שם משתמש | סיסמה | התחבר | זכור אותי! | שחזור סיסמה | הרשמה

ראשי | העלאת תמונות | בדיקת IP | עיתון | קפיצה מהירה | קישורים מהירים

160,985,481 הודעות | 16,967,859 אשכולות | 1,146,952 רשומים | 20,089 מחוברים כרגע

בנט דורש מנתניהו: להפקיע משר הביטחון סמכויות לפינוי בתים | 6.5 שקלים במקום 125: הפחתה בעמלות הבנקים | "הדמות של טנצר המניאק תקעה אותי"

דף הבית | תכנות ובניית אתרים | שפות עיליות ואסמבלי | [#C]מדריך לביטויים רגולריים

הרשמה לאתר | הרשמה באמצעות facebook

[#C]מדריך לביטויים רגולריים

#1 25-04-2011 18:10

talikag
מנהל פורום C++/C#/Java לשעבר

תאריך הצטרפות: 14-01-08
הודעות: 3,886

רקע - מה זה ביטויים רגולריים?

נניח שאתם בונים תוכנית לשליחת וקבלת דואר אלקטרוני, ובמהלך התוכנית אתם צריכים לקלוט מהמשתמש כתובת אימייל, ולודא שהיא אכן כתובת חוקית (כלומר, שהיא מהצורה word1@word2.word3, כאשר word1 מורכבת רק מאותיות (a-z, A-Z), מספרות (0-9) ומקו תחתון (_), word2 מורכבת רק מאותיות ומספרות, ו-word3 מורכבת מאותיות בלבד). איך הייתם עושים זאת? מתכנת שלא מכיר את הביטויים הרגולריים, היה עושה כנראה משהו כזה:

קוד:

```
        public bool IsLetter(char ch)
{
return ('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z');
}
public bool IsDigit(char ch)
{
return ('0' <= ch && ch <= '9');
}
public bool IsValidEmail(string email)
{
int i = 0;
while (i < email.Length && email[i] != '@')
{
if (!(IsLetter(email[i]) || IsDigit(email[i]) || email[i] == '_'))
return false;
}
if (i == email.Length)
return false;
i++;
while (i < email.Length && email[i] != '.')
{
if (!(IsLetter(email[i]) || IsDigit(email[i])))
return false;
}
if (i == email.Length)
return false;
i++;
while (i < email.Length)
{
if (!IsLetter(email[i]))
```

עברה ל BASH
חזרה לנושא הכלה\ירושה

זה יעשה את העבודה, כמובן, אבל למה היינו צריכים לעבור את הגהנום הזה כדי לבצע התאמת מחרוזות כל כך פשוטה?

כאן הביטויים הרגולריים נכנסים לתמונה. ביטויים רגולריים (באנגלית: Regular Expressions, או בקיצור Regex), הם דרך מהירה ונוחה לבצע התאמת מחרוזות (String Maching), כלומר, למצוא במחרוזת נתונה תבניות מותאמות-אישית.

כדי לקבל טעימה מהכח של ביטויים רגולריים, בואו נראה איך מתכנת שמכיר את הביטויים הרגולריים היה כותב את הפונקציה IsValidEmail:

קוד:

```
        public bool IsValidEmail(string email)
{
return Regex.Match(email, @"^[a-zA-Z0-9_]+@[a-zA-Z0-9]+\.[a-zA-Z]+$").Success;
}
```

עדיף, לא?

ביטויים רגולריים בסיסיים

בהתחלה, ייתכן שהביטויים הרגולריים ייראו לכם קצת מפחידים ולא מובנים. עם זאת, אחרי שתכירו את את החוקים של כתיבת ביטויים רגולריים, תמצאו שזה פשוט מאד לכתוב ולהבין ביטויים רגולריים.

