

Ruby für Fortgeschrittene

Advanced Features

Die Programmiersprache Ruby ist eine objektorientierte und plattformunabhängige Interpreter-Sprache. Sie hat sich in wichtigen Bereichen der Anwendungsprogrammierung zu einer Alternative zu etablierten Sprachen wie Java oder Python gemausert. In dem Training werden einige fortgeschrittene Fähigkeiten von Ruby behandelt, die ein flexibleres und effizienteres Programmieren ermöglichen. Dabei wird auf einige Aspekte des Meta Programming wie z.B. dynamische Methoden und Dekoratoren sowie Good Practices eingegangen. Die Inhalte werden durch Übungsbeispiele in der Praxis begleitet und vertieft.

Kursinhalt

- Rekapitulation einiger Ruby-Grundlagen
- Metaprogramming
- Procs und Lambdas
- Dynamische Methoden
- Dekoratoren
- Good Practice
- Logging
- Rspec-Testing
- Dokumentation
- Praxis-Übungen

E-Book Sie erhalten das ausführliche deutschsprachige Unterlagenpaket aus der Reihe ExperTeach Networking – Print, E-Book und personalisiertes PDF! Bei Online-Teilnahme erhalten Sie das E-Book sowie das personalisierte PDF.

Zielgruppe

Der Kurs richtet sich an alle, die ihre vorhandenen Ruby-Kenntnisse verfeinern und ausbauen wollen.

Voraussetzungen

Grundkenntnisse der Ruby-Programmierung.

Dieser Kurs im Web



Alle tagesaktuellen Informationen und Möglichkeiten zur Bestellung finden Sie unter dem folgenden Link: www.experteach.ch/go/RUBA

Vormerkung

Sie können auf unserer Website einen Platz kostenlos und unverbindlich für 7 Tage reservieren. Dies geht auch telefonisch unter 06074 4868-0.

Garantierte Kurstermine

Für Ihre Planungssicherheit bieten wir stets eine große Auswahl garantierter Kurstermine an.

Ihr Kurs maßgeschneidert

Diesen Kurs können wir für Ihr Projekt exakt an Ihre Anforderungen anpassen.

Training		Preise zzgl. MwSt.
Termine in Deutschland	2 Tage	CHF 1.975,-
Online Training	2 Tage	CHF 1.975,-
Termin/Kursort	Kurssprache Deutsch	
30.09.-01.10.24	Düsseldorf	30.09.-01.10.24 Online

Stand 05.03.2024



Inhaltsverzeichnis

Ruby für Fortgeschrittene – Advanced Features

- 1 Wiederholung**
 - 1.1 Variablen
 - 1.2 Schleifen
 - 1.3 Methoden und Klassen
 - 1.4 Vererbung und Super
 - 1.5 Module
 - 1.6 Ausnahmen und Rescue
- 2 Metaprogramming**
 - 2.1 Vorbereitung
 - 2.1.1 Procs und Lambdas
 - 2.1.2 Send & define_method – Lösung
 - 2.2 Dynamische Methoden
 - 2.2.1 Methoden kopieren
 - 2.2.2 Methode erzeugt Methoden
 - 2.3 Dekoratoren
 - 2.3.1 Basisdekorator
 - 2.3.2 SimpleDelegator
 - 2.3.3 Hausgemachter Dekorator – method_missing
- 3 Good Practice**
 - 3.1 Was ist Logging und brauche ich es?
 - 3.1.1 Logger
 - 3.1.2 Formatter
 - 3.2 Rspec – Testing
 - 3.2.1 Weitere nützliche Matcher
 - 3.3 Dokumentation
 - 3.3.1 Dokumentation – besser gemacht
- 4 Übungen**
 - 4.1 Wiederholung
 - 4.1.1 Rekursive Methoden
 - 4.1.2 Vererbung
 - 4.1.3 Module und Ausnahmen
 - 4.2 Metaprogramming
 - 4.2.1 Lambdas und Procs
 - 4.2.2 Send & define_method
 - 4.2.3 Dynamische Methoden
 - 4.2.4 Methode erzeugt Methoden
 - 4.2.5 Basisdekorator
 - 4.2.6 SimpleDelegator
 - 4.2.7 Method Missing
 - 4.3 Good Practice
 - 4.3.1 Logger
 - 4.3.2 Formatter
 - 4.3.3 Automatische Tests – Fibonacci
 - 4.3.4 Automatische Tests – Stringsuche
 - 4.3.5 Automatische Tests zusammen mit Logger
 - 4.3.6 Dokumentation
 - 4.4 Wiederholung – Lösungen
 - 4.4.1 Rekursive Methoden – Lösung
 - 4.4.2 Vererbung – Lösung
 - 4.4.3 Module und Ausnahmen – Lösung
 - 4.5 Metaprogramming – Lösungen
 - 4.5.1 Lambdas und Procs – Lösung
 - 4.5.2 Send & define_method – Lösung
 - 4.5.3 Dynamische Methoden – Lösung
 - 4.5.4 Methode erzeugt Methoden – Lösung
 - 4.5.5 Basisdekorator – Lösung
 - 4.5.6 SimpleDelegator – Lösung
 - 4.5.7 Method Missing – Lösung
 - 4.6 Good Practice – Lösungen
 - 4.6.1 Logger – Lösung
 - 4.6.2 Formatter – Lösung
 - 4.6.3 Automatische Tests – Fibonacci – Lösung
 - 4.6.4 Automatische Tests – Stringsuche – Lösung
 - 4.6.5 Automatische Tests zusammen mit Logger – Lösung
 - 4.6.6 Dokumentation – Lösung

