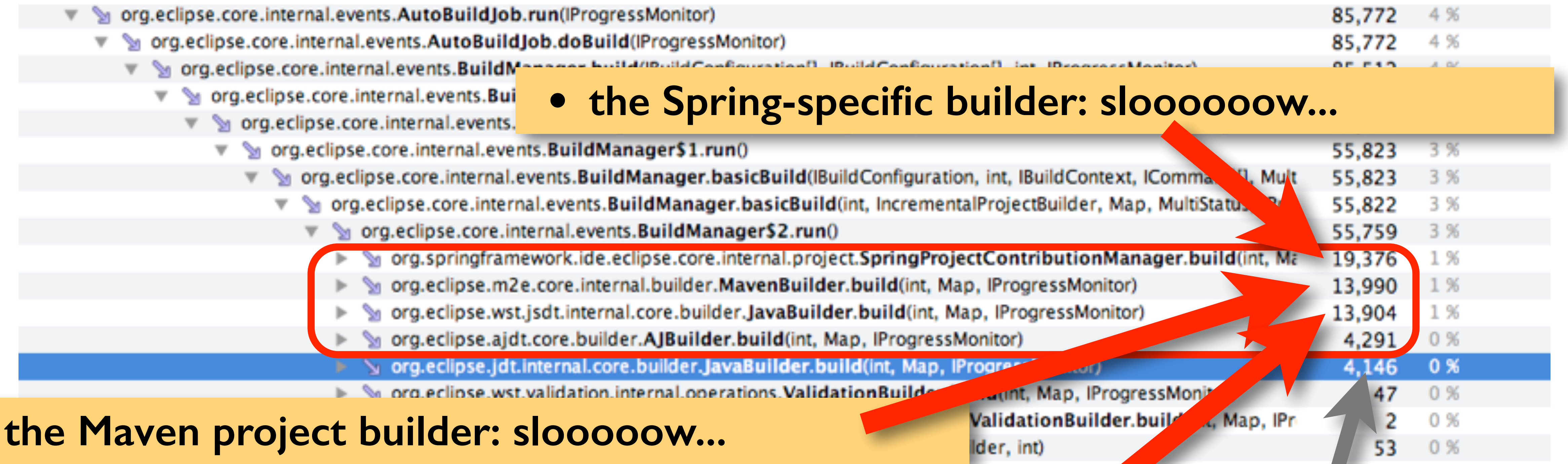
```
▼ org.eclipse.core.internal.events.AutoBuildJob.run(IProgressMonitor)                              85,772   4 %
  ▼ org.eclipse.core.internal.events.AutoBuildJob.doBuild(IProgressMonitor)                        85,772   4 %
    ▼ org.eclipse.core.internal.events.BuildM
      ▼ org.eclipse.core.internal.events.Bui
        ▼ org.eclipse.core.internal.events.
          ▼ org.eclipse.core.internal.events.BuildManager$1.run()                                  55,823   3 %
            ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, IComma    , Mult   55,823   3 %
              ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(int, IncrementalProjectBuilder, Map, MultiStatu   55,822   3 %
                ▼ org.eclipse.core.internal.events.BuildManager$2.run()                            55,759   3 %
                  ▶ org.springframework.ide.eclipse.core.internal.project.SpringProjectContributionManager.build(int, Ma   19,376   1 %
                  ▶ org.eclipse.m2e.core.internal.builder.MavenBuilder.build(int, Map, IProgressMonitor)   13,990   1 %
                  ▶ org.eclipse.wst.jsdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)   13,904   1 %
                  ▶ org.eclipse.ajdt.core.builder.AJBuilder.build(int, Map, IProgressMonitor)       4,291   0 %
                  ▶ org.eclipse.jdt.internal.core.builder.JavaBuilder.build(int, Map, IProgre       4,146   0 %
                  ▶ org.eclipse.wst.validation.internal.operations.ValidationBuild     (int, Map, IProgressMoni   47   0 %
                                                     ValidationBuilder.buil     , Map, IPr          2   0 %
                                                     lder, int)                                    53   0 %
```

- the Spring-specific builder: slooooow...
- the Maven project builder: slooooow...
- the WTP JS builder: slooooow...

