

Informatik II

Bachelor of Arts: TSIT

Teil 6 : Internationalisierung / Lokalisierung / Qualitätskontrolle

How many translators
does it take to
change a lightbulb?

....hmm,
depends on the context!

- Weltweite Märkte und Vernetzung
- Vertrieb bzw. Nutzung von Programmen weltweit.
 - ◆ Ein Programm für jede Sprache schreiben?

 - ◆ NEIN!

 - ◆ Antwort: Lokalisierung von zuvor Internationalisierten Programmen

Was ist Internationalisierung?

- Software wird so konzeptioniert dass Sie
 - ◆ leicht
 - ◆ An Kulturen
 - ◆ Sprachenangepasst werden kann

- Kurz: i18n

I18n = INTERNATIONALIZATION



Was ist Lokalisierung?

- Anpassung der Inhalte einer Software an
 - ◆ Region
 - ◆ Kulturen
 - ◆ Sprachen

- Kurz: l10n

Betroffene Inhalte?

- Farben, Grafiken, Farbpaletten
- Zeichensätze, Schreibrichtungen
- Texte
- Video, Audio
- Konventionen in Bezug auf Datum, Dezimalzahlen, Währungen, Telefonnummern

??? Fragen



**Welche
Fragen
haben Sie?**

JETZT



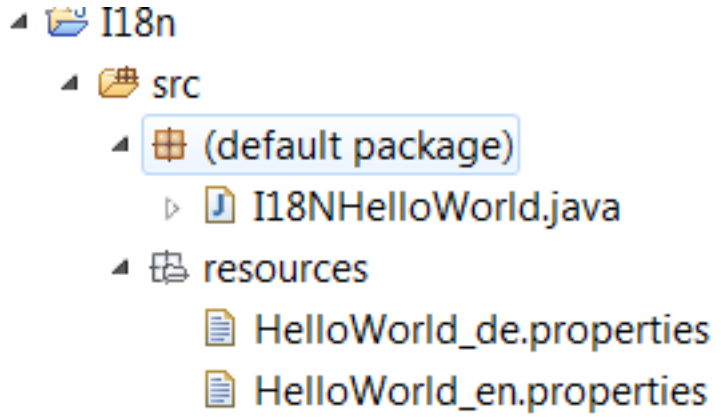
Internationalisierung und Lokalisierung mit Java


```
public class Hello {
```

```
    public static void main(String[ ] args) {  
        System.out.println("Hallo Mannheim!");  
    }
```

```
}
```

Internationalisierung



```
# HelloWorld_de.properties  
HelloWorld=Hallo Welt.  
Bye=Tschüss.
```

```
# HelloWorld_en.properties  
HelloWorld=Hello World.  
Bye=Bye.
```

