Smart, slim and good looking

Tom Schindl  <tom.schindl@bestsolution.at>

Twitter: @tomsontom
Blog:  http://tomsondev.bestsolution.at
Website:  http://www.bestsolution.at
About Tom

- CTO BestSolution.at Systemhaus GmbH
- Eclipse Committer
  - e4
  - Platform
  - EMF
- Project lead
  - e(fx)clipse
- Twitter: @tomsontom
- Blog: tomsondev.bestsolution.at
- Corporate: http://bestsolution.at
JavaFX applications

- By default used for database frontends and visualization applications
JavaFX applications

• By default used for database frontends and visualization applications
JavaFX applications

- By default used for database frontends and visualization applications
The requirement

- Provide a lightweight text-editor framework who
  - Supports declarative definition for syntax highlighting like any sane editor since vi has (exception is Eclipse)
  - Must be themable
  - Should not require a workspace (aka core.resources)
  - Should run in/outside OSGi
  - Should optionally support Autocomplete, Error Markers, ...
The plan
The plan

- Make use of org.eclipse.text and provide a declarative way to configure syntax highlighting
The plan

- Make use of org.eclipse.text and provide a declarative way to configure syntax highlighting

- Everything is a service (Highlighting, Auto-complete, …)
The plan

- Make use of org.eclipse.text and provide a declarative way to configure syntax highlighting
- Everything is a service (Highlighting, Auto-complete, ...)
- Fork org.eclipse.jface.text and replace SWT API calls with JavaFX APIs
The plan

- Make use of org.eclipse.text and provide a declarative way to configure syntax highlighting
- Everything is a service (Highlighting, Auto-complete, …)
- Fork org.eclipse.jface.text and replace SWT API calls with JavaFX APIs
- „Houston we have a problem“ JavaFX has no StyledText-Control
The plan

- Make use of `org.eclipse.text` and provide a declarative way to configure syntax highlighting

- Everything is a service (Highlighting, Auto-complete, …)

- Fork `org.eclipse.jface.text` and replace SWT API calls with JavaFX APIs

- „Houston we have a problem“ JavaFX has no StyledText-Control

  - Ok let’s create one!
The result
The result
The result
The result
Basic Architecture

Editor-Control
(file:/mysample.ts)

DI
- content
- syntax highlighting
- auto-complete
- error-marker
Basic Architecture

Editor-Control (file:/mysample.ts)

content

DI

syntax highlighting

auto-complete

error-marker
Basic Architecture

Editor-Control
(file:/mysample.ts)

- content
- syntax highlighting

DI

- auto-complete
- error-marker
Basic Architecture

Editor-Control
(file:/mysample.ts)

content

syntax highlighting

auto-complete

DI

error-marker
Basic Architecture

Editor-Control
(file:/mysample.ts)

- content
- syntax highlighting
- auto-complete
- error-marker

DI
Why Eclipse Text??
Why Eclipse Text??

- Why using Eclipse Text
Why Eclipse Text??

‣ Why using Eclipse Text

‣ It’s fast
Why Eclipse Text??

- Why using Eclipse Text
  - it’s fast
  - it’s faster than fast
Why Eclipse Text??

- Why using Eclipse Text
  - it’s fast
  - it’s faster than fast

| StyleText (Eclipse Text) | RichText (regex) |
Why Eclipse Text??

- Why using Eclipse Text
  - it’s fast
  - it’s faster than fast

<table>
<thead>
<tr>
<th>StyleText (Eclipse Text)</th>
<th>RichText (regex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init - 10 000 Loc</td>
<td>327ms</td>
</tr>
</tbody>
</table>
Why Eclipse Text??

Why using Eclipse Text

- it’s fast
- it’s faster than fast

<table>
<thead>
<tr>
<th></th>
<th>StyleText (Eclipse Text)</th>
<th>RichText (regex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init - 10 000 Loc</td>
<td>327ms</td>
<td>400ms</td>
</tr>
<tr>
<td>init - 150 000 Loc</td>
<td>1100ms</td>
<td>3300ms</td>
</tr>
</tbody>
</table>
Why Eclipse Text??

‣ Why using Eclipse Text

‣ it’s fast

‣ it’s faster than fast

<table>
<thead>
<tr>
<th></th>
<th>StyleText (Eclipse Text)</th>
<th>RichText (regex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init - 10 000 Loc</td>
<td>327ms</td>
<td>400ms</td>
</tr>
<tr>
<td>init - 150 000 Loc</td>
<td>1100ms</td>
<td>3300ms</td>
</tr>
<tr>
<td>change - 10 000 Loc</td>
<td>30ms</td>
<td>110ms (*)</td>
</tr>
</tbody>
</table>
Why Eclipse Text??