Build workspace (16%)...

taken from a different case

what is exactly going on under the hood?



```
▼ org.eclipse.core.internal.events.AutoBuildJob.run(IProgressMonitor)                                                  85,772  4 %
  ▼ org.eclipse.core.internal.events.AutoBuildJob.doBuild(IProgressMonitor)                                            85,772  4 %
    ▼ org.eclipse.core.internal.events.BuildManager.build(IBuildConfiguration[], IBuildConfiguration[], int, IProgressMonitor)  85,513  4 %
      ▼ org.eclipse.core.internal.events.Bui
        ▼ org.eclipse.core.internal.events.
          ▼ org.eclipse.core.internal.events.BuildManager$1.run()                                                      55,823  3 %
            ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, IComma  Mult  55,823  3 %
              ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(int, IncrementalProjectBuilder, Map, MultiStatu  55,822  3 %
                ▼ org.eclipse.core.internal.events.BuildManager$2.run()                                                55,759  3 %
                  ▶ org.springframework.ide.eclipse.core.internal.project.SpringProjectContributionManager.build(int, Ma  19,376  1 %
                  ▶ org.eclipse.m2e.core.internal.builder.MavenBuilder.build(int, Map, IProgressMonitor)               13,990  1 %
                  ▶ org.eclipse.wst.jsdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)           13,904  1 %
                  ▶ org.eclipse.ajdt.core.builder.AJBuilder.build(int, Map, IProgressMonitor)                           4,291  0 %
                  ▶ org.eclipse.jdt.internal.core.builder.JavaBuilder.build(int, Map, IProgre                           4,146  0 %
                  ▶ org.eclipse.wst.validation.internal.operations.ValidationBuild  (int, Map, IProgressMoni              47  0 %
                                                                ValidationBuilder.buil  , Map, IPr                       2  0 %
                                                                lder, int)                                              53  0 %
```

- the Spring-specific builder: slooooow...

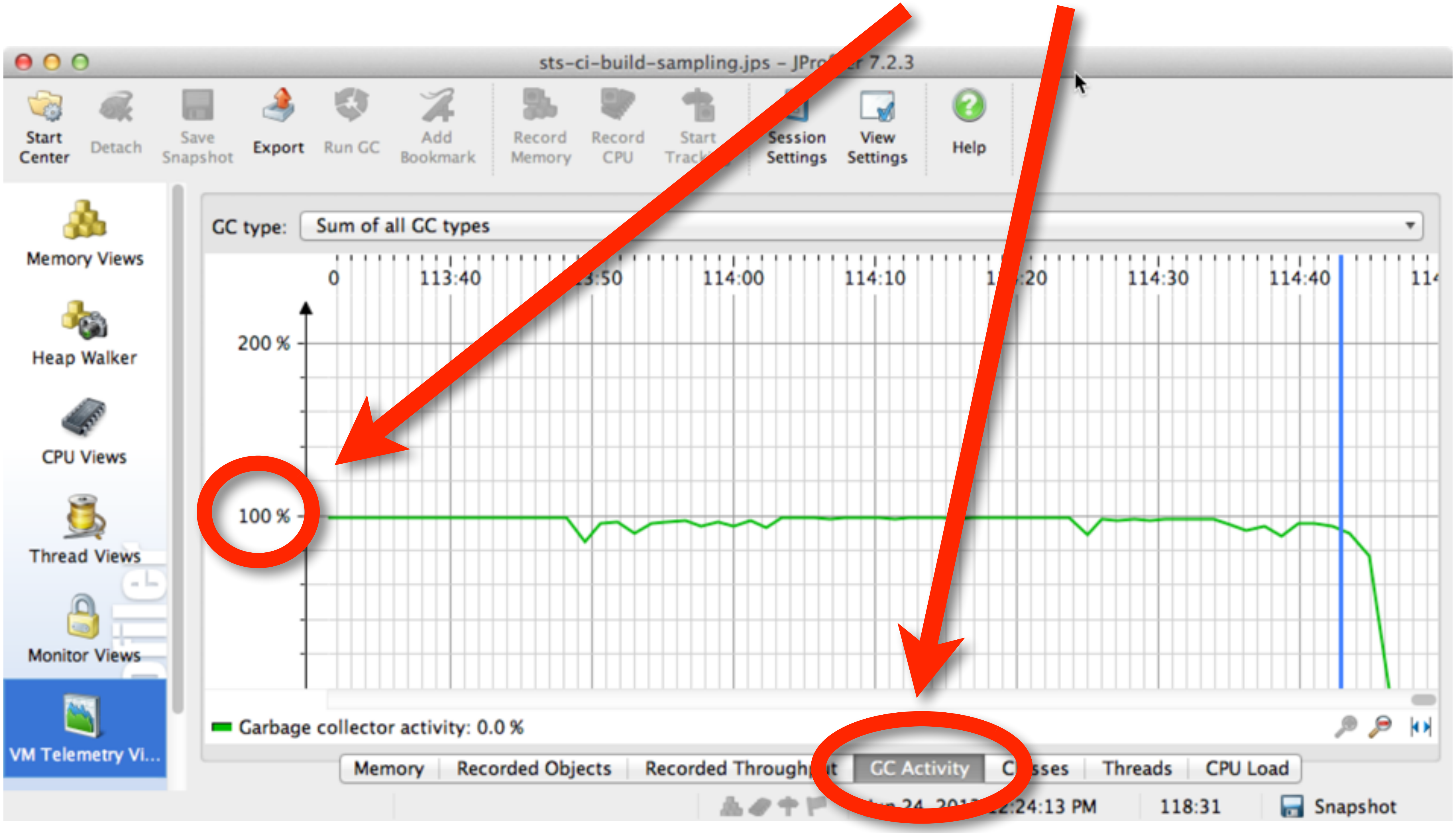
- the Maven project builder: slooooow...