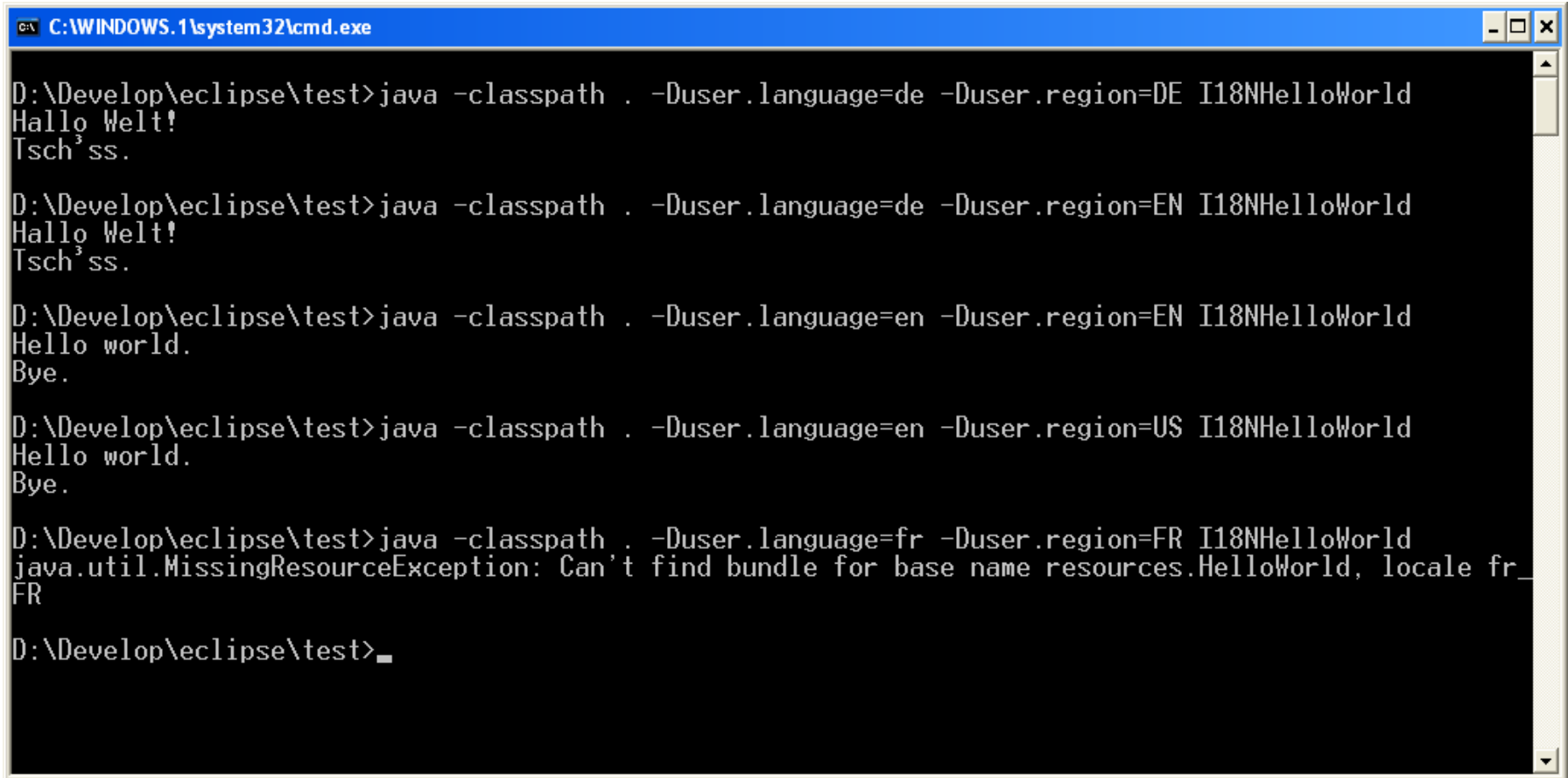
<name>_<language>_<country>.properties



```
import java.util.*;

public class I18NHelloWorld {
    public static void main( String[] args ) {
        String baseName = "resources.HelloWorld";
        try {
            ResourceBundle bundle =
                ResourceBundle.getBundle( baseName );
            System.out.println( bundle.getString("HelloWorld") );
        } catch ( MissingResourceException e ) {
            System.err.println( e );
        }
    }
}
```

Anderer Sprachen und Regionen



```
C:\WINDOWS.1\system32\cmd.exe

D:\Develop\eclipse\test>java -classpath . -Duser.language=de -Duser.region=DE I18NHelloWorld
Hallo Welt!
Tsch3ss.

D:\Develop\eclipse\test>java -classpath . -Duser.language=de -Duser.region=EN I18NHelloWorld
Hallo Welt!
Tsch3ss.

D:\Develop\eclipse\test>java -classpath . -Duser.language=en -Duser.region=EN I18NHelloWorld
Hello world.
Bye.

D:\Develop\eclipse\test>java -classpath . -Duser.language=en -Duser.region=US I18NHelloWorld
Hello world.
Bye.

D:\Develop\eclipse\test>java -classpath . -Duser.language=fr -Duser.region=FR I18NHelloWorld
java.util.MissingResourceException: Can't find bundle for base name resources.HelloWorld, locale fr
FR

D:\Develop\eclipse\test>_
```