- **|** - ביטוי "או". למשל, הביטוי הרגולרי a|b יקבל את המחרוזת "a" וגם את המחרוזת "b".
- **()** - סוגריים, המאפשרים להגדיר סדר. למשל, הביטוי הרגולרי hi|bye יקבל את המחרוזת hi ואת המחרוזת bye, אך הביטוי הרגולרי h(i|by)e יקבל את המחרוזת hie ואת המחרוזת hbye.
- **-** - טווח של תווים סמוכים. למשל, במקום לכתוב a|b|c|...|z, נוכל לכתוב בקיצור a-z.
- **[]** - אוסף של תווים. למשל, הביטוי הרגולרי [ab0-9] יתאים למחרוזת "a", למחרוזת "b", וגם למחרוזת "0", "1", "2", וכן הלאה עד "9".
- **[^]** - אוסף של תווים, אך הפעם, הביטוי יקבל כל תו שאינו נמצא באוסף. למשל, הביטוי הרגולרי [^a-zA-Z] יקבל כל תו **שאינו** אות לועזית (לדוגמא, המחרוזות "0", "\r", "{" יתקבלו, אך המחרוזת "k" לא תתקבל).
- **^** - יחפש התאמה לביטוי הרגולרי אך ורק בתחילת המחרוזת. למשל, הביטוי הרגולרי ^hurf! יקבל את כל המחרוזות שמתחילות ב"hurf!".
- **$** - יחפש התאמה לביטוי הרגולרי אך ורק בסוף המחרוזת. למשל, הביטוי הרגולרי ^hurf!$ יקבל את כל המחרוזות שמתחילות ונגמרות ב"hurf!" כלומר, ביטוי זה יקבל אך ורק את המחרוזת "hurf!".
- **+** - מציין שהביטוי שלפניו יופיע לפחות פעם אחת. לדוגמא, הביטוי הרגולרי +(a|b) יקבל את המחרוזות a,b,aa,ab,ba,bb,aaa, וכן הלאה. כלומר, ביטוי זה יקבל כל מחרוזת שמורכבת רק מ-aים ו-bים.
- ***** - מציין שהביטוי שלפניו יופיע 0 פעמים או יותר. למשל, הביטוי הרגולרי *[0-9][1-9] יקבל כל מספר טבעי.
- **?** - מציין שהביטוי שלפניו יופי 0 או 1 פעמים. למשל, הביטוי הרגולרי http://)?www.google.com) יקבל את המחרוזות "www.google.com" ו-"http://www.google.com".
- **{n}** - מציין שהביטוי שלפניו יחזור על עצמו בדיוק n פעמים. לדוגמא, הביטוי הרגולרי [abc]{5} יקבל כל מחרוזת שאורכה 5, שמורכבת מהתווים a,b ו-c.
- **{m,n}** - מציין שהביטוי שלפניו יחזור על עצמו לכל הפחות m פעמים, ולכל היותר n פעמים. לדוגמא, הביטוי (ab){2,4} יקבל את המחרוזות abab, abababab ו-abababab.
- **{m,}** - מציין שהביטוי שלפניו יחזור על עצמו לפחות m פעמים. לדוגמא, הביטוי a{2,} יקבל את המחרוזות aa, aaa, aaaa, aaaaa וכן הלאה.
- **.** - מציין תו כלשהו (חוץ מהתו '\n'). למשל, הביטוי הרגולרי "^a.*b$" יקבל את כל המחרוזות שמתחילות ב-a ומסתיימות ב-b.
- **\d** - מציין תו שהוא ספרה. שקול ל-[0-9].
- **\D** - מציין תו שאינו ספרה. שקול ל-[^0-9].
- **\w** - מציין תו שהוא ספרה, אות (לא רק באנגלית) או קו-תחתון.
- **\W** - מציין תו שהוא לא ספרה, אות (לא רק באנגלית) או קו-תחתון.
- **\s** - מציין תו שהוא white space. שקול ל-[\f\n\r\t]
- **\S** - מציין תו שאינו white space. שקול ל-[^\f\n\r\t]
- **\b** - מציין תו שמפריד בין תו מסוג \w (אות, ספרה או קו-תחתון) לתו מסוג \W (תו שאינו אות, ספרה או קו-תחתון). בדרך כלל נשתמש בתו זה כדי לזהות את הקצוות (התחלה וסיום) של מילים.
- **** - escape character. נשים אותו לפני תווים מיוחדים (|,(,),-,[,],^,$,+,*,?,},{) ותווים מיוחדים נוספים שלא הוזכרו פה) על מנת להפוך אותם לתווים רגילים. למשל, הביטוי הרגולרי a\|b יקבל את המחרוזת "a|b", בעוד שהביטוי הרגולרי a|b יקבל את המחרוזת "a" ואת המחרוזת "b".