Why using Eclipse Text

- it's fast
- it's faster than fast

<table>
<thead>
<tr>
<th></th>
<th>StyleText (Eclipse Text)</th>
<th>RichText (regex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init - 10 000 Loc</td>
<td>327ms</td>
<td>400ms</td>
</tr>
<tr>
<td>init - 150 000 Loc</td>
<td>1100ms</td>
<td>3300ms</td>
</tr>
<tr>
<td>change - 10 000 Loc</td>
<td>30ms</td>
<td>110ms (*)</td>
</tr>
<tr>
<td>change - 150 000 Loc</td>
<td>50ms</td>
<td>1800ms (*)</td>
</tr>
</tbody>
</table>
Why Eclipse Text??

- Why using Eclipse Text

- It’s fast

- It’s faster than fast

<table>
<thead>
<tr>
<th></th>
<th>StyleText (Eclipse Text)</th>
<th>RichText (regex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>init - 10 000 Loc</td>
<td>327ms</td>
<td>400ms</td>
</tr>
<tr>
<td>init - 150 000 Loc</td>
<td>1100ms</td>
<td>3300ms</td>
</tr>
<tr>
<td>change - 10 000 Loc</td>
<td>30ms</td>
<td>110ms (*)</td>
</tr>
<tr>
<td>change - 150 000 Loc</td>
<td>50ms</td>
<td>1800ms (*)</td>
</tr>
</tbody>
</table>

* Potential Bug: numbers might be devided by 2
Why is it so fast

- Eclipse Text is parsing text in 2 phases
  - First it partitions text
  - Afterwards it tokenizes text
- On change it only has to parse the modified partition instead of the complete file
Why is it so fast?

```java
/**
 * This is a sample class
 */
class Sample {
    //Some information

    public void test() {
        /*
         * Some more information
         */
        var s =
            "Hello World"

        ;
        print(s);
    }
}
```
Why is it so fast?

```dart
/**
 * This is a sample class
 */

class Sample {
  //Some information

  public void test() {
    /*
     * Some more information
     */
    var s =
      "Hello World"

    ;
    print(s);
  }
}
```
Why is it so fast?

```dart
/**
 * This is a sample class
 */
class Sample {
  //Some information

  ///Some more information
  public void test() {
    /*
     * Some more information
     */
    var s = "Hello World"
    ;
    print(s);
  }
}
```

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommerical-ShareAlike 3.0
Why is it so fast?

```dart
/**
 * This is a sample class
 */

class Sample {
  //Some information

  public void test() {
    /*
     * Some more information
     */
    var s =
      "Hello World"
      ;
    print(s);
  }
}
```
Why is it so fast?

```dart
/**
 * This is a sample class
 */

class Sample {
  //Some information

  public void test() {
    /*
     * Some more information
     */
    var s = "Hello World"
    ;
    print(s);
  }
}
```

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommerical-ShareAlike 3.0
Why is it so fast?

```dart
/**
 * This is a sample class
 */

class Sample {
  //Some information
  public void test() {
    /*
     * Some more information
     */
    var s = "Hello World"
    ;
    print(s);
  }
}
```

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommercial-ShareAlike 3.0
Partitioning Rules
## Partitioning Rules

<table>
<thead>
<tr>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
</table>

## Partitioning Rules

<table>
<thead>
<tr>
<th></th>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>dartdoc</td>
<td>/**</td>
<td>*/</td>
<td>✔</td>
</tr>
</tbody>
</table>
# Partitioning Rules

<table>
<thead>
<tr>
<th></th>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>dartdoc</td>
<td>/**</td>
<td>*/</td>
<td>✔</td>
</tr>
<tr>
<td>comment single</td>
<td>//</td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
## Partitioning Rules

<table>
<thead>
<tr>
<th></th>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>dartdoc</td>
<td>/**</td>
<td>*/</td>
<td>✓</td>
</tr>
<tr>
<td>comment single</td>
<td>//</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>comment multi</td>
<td>/*</td>
<td>*/</td>
<td>✓</td>
</tr>
</tbody>
</table>
# Partitioning Rules

<table>
<thead>
<tr>
<th></th>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dartdoc</strong></td>
<td>/**</td>
<td>*/</td>
<td>✓</td>
</tr>
<tr>
<td><strong>comment single</strong></td>
<td>//</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><strong>comment multi</strong></td>
<td>/*</td>
<td>*/</td>
<td>✓</td>
</tr>
<tr>
<td><strong>string</strong></td>
<td>&quot;</td>
<td>&quot;</td>
<td>✗</td>
</tr>
</tbody>
</table>
# Partitioning Rules

<table>
<thead>
<tr>
<th>dartdoc</th>
<th>Start</th>
<th>End</th>
<th>MultiLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment single</td>
<td>//</td>
<td></td>
<td></td>
</tr>
<tr>
<td>comment multi</td>
<td>/*</td>
<td>*/</td>
<td></td>
</tr>
<tr>
<td>string</td>
<td>&quot;</td>
<td>&quot;</td>
<td></td>
</tr>
<tr>
<td>character</td>
<td>'</td>
<td>'</td>
<td></td>
</tr>
</tbody>
</table>
Tokenizing a Partition

```java
/**
 * This is a sample class
 */
class Sample {

    public void test(String s) {
        print(s);
    }
}
```
Tokenizing a Partition

```dart
/**
 * This is a sample class
 */

class Sample {
  public void test(String s) {
    print(s);
  }
}
```
Tokenizing a Partition

```dart
/**
 * This is a sample class
 */
class Sample {
    public void test(String s) {
        print(s);
    }
}
```
Tokenizing a Partition

```dart
/**
 * This is a sample class
 */
class Sample {
    public void test(String s) {
        print(s);
    }
}
```
Tokenizing a Partition

```dart
/**
 * This is a sample class
 */

class Sample {
  public void test(String s) {
    print(s);
  }
}
```
Service 1 - Syntax Highlighting
Syntax highlighting
Syntax highlighting