- the WTP JS builder: slooooow...

- the core implementation is ultra fast (compiling Java, for example, but also reconciling, invoking content assist, etc.)

But wait a moment...

But wait a moment... what is this ?!?

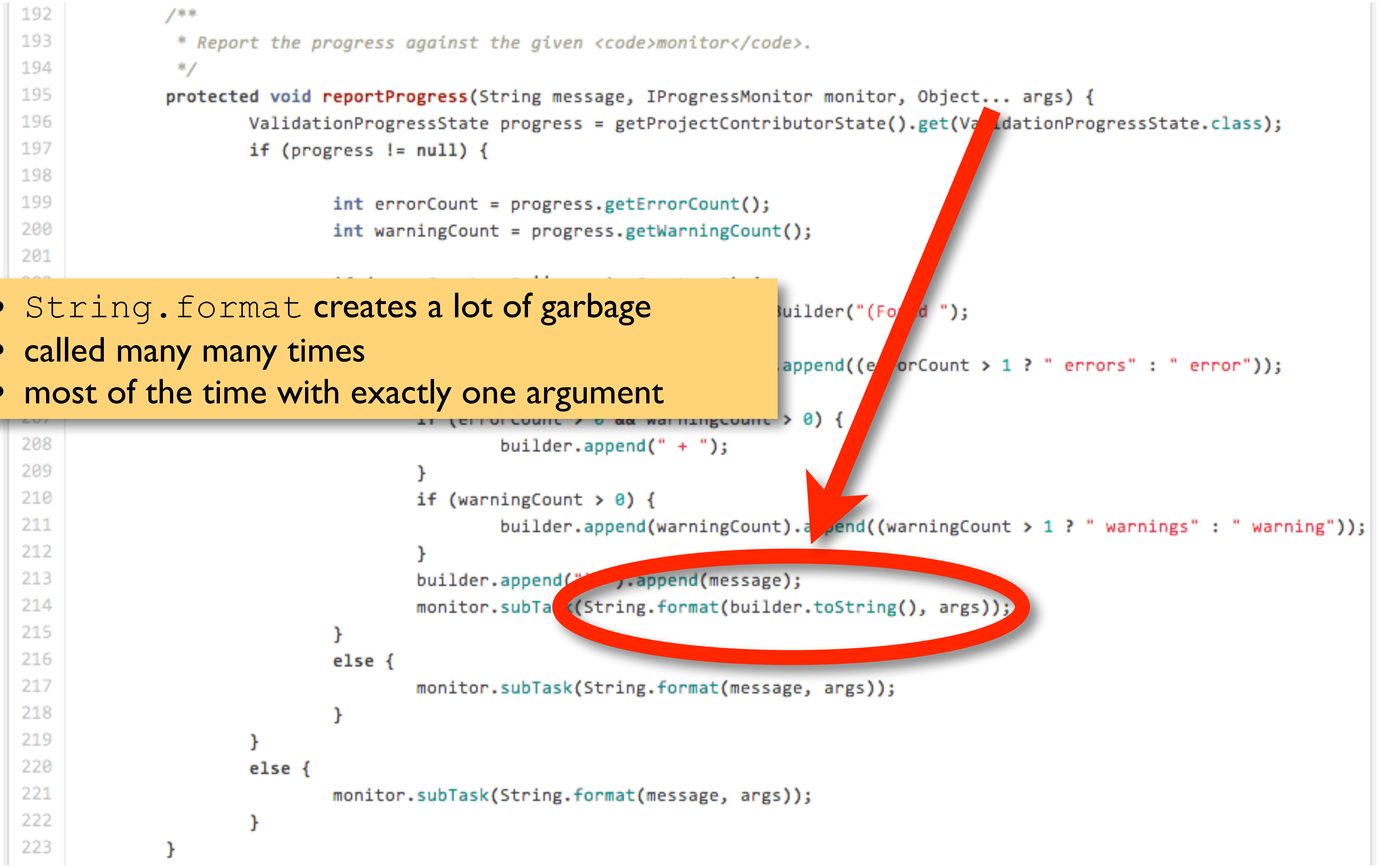


Action 1: Configure your Eclipse wisely

max heap size	build workspace
-Xmx768m	**30min**
↓	↓
-Xmx1024m	**~7min**

Action 2: Reduce garbage and memory usage in general

- `String.format` creates a lot of garbage
- called many many times
- most of the time with exactly one argument



```
        /**
         * Report the progress against the given <code>monitor</code>.
         */
        protected void reportProgress(String message, IProgressMonitor monitor, Object... args) {
                ValidationProgressState progress = getProjectContributorState().get(Va idationProgressState.class);
                if (progress != null) {

                        int errorCount = progress.getErrorCount();
                        int warningCount = progress.getWarningCount();

[illegible]
                                                        Builder("(Fo  d ");
[illegible]
                                                        .append((e  orCount > 1 ? " errors" : " error"));
[illegible]
                                if (errorCount > 0 && warningCount > 0) {
                                        builder.append(" + ");
                                }
                                if (warningCount > 0) {
                                        builder.append(warningCount).a  end((warningCount > 1 ? " warnings" : " warning"));
                                }
                                builder.append("   ).append(message);
                                monitor.subTa (String.format(builder.toString(), args));
                        }
                        else {
                                monitor.subTask(String.format(message, args));
                        }
                }
                else {
                        monitor.subTask(String.format(message, args));
                }
        }
```

Action 2: Reduce garbage and memory usage in general

```
public Set<IBean> getBeans() {
    Set<IBean> allBeans = new LinkedHashSet<IBean>(beans.values());
    for (IBeansImport beansImport : imports) {
        for (IBeansConfig bc : beansImport.getImportedBeansConfigs()) {
            allBeans.addAll(bc.getBeans());
        }
    }
    return Collections.unmodifiableSet(allBeans);
}
```