??? *Fragen*



**Haben Sie
Fragen?**



2. Beispiel

```
import java.util.MissingResourceException;
import java.util.ResourceBundle;
import java.util.Locale;

public class I18NHelloWorld2 {
    public static void main( String[] args )
    {
        String baseName = "resources.HelloWorld";

        try {
            ResourceBundle bundle = ResourceBundle.getBundle( baseName );
            System.out.println( bundle.getString("HelloWorld") );
            System.out.println( bundle.getString("Bye") );
        } catch ( MissingResourceException e ) {
            System.err.println( e );
        }
    }
}
```



3. Beispiel

```
import java.util.MissingResourceException;
import java.util.ResourceBundle;
import java.util.Locale;

public class I18NHelloWorld3 {
    public static void main( String[] args ) {
        String baseName = "resources.HelloWorld";
        Locale.setDefault( new Locale("en", "GB") );
        try {
            ResourceBundle bundle = ResourceBundle.getBundle( baseName );
            System.out.println( bundle.getString("HelloWorld") );
            System.out.println( bundle.getString("Bye") );
        } catch ( MissingResourceException e ) {
            System.err.println( e );
        }
    }
}
```

??? Fragen



***Welche Fragen
gibt es?***

JETZT

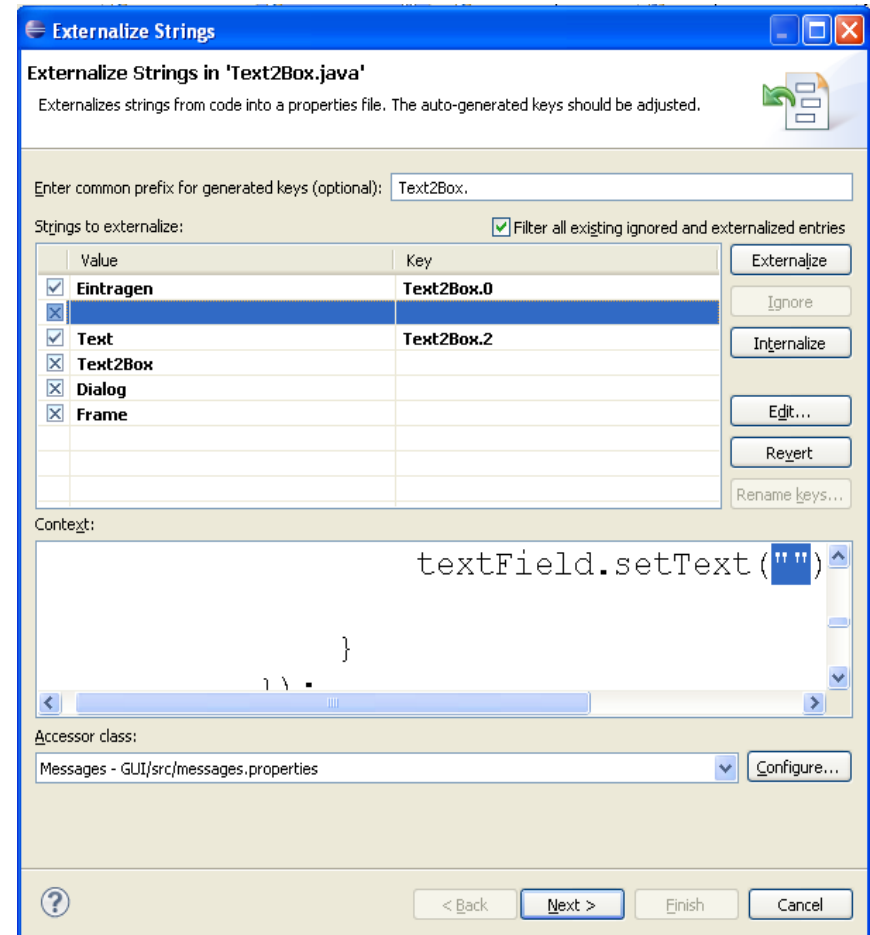
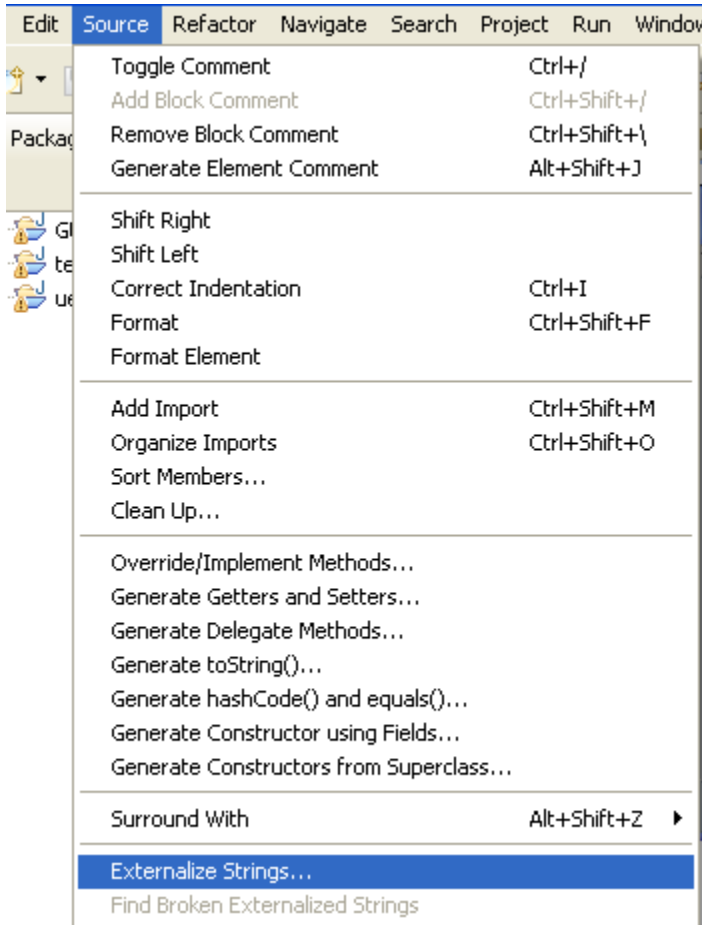