- Define by DSL (with the help of Xtext)
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
  - Holds the partitioning and tokenizing information
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
  - Holds the partitioning and tokenizing information
  - Generated artifacts
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
  - Holds the partitioning and tokenizing information
  - Generated artifacts
    - CSS definitions
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
  - Holds the partitioning and tokenizing information
- Generated artifacts
  - CSS definitions
  - JSON-Configurations
Syntax highlighting

- Define by DSL (with the help of Xtext)
  - Designed to be as close to Eclipse-Text as possible
  - Holds the partitioning and tokenizing information
  - Generated artifacts
    - CSS definitions
    - JSON-Configurations

- Artifacts contributed through OSGi-services

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommercial-ShareAlike 3.0
Syntax highlighting

- Define by DSL (with the help of Xtext)
- Designed to be as close to code as possible
- Holds the partitioning
- Generated artifacts
- CSS definitions
- JSON-Configurations

Artifacts contributed through

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommerical-ShareAlike 3.0
Livecoding
There’s trend going on

- Language inventors ship headless services providing access to
  - Semantic information used to build e.g. Outline-Views
  - Autocompletion
  - Error-Reconciling
- Benefit for them ALL IDE can add support for the language more easily
Need examples
Need examples

- Dart-Ships the Dart-Analysis-Server
Need examples

- Dart-Ships the Dart-Analysis-Server
- Typescript ships ts-server and/or an embeddable JS-Component named LanguageService supporting Typescript and since 1.7 JavaScript
Need examples

- Dart-Ships the Dart-Analysis-Server

- Typescript ships ts-server and/or an embeddable JS-Component named LanguageService supporting Typescript and since 1.7 JavaScript

- For C# there’s omnisharp built on top of roslyn
Need examples

- Dart-Ships the Dart-Analysis-Server

- Typescript ships ts-server and/or an embeddable JS-Component named LanguageService supporting Typescript and since 1.7 JavaScript

- For C# there’s omnisharp built on top of roslyn

- go and rust have headless server apps to talk to
TS-Language-Service in Java

- Java-API provided by BestSolution at github
  - Uses j2v8 if available else falls back to nashorn
TS-Language-Service in Java

- Java-API provided by BestSolution at github

- Uses j2v8 if available else falls back to nashorn

```java
import at.bestsolution.typescript.service.api.TSServer;
import at.bestsolution.typescript.service.api.TSServerFactory;
import at.bestsolution.typescript.service.api.services.LanguageService;

TSServerFactory factory = Util.lookupService(TSServerFactory.class);
TSServer server = factory.getServer(UUID.randomUUID().toString(), s -> {
    // ... project init
});
LanguageService s = server.getService(LanguageService.class);
s.getCompletionsAtPosition("myfile.ts", 100);
```
efxcode - Autocomplete
efxcode - Autocomplete

- Autocomplete is provided by a 2 services
efxcode - Autocomplete

- Autocomplete is provided by a 2 services
  - Core service to get the completion entries
efxcode - Autocomplete

- Autocomplete is provided by 2 services

- Core service to get the completion entries

```
ProposalContext
- document : IDocument
- input : Input<?>
- location : int
- toString () : String

ProposalComputer
- compute (context : ProposalContext) : CompletableFuture<List<CompletionProposal>>

ProposalComputerTypeProvider
```
efxcode - Autocomplete

- Autocomplete is provided by 2 services
  - Core service to get the completion entries
  - UI service to present them
efxcode - Autocomplete

- Autocomplete is provided by 2 services
  - Core service to get the completion entries
  - UI service to present them
Livecoding
Error markers have use two services

- Core-Service to build an Annotation-Model
- UI-Bound-Service to visualize the Errors as
  - Markers next to the Line-Numbers
  - Overlay in the content area underlining the error
efxcode - Error-Markers

- Error markers have use two services
  - Core-Service to build an Annotation-Model
  - UI-Bound-Service
    - Markers next to
    - Overlay in the
efxcode - Error-Markers

- Error markers have use two services
  - Core-Service to build an Annotation-Model
  - UI-Bound-Service

(c) BestSolution.at - Licensed under Creative Commons Attribution-NonCommerical-ShareAlike 3.0
Livecoding
Resources

- e(fx)clipse - http://efxclipse.org/

- Code-Editor Sources - https://github.com/BestSolution-at/eclipsecon2016code