What is a component

Asked 16 years, 2 months ago Modified 10 months ago

Viewed 16k times

13

I listen to the podcast java posse, on this there is often discussion about components (note components are not (clearly) objects). They lament the fact that Java does not have components, and contrast with .NET that does. Components apparently makes developing applications (not just GUI apps) easier.

I can figure from the discussion certain qualities that a component has, its something to-do with decoupling (substituting one component for another is just a matter of plumbing). It has something to-do with properties, it definitely has something to-do with events and delegates.

So to the questions:

./ can anyone explain to me what a component is. (and why java beans are not components).

./ can anyone explain how they help development.

./ can anyone explain why java does not have them if they are so useful.

java .net object

Share

Improve this question

Follow

edited Oct 12, 2008 at 21:12

asked Oct 12, 2008 at 21:06

Scott James
684 ● 1 ● 5 ● 12

6 Answers

Sorted by: Highest score (default)

4

The term component is one of the most ambiguous and overused ones in OO.

Most people would agree that a component is made up of a group of classes, which collaborate together to implement one or more interfaces. One of the classes takes on the role of the 'front-end' i.e. it implements the interface but delegates the work to the other classes within the group. As you say components should be replaceable without the rest of the system knowing.

A great example of a component based architecture was COM. Its a great example because it was so heavily used and rigidly specified. But note that the need for this architecture was based on the inflexibility of the C++ compilation and deployment model.

in Java you can do an awful lot to a class without breaking binary compatability with the rest of the system. So there is not as much need to build rigid, component based architectures. But it all depends on how you define the term, e.g any project built using dependency injection could count as 'component based'.

Share Improve this answer Follow

edited Oct 12, 2008 at 21:29

answered Oct 12, 2008 at 21:23

Garth Gilmour
11.2k ● 5 ● 28 ● 36

3

Software Engineering Radio has an episode on exactly this topic: https://se-radio.net/2008/02/episode-87-software-components/

The general idea is that a software component can describe what its own dependencies and services are, in the form of metadata. I don't know why you might have heard that Java does not have components, since you can certainly imagine an architecture in Java where components do describe themselves through metadata. I suppose it's just that the Java platform definition by itself doesn't really have a component architecture.

Update: Indeed, one need not imagine such an architecture because as others pointed out, Java Beans or Servlets could certainly be considered component-based architectures.

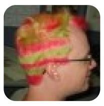
Share Improve this answer Follow

edited Jan 28 at 23:05

LennoxIsReallyBadAtCoding

3 ● 3

answered Oct 12, 2008 at 21:19



Greg Hewgill
990k ● 191 ● 1.2k ● 1.3k

Thanks for pointing out the link, I will certainly take a look. As to why I dont think components dont exist .. well at this point I am not clear what they are so I make no assertions either way. Ask Joe Nuxoll he is the chap that seems keen to say they dont (yet) exist. – Scott James Oct 12, 2008 at 21:25

PS Joe Nuxoll is one of the presenters on java posse (javaposse.com) totally recomend this podcast .. to anyone interested in java development. – Scott James Oct 12, 2008 at 21:26

When I listened this podcast, I was confused when he said something like: “There is a component marketplace where people create components and then they put them in this global repository where people can then grab… and assembly their systems from … I don’t believe this idea can be materialized, I don’t see that happening.” So, what about repositories like Maven Central Repository, or NPM for Node JS? – Miguel Gamboa Mar 17, 2014 at 21:10

2

Although in Java beginnings the notion of component was many times related with Gui components, the generic sense of component in software engineering goes beyond that notion.

Simply put, a **component is a piece of software, which is reusable**. Like bricks, we combine and join them to build a whole application. The key insight of software components in modern environments is the **Metadata**, which **describes the component's content and enables reuse**.

In 1996, the JDK 1.0 was the first managed runtime environment providing components with metadata. In this case, components are `.class` files containing *bytecodes* and metadata. Yet, and according to the Java Specification, **a `.class` file only contains one type definition**. So, to deploy a bundle of types as a component we may use a Jar archive containing several `.class` files.

On the other hand, in the .Net platform, which provides the same idea of reusable components, a component may contain more than one type definition. In this case a component (aka *assembly* in .Net) is a `.dll` or a `.exe` file.

Share Improve this answer Follow

edited Mar 17, 2014 at 23:10

answered Mar 17, 2014 at 21:22

Miguel Gamboa

9,323 ● 7 ● 51 ● 103

1

It depends on what you mean by "component". The term can mean lots of different things in a lot of different contexts, so it can easily get confusing. With that said, here's my understanding of the subject:

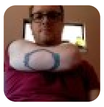
A component is different from an object (although objects are often used to represent and build components). The difference being a couple of things:

1. Objects tend to just be "things" whereas components are actors. The difference being that a component is a part of a process, while an object represents some kind of abstract idea.
2. Components help ensure code re-use and pluggability because they are basically small "sub-programs" (or sometimes programs in their own right) that ideally can be adapted to work well with other components.
3. You tend to see components more in "nothing shared" message-passing systems like Erlang or Kamaelia, mainly because those types of frameworks tend to be best suited for component-oriented design.

There are a lot of good examples of component-oriented design, but my first choice would be UNIX. The basic idea behind UNIX is that it is more a set of small programs designed to work together rather than being made up of several more monolithic programs.

Share Improve this answer Follow

answered Oct 12, 2008 at 21:47



Jason Baker
198k ● 138 ● 382 ● 520

1

Software comes in several groupings. Here are the Java chunks.

- Statements.
- Method functions. Multiple statements.
- Class. Multiple attributes and method functions.
- File. One or more classes. One class is the public class in the file, other classes are hidden in the file.
- Package. Multiple classes. These form a hierarchy.

"Component", "Layer", "Tier" and other philosophical groupings are -- generally -- notional. The VB COM environment had a formalism for components. Everyone else treats them as just ideas.

Beans are classes. Can be single class be a component? Maybe. A component is usually a bunch of classes. Sometimes as few as two - a formal interface and an implementation.

Components help you focus on a logical grouping of classes, packages, groupings, etc.

Since a component is notional, every language more-or-less has them. There are few language formalisms for components. They aren't really needed. It's an idea or a principle you use to structure your thinking.

You can, with some care, define an approach to "components" with an interface and metadata and numerous other features.

Share Improve this answer Follow

answered Oct 12, 2008 at 22:02

S.Lott
391k ● 82 ● 517 ● 788

0

I don't know .NET components particularly, but from the Java POV, I'd say that a component is some functional unit that should have a defined interface/usage principle. While Java does not have components as a language concept, there are IMHO components in Java. Technical components would be e.g.:

- EJBs
- Servlets

Functional components would be e.g.:

- Automatic Update for an application
- Formula mechanism that enables calculations on a data model

Architectural components could be JAR files or OSGi bundles.

Of course, there is always room for interpretation ;)

Share Improve this answer Follow

answered Oct 12, 2008 at 21:25

Martin Klinke
7,322 ● 6 ● 43 ● 64