Handout zur Präsentation: Einführung in Regular Expressions (Regex) in C#

---

1. Einführung: Was sind Regular Expressions (Regex)?

Regex sind Muster, die verwendet werden, um Zeichenfolgen zu durchsuchen, zu validieren oder zu manipulieren.
Verwendungszwecke:

- **Validierung von Eingaben** (z. B. E-Mail-Adressen, Telefonnummern)
- **Suchen und Ersetzen** von Text
- **Extraktion von Informationen** aus Textdaten
- **Textanalyse** und **Textmanipulation**

---

2. Grundlagen der Regex-Syntax

Element	Bedeutung
.	Beliebiges Zeichen außer Zeilenumbruch
*, +, ?	Quantifizierer für Wiederholungen
[]	Zeichenklassen (z. B. [a-z])
^, $	Anfang bzw. Ende einer Zeile
\	Escape-Zeichen (z. B. \d für Ziffern)
()	Gruppierung und Capturing

---

3. Regex in C#

Namespace: System.Text.RegularExpressions

Wichtige Klassen:

- **Regex**: Für die Definition und Ausführung von Regex-Mustern
- **Match**: Einzelnes Match-Ergebnis
- **MatchCollection**: Sammlung aller gefundenen Matches

---

4. MatchCollection

- **Definition:** Eine Sammlung von Match-Objekten, die alle gefundenen Übereinstimmungen eines Regex-Musters enthält.
- **Erstellung:** Wird durch die Methode Regex.Matches() erzeugt.

Anwendungsbeispiel:

```
string text = "Hier sind Zahlen: 10, 20, 30.";
Regex regex = new Regex(@"\d+");
MatchCollection matches = regex.Matches(text);

foreach (Match match in matches)
{
   Console.WriteLine(match.Value); // Gibt 10, 20, 30 aus
}
```

Wichtige Eigenschaften:

- **Count**: Anzahl der gefundenen Matches
- **IsReadOnly**: Gibt an, ob die Sammlung schreibgeschützt ist
- **Item[int index]**: Zugriff auf ein Match über den Index

---

5. Typische Anwendungsbeispiele

- **Validierung von E-Mail-Adressen:**

```
Regex emailRegex = new Regex(@"^[\w\.-]+@[\w\.-]+\.\w+$");
```

- **Ersetzen von Datumsformaten:**

```
Regex dateRegex = new Regex(@"\d{4}-\d{2}-\d{2}");
string result = dateRegex.Replace("2024-12-03", "03.12.2024");
```

- **Extraktion von Wörtern:**

```
Regex wordRegex = new Regex(@"\b\w+\b");
```

---

6. Häufige Fehlerquellen und Debugging-Tipps

Typische Fehler:

- Falsches **Escaping** (\ muss in C# als \\ geschrieben werden).
- Fehlerhafte **Quantifizierer** oder **Gruppierungen**.
- Nicht berücksichtigte **Sonderzeichen**.

Debugging-Tipps:

- Verwenden von Tools wie Regex101 zur Visualisierung und Analyse.
- Schreiben von **Unit-Tests** für verschiedene Eingaben.

---

7. Fazit

- **Regex** ist ein mächtiges Werkzeug für die Arbeit mit Texten in C#.
- Es ermöglicht die effiziente Lösung komplexer Textoperationen.
- Beachte jedoch die Performance und Lesbarkeit des Codes.

Für eine Übersicht zu Regex empfehlen wir: https://www.rexegg.com/regex-quickstart.php und https://regex101.com/

Präsentiert von:
Mert Iscan, Christian Vasiu, Dennis Wagner
IT22a – 03.12.2024