Action 2: Reduce garbage and memory usage in general

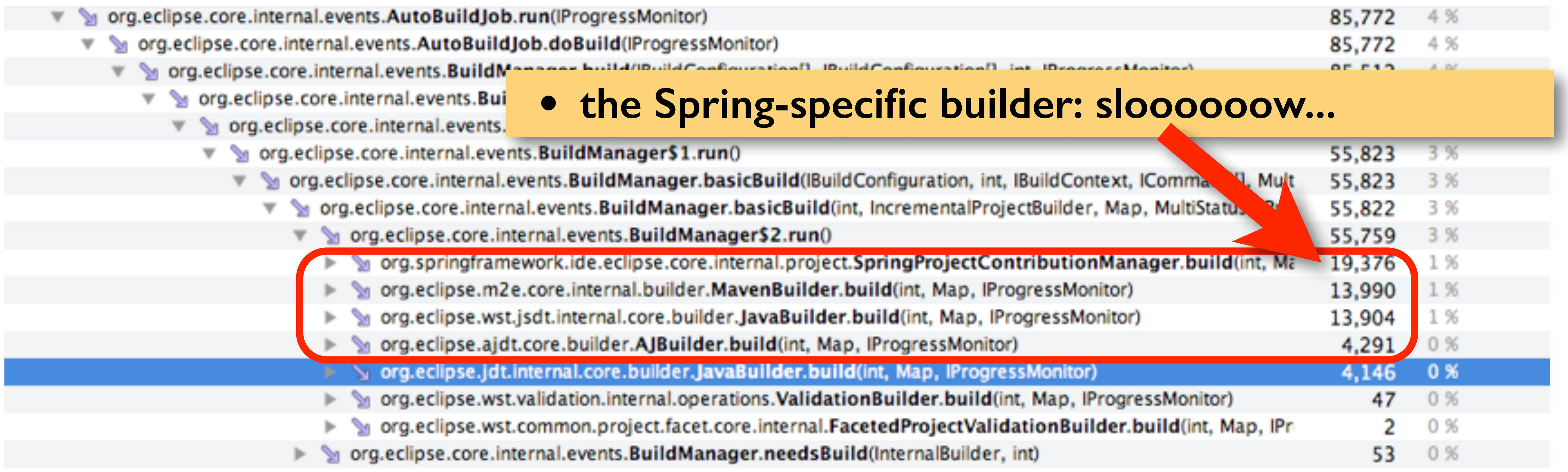
new set with copied content

```
public Set<IBean> getBeans() {
    Set<IBean> allBeans = new LinkedHashSet<IBean>(beans.values());
    for (IBeansImport beansImport : imports) {
        for (IBeansConfig bc : beansImport.getImportedBeansConfigs()) {
            allBeans.addAll(bc.getBeans());
        }
    }
    return Collections.unmodifiableSet(allBeans);
}
```

recursive call

- imagine this is called with deep recursion
- but since the method looks quite innocent, it is called many times while doing the build

Now back to the details of this…



- the Spring-specific builder: slooooooow...

```
▼ org.eclipse.core.internal.events.AutoBuildJob.run(IProgressMonitor)                                          85,772   4 %
  ▼ org.eclipse.core.internal.events.AutoBuildJob.doBuild(IProgressMonitor)                                    85,772   4 %
    ▼ org.eclipse.core.internal.events.BuildM
      ▼ org.eclipse.core.internal.events.Bui
        ▼ org.eclipse.core.internal.events.
          ▼ org.eclipse.core.internal.events.BuildManager$1.run()                                              55,823   3 %
            ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(IBuildConfiguration, int, IBuildContext, IComma   , Mult   55,823   3 %
              ▼ org.eclipse.core.internal.events.BuildManager.basicBuild(int, IncrementalProjectBuilder, Map, MultiStatu     55,822   3 %
                ▼ org.eclipse.core.internal.events.BuildManager$2.run()                                        55,759   3 %
                  ▶ org.springframework.ide.eclipse.core.internal.project.SpringProjectContributionManager.build(int, Ma   19,376   1 %
                  ▶ org.eclipse.m2e.core.internal.builder.MavenBuilder.build(int, Map, IProgressMonitor)         13,990   1 %
                  ▶ org.eclipse.wst.jsdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)     13,904   1 %
                  ▶ org.eclipse.ajdt.core.builder.AJBuilder.build(int, Map, IProgressMonitor)                    4,291    0 %
                  ▶ org.eclipse.jdt.internal.core.builder.JavaBuilder.build(int, Map, IProgressMonitor)          4,146    0 %
                  ▶ org.eclipse.wst.validation.internal.operations.ValidationBuilder.build(int, Map, IProgressMonitor)   47   0 %
                  ▶ org.eclipse.wst.common.project.facet.core.internal.FacetedProjectValidationBuilder.build(int, Map, IPr   2   0 %
                ▶ org.eclipse.core.internal.events.BuildManager.needsBuild(InternalBuilder, int)                   53   0 %
```