Internationalisierung mit eclipse

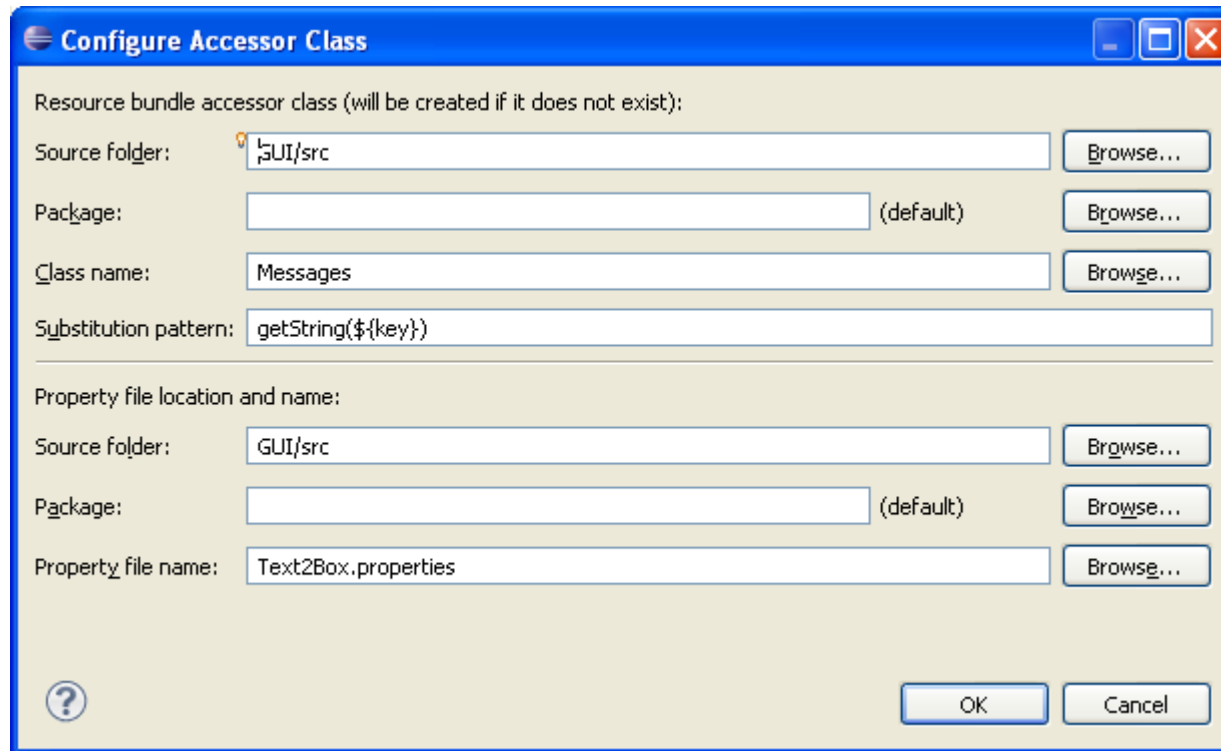
Mit Eclipse

- Eigene Funktion um feste Strings in Ressourcen zu wandeln
 - ◆ Externalize String
- Eclipse, WB und Locale ermöglichen damit I18N
- Auch nachträglich

3. Beispiel



3. Beispiel



```
public class Messages {  
    private static final String BUNDLE_NAME = "Text2Box"; //$NON-NLS-1$  
  
    private static final ResourceBundle RESOURCE_BUNDLE =  
        ResourceBundle.getBundle(BUNDLE_NAME);  
  
    private Messages() {  
    }  
  
    public static void update() {  
        Messages.RESOURCE_BUNDLE =  
            ResourceBundle.getBundle(Messages.BUNDLE_NAME);  
    }  
  
    public static String getString(String key) {  
        try {  
            return RESOURCE_BUNDLE.getString(key);  
        } catch (MissingResourceException e) {  
            return '!' + key + '!';  
        }  
    }  
}
```

JETZT



Qualitätssicherung

- Qualitätssicherung mittlerweile bei der Softwareentwicklung einen wichtigen Stellenwert.
- Alle Teile eine Software durchlaufen heutzutage Qualitätssicherungsprozesse
- Der Teil der Internationalisierung / Lokalisierung nimmt hierbei eine eigene Position mit eigenen Aufgaben und Techniken ein.
- Verschiedene Firmen/Projekte/Teams können hierbei verschiedene QA Prozesse verwenden



- Für die Qualitätssicherung/Tests für I18n mit Java können Sie sich an folgenden Richtlinien und Techniken orientieren.
- (https://web.archive.org/web/20120816030156/http://developers.sun.com/solaris/articles/i18n/I18N_Testing.html)
- Grundsätzlich werden alle Tests vorgehensweisen in einem sogenannten QS-Handbuch festgehalten. Prozeduren zum Test in Testplänen, die nach festgelegten Schema/Zyklen abgearbeitet werden



Test lokalisierter Text-Ressourcen

- Kommen alles Text aus dem Katalog
 - ◆ Automatische Pseudo-Übersetzung aller Labels
- Sprachspezifische Dateien prüfen
 - ◆ Werden alle sprachspezifische Dateien gefunden?
- Stimmen die Größen der Texte bei GUI Objekten, wird die Größe richtig angepasst?
 - ◆ Pseudo-Übersetzung mit kurzen / mittleren / langen Texten

Test lokalisierter „nicht“ Text-Ressourcen

- Jedes Produkt hat auch „nicht Text“ Ressourcen
 - ◆ Images / Icons
 - ◆ Soundfiles
 - ◆ Hilfe
 - ◆ Use.
- Test dieser Daten durch Test mit Pseudodaten