דוגמאות

1. **בדיקת תקינות של אימייל** (הדוגמא שהוצגה בתחילת המדריך)

קוד:

```
^[a-zA-Z0-9_]+@[a-zA-Z0-9]+\.[a-zA-Z]+$
```

הסבר: בהתחלה נרצה רצף כלשהו של אותיות, ספרות וקו תחתון. לאחר מכן נצפה לתו @. אחר כך נרצה רצף כלשהו של אותיות וספרות. לאחר מכן נצפה לתו ., ולאחריו נצפה לרצף של אותיות בלבד.

2 . **בדיקת תקינות פורמט של תאריך מהצורה MM/DD/YY HH:MM:SS**

קוד:

```
(\d|\d\d)/(\d|\d\d)/(\d\d(\d\d)?) (\d|\d\d):(\d|\d\d):(\d|\d\d)
```

3 . **בדיקת תקינות של כתובת IP**

קוד:

```
^([01]?\d\d|2[0-4]\d|25[0-5])\.([01]?\d\d|2[0-4]\d|25[0-5])\.([01]?\d\d|2[0-4]\d|25[0-5])\.([01]?\d\d|2[0-4]\d|25[0-5])$
```

4 . **מציאת לינקים במסמך HTML**

קוד:

```
<A[^>]*?HREF\s*=\s*["']?([^'" >]+?)[ '"]?>
```

נערך לאחרונה על ידי talikag, 26-04-2011 בשעה 00:43

הגב עם ציטוט

תאריך שחרור ל-DLC החדש של Rise of the Tomb Raider הוכרז

סיכום שנה בתחום הגיימינג

נוסף מצב מולטיפלייר ל-MGSV

סיכום השנה באקטואליה

Samsung תיקנה את
בעיית ה־S-Pen
ב־Note 5

ההרחבה הבאה של
The Witcher 3 "יותר
טובה מהמשחק
הראשי"

#2 25-04-2011 18:12

talikag

מנהל פורום C++/C#/Java לשעבר

תאריך הצטרפות: 14-01-08
הודעות: 3,886

ביטויים רגולרים בשפת C#

ב-C#, מתחם השמות System.Text.RegularExpressions מכיל את כל המחלקות של ביטויים רגולריים. המחלקות השימושיות ביותר הן Regex, Match ו-MatchCollection.
המחלקה Match מכילה מידע על התאמת מחרוזת אחת.
המחלקה MatchCollection מכילה אוסף של אובייקטים מסוג Match.
המחלקה Regex מכילה אובייקט של ביטוי רגולרי. יש לה 3 מתודות חשובות:
1. Match - בודק האם יש התאמה של הביטוי הרגולרי על מחרוזת נתונה.
2. Matches - מחזיר אוסף של התאמות (MatchCollection) של הביטוי הרגולרי על מחרוזת נתונה.
3. Replace - מחליף כל התאמה במחרוזת הקלט עם מחרוזת אחרת.

דוגמא פשוטה:

קוד:

```
        public void SimpleRegexMatch(string input)
{
Regex regex = new Regex("I|Love|Rockenroll");
Match match = regex.Match(input);
if (match.Success)
Console.WriteLine("The string {0} matches! The first match is: {1}", input, match.ToString());
else
Console.WriteLine("The string {0} does not mach!", input);
}
```

הפונקצייה מקבלת מחרוזת ובודקת אם היא מתאימה לביטוי הרגולרי "I|Love|Rockenroll". עבור כל המחרוזות שמכילות את תת-מחרוזות "Love" ,"I" או "Rockenroll" תהיה התאמה. לכל שאר המחרוזות, לא תהיה התאמה.

לפעמים ייתכן שתהיה יותר מהתאמה אחת, כי ביטויים רגולרים מחפשים התאמה **בתוך הטקסט** ולא רק התאמה על **הטקסט כולו**. למשל, עבור הביטוי הרגולרי "a.b" והקלט "acbadbaeb" ימצאו 3 התאמות: "acb", "adb" ו-"aeb".
ישנן 2 דרכים למצוא את כל ההתאמות, האחת היא באמצעות קריאה למתודה Matches, והשניה באמצעות קריאות חוזרות ונשנות למתודה Match:

קוד:

```
        public void FindAllMatches1(string input)
{
Regex regex = new Regex("I|Love|Rockenroll");
MatchCollection matches = regex.Matches(input);
for (int i = 0; i < matches.Count; i++)
{
Console.WriteLine("Match #{0}", i + 1);
Console.WriteLine("\tThe match is: {0}", matches[i].Value);
Console.WriteLine("\tThe match starts at index {0}, and  its length it {1}", matches[i].Index, matches[i].Length);
}
}
```