O(n)

matters for scalability

watch out for visitors

```
class ResourceDeltaVisitor implements IResourceDeltaVisitor {

    public boolean visit(IResourceDelta aDelta) throws CoreException {
        IResource resource = aDelta.getResource();
        if (resource instanceof IFile) {
            checkResource(resource);
        }
        return true;
    }
}
```

watch out for visitors

```
class ResourceDeltaVisitor implements IResourceDeltaVisitor {

    public boolean visit(IResourceDelta aDelta) throws CoreException {
        IResource resource = aDelta.getResource();
        if (resource instanceof IFile) {
            checkResource(resource);
        }
        return true;
    }
}
```

- this might be called many many times
- take care to make this simple and fast
- not allowed to iterate over collections

watch out for visitors

```
class ResourceDeltaVisitor implements IResourceDeltaVisitor {

    public boolean visit(IResourceDelta aDelta) throws CoreException {
        IResource resource = aDelta.getResource();
        if (resource instanceof IFile) {
            checkResource(resource);
        }
        return true;
    }
}
```

- this might be called many many times
- take care to make this simple and fast
- not allowed to iterate over collections

- in our case:
  - takes a look at individual IResource objects
  - identify the defined types
  - iterate over all defined beans and check for type dependency

the case: type checks

```
Set<IType> typesToCheck = new HashSet<IType>();

IType[] types = cu.getAllTypes();
for (IType type : types) {
    IType[] subTypes = type.newTypeHierarchy(monitor).getAllSubtypes(type);
    if (subTypes != null && subTypes.length > 0) {
        typesToCheck.addAll(Arrays.asList(subTypes));
    }
}
```

loop over beans and check each bean type whether it is contained in typesToCheck

the case: type checks

```
Set<IType> typesToCheck = new HashSet<IType>();

IType[] types = cu.getAllTypes();
for (IType type : types) {
    IType[] subTypes = type.newTypeHierarchy(monitor).getAllSubtypes(type);
    if (subTypes != null && subTypes.length > 0) {
        typesToCheck.addAll(Arrays.asList(subTypes));
    }
}
```

loop over beans and check each bean type whether it is contained in typesToCheck

- asking a type for its hierarchy is slow
- cached, but only for a limited number of hierarchies
- doing this for all resources of a build can take a very long time

instead: we built our own type hierarchy engine

TypeHierarchyEngine

it reads bytecode (only type information)

it walks up the super classes and interfaces

it caches already loaded type information

instead:
we built our own type hierarchy engine

TypeHierarchyEngine
it reads bytecode (only type information)
it walks up the super classes and interfaces
it caches already loaded type information

Lessons Learned
reading bytecode is super super fast
finding the bytecode on the classpath is super slow

What is designed to be fast?

Reconciling

Be extremely careful when implementing a reconcile participant

Content-Assist

Has to be fast

Don‘t do anything if its not your job

...

