Eclipse 4 RCP Delta training agenda

- Introduction into Eclipse and Eclipse 4
  - Components of the Eclipse platform
  - Eclipse 3.x in comparison with Eclipse 4.x
  - Eclipse license
  - Internet information sources

- Eclipse architecture
  - Software components
  - Configuration files (plugin.xml, MANIFEST.MF)
  - Extensions and extension points
  - Important user interface components

- Eclipse 4 application model
  - Application model and model components
  - Model editor
  - Naming schema for ID's

- Dependency injection and annotations
  - Overview dependency injection
  - Dependency injection framework in Eclipse
  - Field, method and constructor dependency injection
  - Behavior annotations
  - Application lifecycle annotations

- Scope of injection
  - IEclipseContext
  - Injection search strategy
  - Creation of injectable objects
  - Model elements and dependency injection

- OSGi services
  - Services and the OSGi service registry
  - Publishing services via OSGi declarative services
  - Usage of services in Eclipse 4
  - OSGi declarative service definition with annotations

- Commands, Handlers, Menus and Toolbars
  - Contributing to the menu and the toolbar
  - Handling of popup menus
  - Scope of handlers and core expressions
  - Defining keybindings

vogella GmbH, Haindaalwisch 17a, 22395 Hamburg, T +49 40 78 80 4360, F +49 40 52476322, Email info@vogella.com
HypoVereinsbank Hamburg, IBAN: DE82200300000015620690 , BIC/Swift-Code: HYVEDEMM300
Steuernummer: 50/766/01072; USt-IdentNr.: DE-284122352, Amtsgericht Hamburg: HRB 127058, Sitz der Gesellschaft: Hamburg
Geschäftsführer: Lars Vogel, Jennifer Nerlich de Vogel

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

- Editor handling in Eclipse 4
  - Comparison Views and Editors
  - Getting parts which behave as editors
  - Using services to interact with parts

- Accessing and extending the Eclipse context
  - Accessing the context
  - Extending the Eclipse context with own objects
  - Using dependency injection to create own objects

- Settings and preferences
  - Configuration area and workspace
  - Persistence of the Eclipse application
  - Part persistence
  - Dependency injection for preference values

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

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

- Migrating Eclipse 3.x applications
  - Running Eclipse 3.x applications on top of Eclipse 4
  - Mixing Eclipse 3.x and Eclipse 4.x components
  - Discussion: Migration path for existing applications

- Definition of own annotations for dependency injection
  - Definition of new annotations
  - Evaluation of new annotations
  - Use cases

- 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

- The Renderer framework
  - Purpose of the Renderer framework
  - Define your own renderer
  - Outlook: Using an alternative renderer
  - Outlook: Extending the application model

- **Best practices and tips & tricks**