Modernizing Eclipse RCP/e4 applications and tools

Jonas Helming
EclipseSource
Evolution
Agenda

- Eclipse 4?
- UI Technologies
- Eclipse Che and Theia?
- Basic architecture pattern
- “Good and bad technologies”
Eclipse 4?

- Do we need to migrate to Eclipse 4
- When will the compat layer be deprecated?
- Are extension points deprecated?
Eclipse 3.x vs. Eclipse 4.x vs. e4?

3.x Application

- Eclipse runtime
- OSGi

- Workbench (org.eclipse.ui)
- JFace
- SWT

Eclipse runtime

OSGi

JFace

SWT
Eclipse 3.x vs. Eclipse 4 vs. e4?

- **Eclipse runtime**
  - OSGi
  - E4: Application Model, Services, Dependency Injection
  - SWT
  - JFace

- "pure" Eclipse 4 Application
Eclipse 3.x vs. Eclipse 4 vs. e4?

- **Eclipse 4.x Application (Compatibility Layer)**
  - Workbench Compatibility Layer (org.eclipse.ui.e4)
  - e4: Application Model, Services, Dependency Injection

- **Eclipse runtime**
  - OSGi

- **JFace**
  - SWT
Advantages of “pure” Eclipse 4

- Application Model
- Cleaner architecture (i.e. dependency injection and POJOs)
- More flexibility, e.g.:
  - Better styling of Tabs
  - JavaFX

- Disadvantage: No use of 3.x plugins
Eclipse 3.x vs. compat layer vs. “pure” Eclipse 4?

- Don’t go for Eclipse 3.x anymore, choose between:
  - “pure” Eclipse 4
  - Compatibility layer (3.x API based on Eclipse 4.x)
- Existing IDE related frameworks usually use the 3.x API
- Look at the frameworks you are going to use:
  - Many 3.x based => Go for the compatibility layer
  - A few 3.x based => evaluate their adaptation to Eclipse 4
  - No 3.x based => go for “pure” Eclipse 4
- If you develop for the 3.x API, do it in an Eclipse 4 friendly way (see references)
UI Technologies

- SWT provides powerful support for build tools, BUT:
  - Maintenance is decreasing
  - It still lacks styling capabilities (and will probably always do)
- Valid alternatives are:
  - JavaFX
  - HTML5
Migrating to JavaFX

- Advantages:
  - Also Java
  - Perfect styling
  - Well supported in native Eclipse 4
  - Embeddable into SWT for a “soft migration”

- Disadvantages:
  - No path to the browser
  - Difficult to embed web-based UI parts
Migrating to HTML5

● Advantages:
  ○ Perfect styling
  ○ Prepares a migration to the browser
  ○ Future-proof
  ○ Embeddable into SWT for a “soft migration” (but less well compared to JavaFX)

● Disadvantages:
  ○ Technology mix
  ○ Refactoring likely necessary
“Retrofitting” JavaFX or HTML for a soft migration

import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.transform.Rotate;
import javafx.util.Duration;

public class DetailsViewFX {
    @Override
    public void createPartControl(Composite parent) {
        Label label = new Label();
        label.setText("Name:");
        GridPane grid = new GridPane();

        comp = new Composite(parent, SWT.NO_SCROLL);
        comp.setLayout(new GridLayout(2, true));

        browser = new Browser(comp, SWT.NO_SCROLL);
        browser.setLayoutData(new GridData(GridData.FILL, GridData.FILL, true, true));
        browser.clearSessions();
        browser.refresh();
    }
}
Migration to Eclipse Theia or Che?

- Eclipse Theia is a platform for building web-based tools
- Eclipse Che is a “workspace server” to host developer environments in the cloud
- Additional technologies are emerging, such as GLSP, JSON Forms, LSP/XText, Sprotty and EMF.cloud
- This is **NOT** like “Eclipse 5” migrating means:
  - Reimplement all UI components
  - Refactor business logic to be reused
- Why should you ever migrate?
  - Modern UI and styling
  - Zero installation for users
  - Enables “cloud” business models

→ **Important now:** Evaluate benefits, define strategy and timeplan, build POCs and incrementally adapt existing architecture
How to structure bundles / “MVVM-like”

Use OSGi Declarative Services to register this.
Thin UI layer
Good and bad technologies

- Buckminster
- PDE Builder
- 3.x Eclipse version as IDE or target

- Tycho
- BND Tools
- Oomph
- EMF
- Declarative OSGi services (with DS annotations)
Conclusion

- Get rid of any Eclipse 3.x targets or IDEs
- It is valid to remain on the compatibility layer
- Consider JavaFX or HTML for selected UI components
- Keep your UI layer as thin as possible (better maintenance and reliability)
- Develop a strategy “If, when and how” to migrate to the web/cloud

Questions?

Get in contact with us for support around Eclipse Theia and Che!
Here at our booth
jhelming@eclipsesource.com
References

- Eclipse Oomph https://eclipsesource.com/blogs/tutorials/oomph-basic-tutorial/
- Soft migration to Eclipse 4 https://eclipsesource.com/blogs/tutorials/eclipse-4-e4-tutorial-soft-migration-from-3-x-to-eclipse-4-e4/
- Migrating to Eclipse 4 https://eclipsesource.com/blogs/2016/01/08/migrating-from-eclipse-3-x-to-eclipse-4-e4/
EVALUATE THE SESSIONS
Sign in and vote using the conference app or eclipsecon.org

WITH +1 this