Agenda Eclipse IDE Plug-in Development

- Introduction to Plug-in development
  - Components of the Eclipse platform
  - Important configuration files

- Plug-in development and deployment
  - Creating plug-ins
  - Exporting plug-ins
  - Using features
  - Creating update sites

- Using the application model
  - The concept of the dynamic application model
  - Accessing, updating and creating model elements
  - Using model fragments and processors

- Using the Eclipse dependency injection framework
  - Introduction into dependency injection
  - Using DI in Eclipse e4 components
  - Using DI in Eclipse 3.x components
  - Accessing the Eclipse context via singletons

- Extending the IDE with views
  - Contributing 3.x views
  - Contributing e4 view

- User interface programming with SWT and JFace
  - Using SWT
  - Using JFace
  - Using Databinding
  - Implementing wizards and dialogs

- Declarative styling with CSS
• Introduction into CSS
• Definition of styles and themes, colors and gradients
• Styling specific widgets
• Dynamic style switching at runtime
• Using the CSS Spy tooling

• Using commands
  • Contributing commands and menus
  • Using toolbars, view menus and popup menus

• Platform services and interaction of components
  • Service overview
  • Part service
  • Model service
  • Selection service
  • Command and Handler service

• Modularity for Eclipse 4 applications
  • Contributing to the application model
  • Static model contributions with fragments
  • Dynamic model contributions with processors

• Internationalization (i18n)
  • Adding support for multiple languages
  • Usage of fragment projects
  • Outlook: translation services in Eclipse 4

• Concurrent UIs
  • SWT threading
  • Avoiding invalid thread access
  • Asynchronous processing with the Eclipse API

• Resources and markers
  • Integration with the problems view
  • Custom markers
• **Adapter pattern**
  • Implementing the adapter pattern
  • Adaptable objects
  • Using the Properties and Outline view

• **Implementing custom text editors**
  • Implementing a text editor
  • Reusing the generic editor framework
  • Providing syntax highlighting, hover, code completion and validation

• **Extending the Java development tools**
  • Abstract Syntax Tree vs Java Model
  • Custom generators
  • Custom quick fixes
  • Custom templates
  • Using the refactoring API