קוד:

```
        public void FindAllMatches2(string input)
{
Regex regex = new Regex("I|Love|Rockenroll");
Match match = regex.Match(input);
int i = 1;
while (match.Success)
{
Console.WriteLine("Match #{0}", i);
Console.WriteLine("\tThe match is: {0}", match.Value);
Console.WriteLine("\tThe match starts at index {0}, and its length it {1}", match.Index, match.Length);
match = match.NextMatch();
i++;
}
}
```

אם נעביר כפרמטר את המחרוזת "I Hate Rockenroll" לאחת מהפונקציות הנ"ל, נקבל את הפלט הבא:

קוד:

```
Match #1
The match is: I
The match starts at index 0, and its length it 1
Match #2
The match is: Rockenroll
The match starts at index 7, and its length it 10
```

הערה: אם אתה מעוניינים לבדוק התאמה של ביטוי רגולרי רק על המחרוזת כולה, ולא למצוא התאמות שהם תת-מחרוזות של מחרוזת הקלט, אפשר לשים בתחילת הביטוי הרגולרי את התו ^ ובסופו את התו $. לדוגמא, הביטוי הרגולרי $bc^ יקבל את המחרוזת "bc", אך לא את המחרוזת "abcd" (בניגוד לביטוי הרגולרי bc, שיקבל את 2 המחרוזות הנ"ל).

המחלקה Group

לעיתים, נרצה לא רק למצוא התאמה לביטוי רגולרי, אלא גם לקבל מידע על ההתאמה שמצאנו. לדוגמא, נניח שהרצנו את הביטוי הרגולרי a(b|c)d על המחרוזת "abd". ייתכן שנרצה לדעת איזה מבין האותיות b ו-c נמצאו בהתאמה שמצאנו. כאן אמנם אפשר לעשות את זה ידנית (פשוט לבדוק אם ערך ההתאמה הוא "abd" או "acd"), אך זהו לא המצב כאשר משתמשים בביטויים רגולריים מסובכים יותר.

המחלקה Group מאפשרת לנו לקבל מידע כזה. כאשר אנו משתמשים בסוגריים בביטויים רגולריים, הם מגדירים לנו "קבוצה" או "Group". הקבוצה הזאת מספקת לנו מידע על תת-הביטוי שנתפס. למשל, הקבוצה (b|c) בביטוי הרגולרי שבדוגמא, תקבל את הערך "b" כאשר בהתאמה שנמצאה יש b בין האות a לאות d, ואת הערך "c" כאשר בהתאמה שנמצאה יש c בין האות a לאות d.

דוגמא נוספת: הביטוי הרגולרי http://)?www.[a-zA-Z0-9]+.com מתאים לכל המחרוזות שהן כתובות אינטרנט חוקיות עם הסיומת com. על מנת לבדוק אם הכתובת התחילה ב-http://, נוכל לבדוק אם ערך הקבוצה (http://) בהתאמה שנמצאה הוא "" או "http://". אם מדובר במקרה הראשון, אז המחרוזת לא מתחילה ב-http://. במקרה השני, היא כן מתחילה ב-http://

קוד:

```
        public void GroupsExample()
{
Regex r = new Regex("^(http://)?www.[a-zA-Z0-9]+.com$");
Match m = r.Match("http://www.google.com");
if (m.Success)
{
if (m.Groups[1].Value == "http://")
Console.WriteLine("The URL starts with http://");
else
Console.WriteLine("The URL doesn't start with http://");
}
}
```

שימו לב שעל מנת לזהות את הקבוצה (http://), פנינו לקבוצה שמספרה הסידורי הוא 1 (m.Groups[1]). עשינו זאת משום שהקבוצה במקום ה-0 (m.Groups[0]) היא תמיד הקבוצה שמתאימה להתאמה כולה (כלומר, אפשר לדמיין שהביטוי הרגולרי הוא למעשה **(^(http://)?www.[a-zA-Z0-9]+.com$)**). מכיוון שזאת הקבוצה הראשונה שמופיעה בביטוי הרגולרי, מספרה הסידורי יהיה

.1

אם לא רוצים לגשת לקבוצות לפי המספר הסידורי שלהם, ניתן לתת להם שם. למשל, ניתן להחליף את הביטוי הרגולרי הנ"ל בביטוי הבא: http://)?www.[a-zA-Z0-9]+.com(?<mygroup>), ואז לגשת אליה כך:

קוד:

```
m.Groups["mygroup"]
```

ישנה גם מחלקה בשם GroupCollection, אשר משמשת לאחזקת אוסף של קבוצות.

