Eclipse RCP Part XVI – e4 dependency injection

Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities Automotive Financial Services Insurance Life Science & Healthcare Public Sector

.consulting .solutions .partnership

msg systems

Dependency Injection

```
public class MyPart {

  @Inject private Logger logger;
  // DatabaseAccessClass would talk to the DB
  @Inject private DatabaseAccessClass dao;

  @Inject
  public void init(Composite parent) {
    logger.info("UI will start to build");
    Label label = new Label(parent, SWT.NONE);
    label.setText("Eclipse 4");
    Text text = new Text(parent, SWT.NONE);
    text.setText(dao.getNumber());
  }

}
```

- „don‘t call us, we‘ll call you!“
- Dependency injection can happen on
  - The constructor of the class (construction injection)
  - a method (method injection)
  - a field (field injection)

Dependency injection in Eclipse 4

- Eclipse 4 runtime creates objects referred by the application model
- During this instantiation the classes are scanned for injection annotations
- Based on these, the injection is performed
- Eclipse framework also tracks the injected values and re-injects if the change

Dependency injection annotations in Eclipse 4

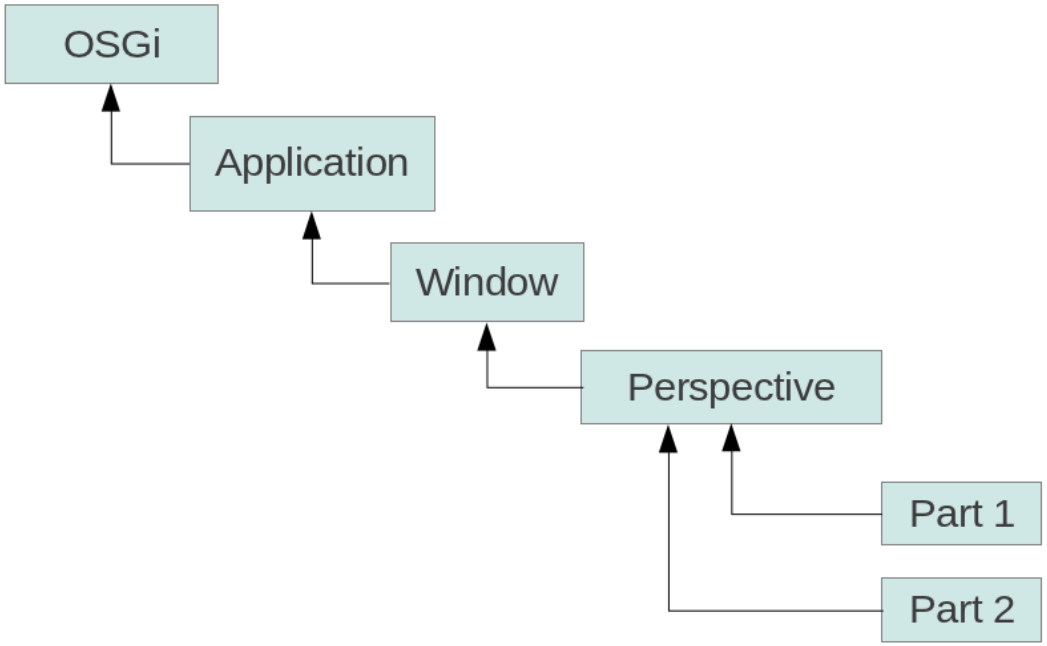
Annotation	Description
@javax.inject.Inject	Marks a field, a constructor or a method. The Eclipse framework tries to inject the parameter.
@javax.inject.Named	Defines the name of the key for the value which should be injected. By default the fully qualified class name is used as key. Several default values are defined as constants in the `IServiceConstants` interface.
@Optional	Marks an injected value to be optional. If it can not be resolved, `null` is injected. Without @Optional the framework would throw an Exception. The specific behavior depends where @Optional is used: • for parameters: a `null` value will be injected; • for methods: the method calls will be skipped • for fields: the values will not be injected.
@Preference	Eclipse can store key/values pairs as so-called preferences. The @Preference annotation defines that the annotated value should be filled from the preference store.

What can be injected?

- Eclipse runtime creates a context in which the possible values for injection are stored.
- The context can be modified
- The context contains
  - All objects associated with the application model
  - All other objects which have explicitly been added to the context
  - All Preferences
  - OSGi services
- Objects can be placed in the context via
  - Class name
  - String key -> context variable
- Several context variable keys are defined in IServiceConstants

How are objects searched?

- Context is hierarchical
- Model elements which implement Mcontext have a local context
- If an object is not found in the context, the parent context is searched



Behavioral annotations

Table 7. Eclipse lifecycle annotations for Parts

Annotation	Description
@PostConstruct	Is called after the class is constructed and the field and method injection has been performed.
@PreDestroy	Is called before the class is destroyed. Can be used to clean up resources.
@Focus	Indicates that this method should be called, once the Part gets the focus. It is ***required*** to set the focus on one user interface control otherwise certain workbench functionality does not work.
@Persist	Is called if a save request on the Part is triggered. Can be used to save the data of the Part.
@PersistState	Is called before the model object is disposed, so that the Part can save its state.

Table 8. Other Eclipse Behavior Annotations

Annotation	Description
@Execute	Marks a method in a handler class to be executed
@CanExecute	Marks a method to be visited by the Command- Framework to check if a handler is enabled
@GroupUpdates	Indicates that updates for this @Inject should be batched. If you change such objects in the `IEclipseContext` the update will be triggered by the `processWaiting()` method on `IEclipseContext`.
@EventTopic and @UIEventTopic	Allows you to subscribe to events send by the EventAdmin service.

Example view

```
package com.example.e4.rcp.todo.ui.parts;

import javax.inject.Inject;

import org.eclipse.swt.widgets.Composite;

public class TodoOverviewPart {

  @Inject
  public TodoOverviewPart(Composite parent) {

  // Assuming that dependency injection works
  // parent will never be null
  System.out.println("Woh! Got Composite via DI.");

  // Does it have a layou manager?
  System.out.println("Layout: " + parent.getLayout().getClass());
  }
}
```

```
@Focus
private void setFocus() {
  label.setFocus();
}
```

Example handler

```java
package com.example.e4.rcp.todo.handlers;

import org.eclipse.e4.core.di.annotations.CanExecute;
import org.eclipse.e4.core.di.annotations.Execute;

public class SaveAllHandler {
  @Execute
  public void execute() {
    System.out.println("Called");
  }

  // Technically not needed
  // will default to true
  @CanExecute
  public boolean canExecute() {
    return true;
  }

}
```

Lab

- Add some controls to one of the views from last exercise
- Add a menu with some command that prints „hello e4“ to the console

Vielen Dank für Ihre Aufmerksamkeit

.consulting .solutions .partnership

.msg systems