Test: Datum, Zeit, Zahlen, Sortierte Daten

- Prüfe alle Anzeigen folgender Funktionen/Daten
 - Uhrzeit und Datumsfunktionen
 - Nummer + Separatoren / Dezimalpunkte
 - Währungen + Separatoren / Dezimalpunkte
 - Suchen
 - Sortierungen und deren Reihenfolge
 - Gebietsschema- und sprachspezifische Silbentrennung, Zeilenumbruch und Interpunktion
- In jedem Gebietsschema:
Alle Vorkommen anzeigen und prüfen.

- Aufruf aller Produktfunktionen die Daten mit multibyte / extended ASCII erzeugen
- Verifikation der gedruckten / geschriebenen Daten
- Wiedereinlesen dieser Daten.

Test: Rückfall auf Standard Gebietsschema

- Was passiert wenn sich der Benutzer in einem Gebietsschema befindet in dem Teile nicht lokalisiert sind
 - ◆ Ausgabe in Standard Gebietsschema?
 - ◆ Nachricht dass gewähltes Gebietsschema nicht/tlw. nicht vorhanden?
 - ◆ Programm beendet sich?
 - ◆ Programm stürzt ab?



Test: Überprüfung der Anzeige

- Werden Ausgaben (Commandline / GUI) richtig angezeigt.
 - ◆ Multibyte Text / Extended ASCII

- Test mit Pseudo-Übersetzung (Prefixing) für Multibyte Gebietsschemata bzw. multibyte Zeichen

- Beispiel: Japanisch:
 - ◆ "File not found" as "JAXXFile not foundXXJA"



Test: Eingabe Methoden, Tastatureingabe

- Test der Eingabe für
 - Extended ASCII
 - Multiybyte (ja, zh, zh_TW, ko)
 - Akzent Charakter (Umlaute)
 - Charakter via Shift/Ctrl/Alt / andere Keyboards

- Testen über manuelle Eingabe in Textfelder (ggf. automatisiert [Maven, AutoLt, Jmeter, Selenium usw.]

Test: Hilfe System and Hilfe Database Suche

- Einige Hilfe-System nutzen HTML/ Browser
- Daher i18N tests
 - ◆ Generell
 - ◆ Browser spezifisch und Encoding (Folie: 26)
- Test: Pseudo Lokalisation (+multibyte)
- Test in verschiedenen Gebietschemata, Prüfung ob geändertes oder default Schema benutzt wird
- Bei Suchfunktionen/DB-Suchfunktion Test, ob gebietsspezifische Daten/Dateien und Daten mit multibytes gefunden werden



Test: Browser und I18n-Encodings

Einige Produkte verwenden einen Webbrowser (ggf. für das Hilfesystem), deshalb müssen neben den grundlegenden Funktionen alle speziellen Funktion getestet werden:

- Wie reagiert Browser auf Dateien mit verschiedenen Encodings?
- Wird multibyte text korrekt dargestellt / gedruckt?
- Kann Encoding geladener Seite umgestellt+ angezeigt werden.
- Wird das <META> -Tag für Encoding einer Seite benutzt.
- Dürfen/Können Dateien im gewählten Encoding gespeichert werden?
- Behalten besuchte Seiten d. Encoding, wenn Sie einmal geladen wurden?
- Wird HTML Quellcode im Editor im ausgewählten Encoding dargestellt?
- Werden Bookmarks usw. mit Multibytezeichen richtig dargestellt?

Test: Browser und I18n-Encodings II

Test: multibyte text:

Wenn das zu testende Betriebssystem die Anzeige von Multibytetext erlaubt.