המחלקה Capture

המחלקה Group מספקת לנו מידע על תת-ביטויים שנתפסו, אבל ייתכן שקבוצה תתפוס יותר מביטוי אחד. לדוגמא, אם נריץ על הביטוי הרגולרי +(abc) על המחרוזת "dabcabcd", נקבל התאמה: "abcabc". עבור ההתאמה הזאת, ישנם **2** תתי-ביטויים שנתפסים על ידי הקבוצה abc. המחלקה Group מאפשרת לנו לקבל מידע רק על אחד מהם (על ה-abc האחרון),

אך לפעמים נרצה לקבל מידע על כל תתי-הביטויים שנתפסים.
לשם כך, ניתן להשתמש במחלקה Capture. מחלקה זו משמשת לייצוג תת-ביטוי בודד שנתפס על ידי קבוצה.
לכל קבוצה (Group) יש CaptureCollection - אוסף של אובייקטים של המחלקה Capture - שנותנים לנו מידע כל אחד מתתי-הביטויים שנתפסו על ידי הקבוצה.

קוד:

```
        public void CaptureExaple()
{
Regex r = new Regex("(abc)+");
Match m = r.Match("dabcabcd");
CaptureCollection captureCollection;
int pos, len;
for (int i = 1; i < m.Groups.Count; i++) //skip m.Grops[0]
{
captureCollection = m.Groups[i].Captures;
Console.WriteLine("Group #{0}", i);
for (int j = 0; j < captureCollection.Count; j++)
{
pos = captureCollection[j].Index;
len = captureCollection[j].Length;
Console.WriteLine("\tCapture #{0}: {1}", j+1, captureCollection[j].Value);
Console.WriteLine("\t\tStarts at index {0}", pos);
Console.WriteLine("\t\tLength is {0}", len);
}
}
```

הפלט עבור תוכנית זו הוא:

קוד:

```
Group #1
Capture #1: abc
Starts at index 1
Length is 3
Capture #2: abc
Starts at index 4
Length is 3
```

נערך לאחרונה על ידי talikag, 25-04-2011 בשעה 18:16

הגב עם ציטוט

#3 25-04-2011 18:12

talikag

מנהל פורום C++/C#/Java לשעבר

תאריך הצטרפות: 14-01-08
הודעות: 3,886

החלפת מחרוזות

לפעמים, נרצה לא רק למצוא תבניות מסויימות בתוך מחרוזת, אלא גם להחליף אותן במחרוזות אחרות. למשל, נניח שאנחנו רוצים לכתוב תוכנית שמקבלת טקסט כלשהו, והופכת כל מילה בו למילה שמתחילה באות גדולה, וכל שאר האותיות בה קטנות. למשל, עבור המחרוזת "wHaT WE always dO, pinKy", נרצה לקבל כפלט את המחרוזת "What We Always Do, Pinky".

מסתבר שניתן להשתמש בביטויים רגולריים גם על מנת לבצע החלפות. בדוגמא שלנו, למשל, נרצה להחליף כל מילה בטקסט. כלומר, נרצה להחליף כל התאמה לביטוי הרגולרי \w+.

ב-C#, החלפה באמצעות ביטויים רגולריים נעשית באמצעות המתודה Replace של המחלקה Regex. המתודה הזאת מקבלים 2 פרמטרים - הפרמטר הראשון הוא מחרוזת הקלט, והפרמטר השני הוא אובייקט של delegate שנקרא MatchEvaluator. ה-delegate הזה מצביע לפונקציה שמקבלת כפרמטר התאמה של הביטוי הרגולרי, ומחזירה את המחרוזת שבו ההתאמה הזאת תוחלף.

למשל, המתודה CapitalizeWords פותרת את הבעיה המתוארת למעלה.