Startup time is important

(even if you start Eclipse just once a day)

Don‘t start

all your bundles and do stuff at startup

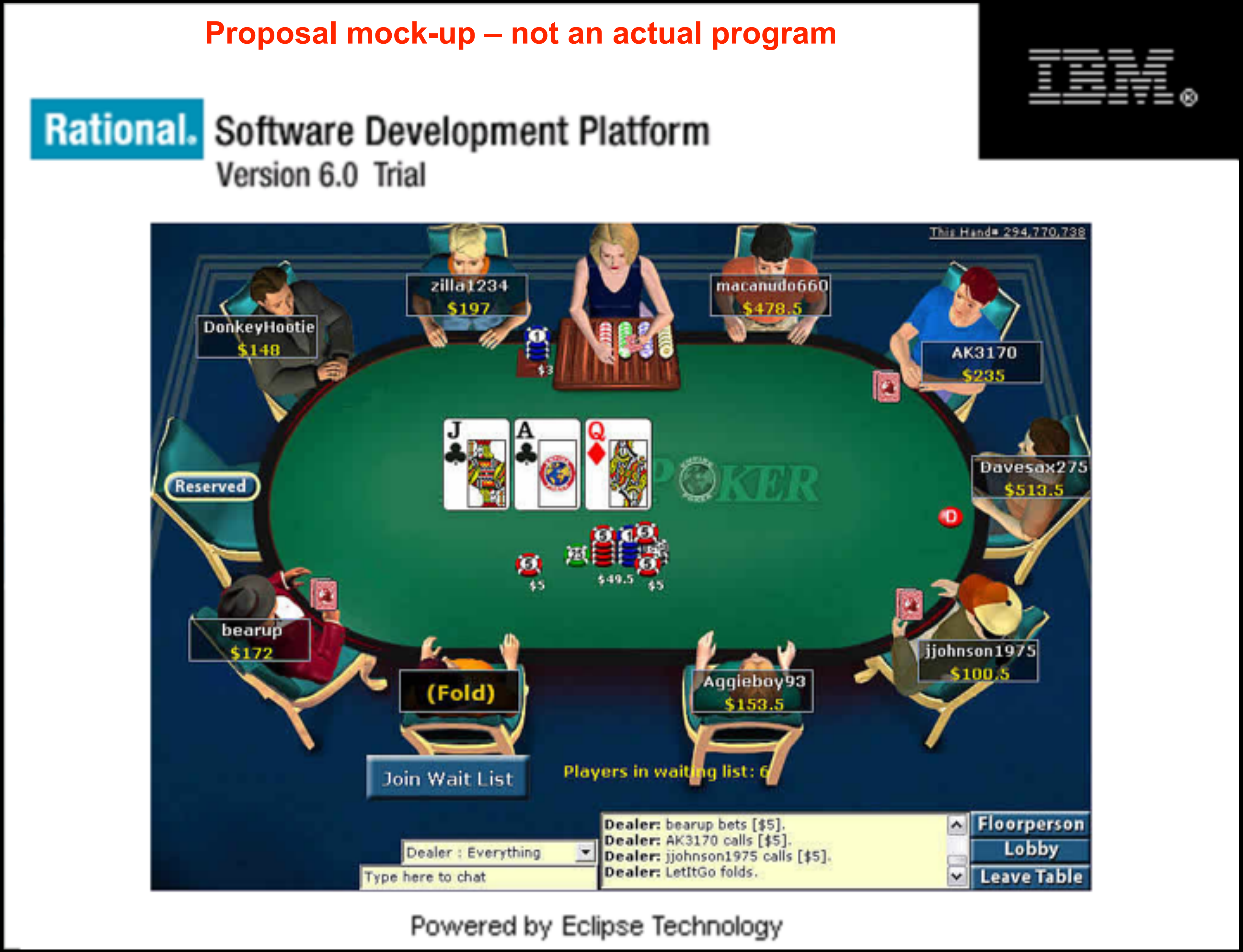
Do caching

(Equinox Weaving, for example)

Uninstall bundles

to get rid of things you don‘t need

A different approach



Proposal mock-up – not an actual program

Rational. Software Development Platform
Version 6.0 Trial

Dealer: bearup bets [$5].
Dealer: AK3170 calls [$5].
Dealer: jjohnson1975 calls [$5].
Dealer: LetItGo folds.

Powered by Eclipse Technology

from Chris Laffras talk on Eclipse performance

1. Measure
2. Optimize

3. Goto 1.

Q&A

and thank you for your attention

Martin Lippert
Principal Software Engineer - Pivotal
mlippert@gopivotal.com
@martinlippert