- Prüfung ob Mix von Einzel- und Multibyte-Zeichen funktioniert (siehe vorherige Folie).
- Prüfung von Dateien/Seiten mit der größtmöglichen Anzahl verschiedener HTML Tags (Forms, Frams, Tables, ggf. JS) in Verbindung mit Multibytezeichen
- Verwendung von Seiten mit <META> tags die ein anderes Encoding anzeigen. Beispiel;

```
<META> HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=x-sjis">
```

Hiermit wird SJIS – Encoding für Japanisch eingestellt



Wenn Compiler Teil des Produktes sind, müssen dessen i18N Gebiete zusätzlich zu den Basistest durchgeführt werden:

- Parst der Compiler Multibyte-Werte, Wo dürfen sie vorkommen: Kommentare, Strings, Argumente, Optionen
- Kommen die Compiler-Meldungen aus Dateien die übersetzt werden können (Ggf. in gebietspezifischen Verzeichnissen).

Test:

- Dateien anlegen mit multibyte text: Z.b. Schlüsselwörter, Kommentare, Strings usw. (wenn vom Compiler erlaubt) und in verschiedenen Schemata compilieren und testen..
- Prüfen ob multibyte Dateinamen und Verzeichnisse für Dateien benutzt werden können. (compilieren und testen)
- Compiler in einem Schemata aufrufen, in dem Compiler -Meldungen übersetzt und installiert sind (und/oder Pseudo Lokalisierung)
- Außerdem siehe: Message Catalog Testing

Verarbeitung von Multibyte Text und Daten

Verarbeitet, liest, speichert oder zeigt das Produkt (oder Teile davon) Pfade, Dateiinhalte oder andere Daten mit Multibyte- bzw. Extended ASCII charactern in verschiedenen Encodings an.

Beispiele:

- Dateinamen, Pfade, (Class-Namen, Archive, HTML-Dateien, Textdateien, Kommentare,)
- Tool tips, Statusleisten
- Applet oder andere HTML- Text oder ähnliches (codebases, silverlight, flash, usw.)
- Kommandozeilen parameter oder Produkt-Optionen
- Registrierungs-, Kommentar- oder Feedbackfunktionen
- Browser: Pfade. Seitennamen, Dateinamen
- Gedruckte Informationen
- Datenbankfunktionalitäten
- Usw.

Verarbeitung von Multibyte Text und Daten

Test:

- Analysieren Sie das Produkt und erstellen Sie eine Liste:
- welche Bereiche Dateien, Pfade, Dateiinhalte und andere Informationen nutzen.
- Das Produkt in anderen Gebietsschemata läuft indem Multibyte oder Extended ASCII zulässig sind.
- Prüfen Sie ob diese Inhalte korrekt geparsed, verarbeitet und ausgegeben werden.



Source Code Tests und Reviews

- Prüfung dass keine hart-kodierten Strings im Quelltext (alles in Ressource-Dateien)
- Ggf. können Strings die nicht in ein i18n Katalog gehören speziell „ausgezeichnet“ werden (NOI18N)
- Wurden die Ressource-Dateien geprüft auf (manuelle / IT-Übersetzer):
 - ◆ Kommentare für Übersetzer vorhanden wenn benötigt
 - ◆ Gibt es fragmentierte Meldungen
 - ◆ Gibt es Meldungen die auf Grund von Argumenten umsortiert werden (sprachlich).
 - ◆ Sind die Meldungen klar, präzise und unmissverständlich
- Prüfung auf:
 - ◆ doppelte Ressource-Identifiers (Keys im Ressource-File),
 - ◆ Prüfung von umgebrochenen Strings,
 - ◆ Sind die Ressource-Dateien korrekt

Test: Email, Registrierung, Installation, usw.

- E-mail Funktionen
- Registrierung
- Installations-Tools
- Lizenzierung
- Hilfe, Anweisungen, Verpackungsinstruktionen usw.
- Tools von Drittfirmen
- Test/Verifikation (nach vor genannten Methoden)

■ I18n Tests ...

- ◆ ... sind umfangreich
- ◆ ... bedürfen Planung
 - (Testkatalog, Projektplanung)
- ◆ ... benötigen geschultes Personal
- ◆ ... benötigen Testumgebungen
- ◆ ... benötigen Kontrolle
- ◆ ... können tlw. automatisiert werden