קוד:

```
        private string CapitalizeWord(Match match)
{
string word = match.Value;
string newWord = word.ToLower();
return char.ToUpper(word[0]) + newWord.Substring(1);
}
public string CapitalizeWords(string text)
{
Regex regex = new Regex(@"\w+");
return regex.Replace(text, new MatchEvaluator(CapitalizeWord));
}
```

כעת, אם נריץ את השורה הבאה:

קוד:

```
Console.WriteLine(CapitalizeWords("wHaT WE always dO, pinKy"));
```

אז נקבל כפלט:

קוד:

```
What We Always Do, Pinky
```

ישנה דרך נוספת לבצע החלפות באמצעות ביטויים רגולריים, והיא מתבצעת באמצעות ביטוי רגולרי נוסף. הביטוי הרגולרי הנוסף מציין את הטקסט שבו תוחלף כל התאמה. מכיוון שבדרך כלל הטקסט המוחלף תלוי בדרך כלל בטקסט המקורי, ביטויים רגולריים מאפשרים לנו גישה לערך שנתפס על ידי הקבוצות (Groups). את ערכי הקבוצות מקבלים על ידי הסימן דולר ('$'), ולאחריו מזהה של הקבוצה הרלוונטית:

$i - ערך הקבוצה שמספרה הסידורי הוא i (גם פה המספר הסידורי הראשון הוא 1).
${group_name} - ערך הקבוצה ששמה הוא group_name.
$& - הטקסט המוחלף כולו (אפשר גם לכתוב $0)
$$ - הסימן $

למשל, נניח שאנחנו רוצים למחוק את כל ה-whitespaces שמופיעים בתחילת ובסוף הטקסט. נוכל לעשות זאת כך:

קוד:

```
Regex r = new Regex(@"^\s*(.+?)\s*$");
Console.WriteLine(r.Replace("      my untrimmed text          ", "$1"));
```

(המשמעות של ?+. היא שבהתאמה שנמצא נרצה שאורך רצף התווים שבין ה-whitespaces יהיה מינימלי).
כלומר, נתפוס את הטקסט שבין ה-whitespaces בקבוצה שמספרה הסידורי 1, ואז נחליף את הטקסט רק במה שנתפס על ידי הקבוצה - מה שיגרום לכך שכל ה-whitespaces שבהתחלה ובסוף ימחקו.

דוגמאות

1 . **מחיקת הערות מהצורה /* ... */ מקוד בשפת C#/Java/C++/C/Etc.**

קוד:

```
string uncommentedCode = Regex.Replace(originalCode, @"/\*.*?\*/", "");
```

2 . **החלפת 2 המילים הראשונות בטקסט**

קוד:

```
string swappedString = Regex.Replace(@"(\S+)(\s+)(\S+)", "$3$2$1", 1);
```

The RegexOptions enum

לפעמים נרצה לעבד טקסט מסויים עם הגדרות מיוחדות שמתאימות לו.
למשל, טקסט בעברית, נרצה לעבד מימין לשמאל, ולא משמאל לימין. נוכל לעשות זאת על ידי שימוש ב-RegexOptions.RightToLeft.
מקרה נוסף בו נזדקק להגדרות מיוחדות הוא כאשר מעבדים מסמך HTML. במקרה זה, נרצה שלא להבדיל בין אותיות קטנות לגדולות. נוכל לעשות גם את זה, הפעם בעזרת שימוש ב-RegexOptions.IgnoreCase.

על מנת להשתמש בהגדרות כאלו, נעביר אותן כפרמטר שני לבנאי של המחלקה Regex. לדוגמא:

קוד:

```
Regex r = new Regex(@"\w+", RegexOptions.IgnoreCase
```

ישנם כמובן הגדרות נוספות ש-RegexOptions מאפשר להשתמש בהם. לא ארחיב על השאר.

סיכום

ביטויים רגולריים הם דרך מהירה ונוחה למצוא ולהחליף תבניות מותאמות-אישית בתוך מחרוזות. הכרתם יכולה לחסוך לכם הרבה זמן וכאב ראש.

נערך לאחרונה על ידי talikag, 27-04-2011 בשעה 12:03

הגב עם ציטוט

#4 25-04-2011 18:19

Skyance2
FxP Idol

תאריך הצטרפות: 03-01-09
הודעות: 8,977

תעשה לי רג'אקס. :ם

הגב עם ציטוט

#5 25-04-2011 18:20

idan3250
Fxp Beginner

תאריך הצטרפות: 21-04-11
הודעות: 37

תודה
הכרתי את זה אבל לא ידעתי בדיוק מה משמעותו של כל סימן

הגב עם ציטוט

#6 25-04-2011 18:20

nadevill
FxP Supreme

תאריך הצטרפות: 10-01-09
שם פרטי: נדב
הודעות: 8,088

וואו זה כמו ללמוד שפה שלמה.
תודה רבה,וכל הכבוד על ההשקעה.

Nothing travels faster than the speed of light with the possible exception of bad news, which obeys its own special laws

הגב עם ציטוט

#7 25-04-2011 18:21

תאריך הצטרפות: 14-01-08
הודעות: 3,886

talikag

מנהל פורום C++/C#/Java לשעבר

> פורסם במקור על ידי **Skyance2**
>
> *תעשה לי רג'אקס. :ם*

תעשה לעצמך 😒

נערך לאחרונה על ידי talikag, 25-04-2011 בשעה 18:22

הגב עם ציטוט

#8 25-04-2011 18:21

תאריך הצטרפות: 02-10-08
שם פרטי: עידן
הודעות: 4,586

WelCoMe2DeT

FxP Master

וואי תשמע אתה מטורף

If one day the speed kills me, do not cry because I was smiling.
-
Paul Walker

הגב עם ציטוט

#9 25-04-2011 19:00

Uneffective
Banned

תאריך הצטרפות: 15-07-10
שם פרטי: אבא
הודעות: 608

ועכשיו מה שנשאר זה לשמור דף זה כסימניה ולפנות אליו לפי הצורך.

הגב עם ציטוט

#10 25-04-2011 19:01

PieThon
מנהל פורום שפות עיליות ואסמבלי לשעבר

תאריך הצטרפות: 01-01-09
הודעות: 8,309

כל הכבוד!!
איזה השקעה..

הגב עם ציטוט

#11 25-04-2011 19:07

Gilnaa
FxP Master

תאריך הצטרפות: 24-06-10
שם פרטי: גלעד
הודעות: 4,895

זאת הפעם השמינית שאני קורא מדריך לRegex, ואני עדיין לא זוכר כלום!

כנראה שאני צריך לממש בשביל לזכור

Ask a stupid question, get a stupid answer

הגב עם ציטוט

#12 25-04-2011 19:21

PaperWings

FxP Magnificent

תאריך הצטרפות: 18-08-06
הודעות: 27,785

התרגיל הבא שלי זה לבנות מחלקה למציאת תאריך ולמצוא תקינות,
זה כנראה יעשה את העבודה :O.

.Hacking means understanding the system, not breaking it

הגב עם ציטוט

#13 25-04-2011 20:46

PieThon

מנהל פורום שפות עיליות ואסמבלי לשעבר

תאריך הצטרפות: 01-01-09
הודעות: 8,309

מחרוזת יכולה להכיל רק אותיות באנגלית גדולות, קטנות ומספרים זוגיים
איך אני עושה לזה ביטוי רגולרי? פשוט לא עובד מה שאני מנסה..

נערך לאחרונה על ידי PieThon, 25-04-2011 בשעה 20:46

הגב עם ציטוט

#14 25-04-2011 21:09

talikag

מנהל פורום C++/C#/Java לשעבר

תאריך הצטרפות: 14-01-08
הודעות: 3,886

למה אתה מתכוון שאתה אומר "מספרים זוגיים"? ספרות זוגיות? אם כן, זה יעבוד:

קוד:

```
^[a-zA-Z02468]+$
```

אם אתה מתכוון ממש למספרים זוגיים, אז צריך לוודא שזה תואם לביטוי הרגולרי הבא:

קוד:

```
^[a-zA-Z0-9]+$
```

וגם שזה לא תואם לביטוי הרגולרי הבא:

קוד:

```
[13579](\D|$)
```

כלומר, שאין מספר שמסתיים בספרה אי-זוגית - אחרת המספר הזה הוא אי-זוגי. שים לב שמספר יכול להסתיים או באות שאינה ספרה או בסיום המחרוזת.

הגב עם ציטוט

#15 25-04-2011 21:20

daniel319
FxP Average

תאריך הצטרפות: 01-03-09
שם פרטי: דניאל
הודעות: 855

איזה השקעה!נראה קשה..
אני בדיוק הייתי צריך את זה אבל לא מצאתי מדריך טוב אז תודה רבה!
(נשמר במועדפים...)