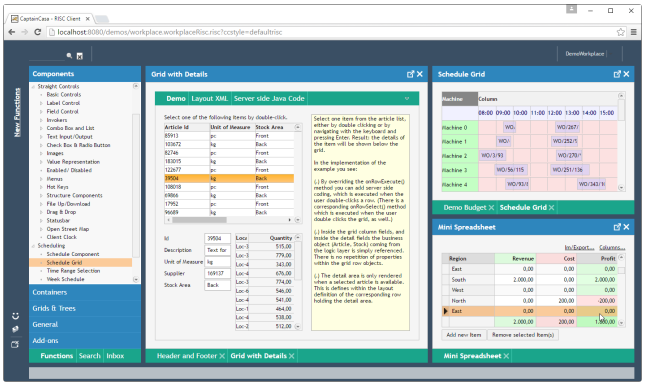
Take the "RISC"!
reduced instruction set client
A substantially different architecture for industry-stable HTML Web UIs.
browser-compatible by design
fast by design robust by design
zero installation - zero maintenance

CaptainCasa Enterprise Client RISC

Die „RISC“ (reduced instruction set client) Methode als Grundlage für stabile, performante Web UIs.

Die RISC Methode ist ein neues Architekturprinzip, das innerhalb des Browsers beim Aufbau von Web-Benutzeroberflächen angewendet wird. Die RISC Methode wurde konzipiert für die Entwicklung des Frontends umfangreicher, langlebiger Unternehmensanwendungen.

Die RISC Methode deckt einerseits die Anforderungen an moderne, ergonomische, qualitativ hochwertige Benutzeroberflächen ab - und lässt hinsichtlich derer Gestaltung alle Freiheiten. Gleichzeitig aber verbindet die RISC Methode diese Freiheiten mit einer neuen, Architektur-bedingten Qualität an langfristiger Stabilität und Browser-Unabhängigkeit.



Benutzeroberflächen im Browser bedeuten traditionell „zero installation“ für den Endbenutzer. Diese „zero installation“ wird beim Softwarehersteller aber teuer erkauft und bedeutet für ihn „endless maintenance“. Die RISC Methode löst diese Kopplung auf und erlaubt es, auch komplexe Benutzeroberflächen im Browser langfristig mit minimalem Wartungsaufwand zu betreiben: „zero installation“ beim Endkunden ist zum ersten Mal mit „zero maintenance“ beim Softwarehersteller verbunden.

MOTIVATION

Waren Sie als Softwarehersteller mit nativen Frontendimplementierungen Ihr eigener „Herr auf dem Frontend“, so erleben Sie bei Web-Oberflächen einen deutlichen Kontrollverlust auf verschiedenen Ebenen:

- **Browser-/Devicekompatibilität**: der Endanwender gibt den Browser vor. Und das bedeutet: Ihre Software muss sowohl auf heutigen Browsern - als auch vielmehr auf zukünftigen Browsern laufen.

  Beispiel: Stieg Microsoft im Bereich seiner Browser von der Generation „Internet Explorer“ auf nun „Edge“ um, dann waren und sind Sie es, der damit verbundene Probleme zu lösen hat.

  Installieren Sie Ihre Software beim Kunden („on premise“), dann verschärft sich die Situation: denn Sie müssen dafür sorgen, dass der jeweilige Versionsstand Ihrer Software auch auf den neuen Browsern zum Laufen kommt!

- **Layout-Flexibilität**: HTML kann einiges gut - das Layouten von Inhalten hat aber inhärente Schwächen. Diese kommen zum Beispiel beim vertikalen Layouten von Inhalten regelmäßig zum Vorschein. Generell erlaubt Ihnen HTML nicht, im Vorfeld des Seitenaufbaus die Größe von Elementen zu erfahren, um diese dann entsprechend zu positionieren.

- **Komplexität**: HTML bietet einen reichen Blumenstrauß an Elementen, Attributen und CSS-Steuerungsmöglichkeiten - die dann auch seitens der Entwicklung genutzt werden. Folge: die Komplexität der entwickelten Anwendungen steigt. Zudem stellte jede dieser Möglichkeiten eine potenzielle Fehlerquelle beim Umstieg von einem Browser auf den anderen dar.

  Beispiel: Stylesheet Definitionen (CSS) bei größeren Szenarien sind zumeist nicht mehr zu verstehen und besitzen eine Komplexität, an die sich „keiner mehr rantraut“.

Aus Sicht eines Softwareherstellers wird dieser Kontrollverlust von außen aufgezwungen - und ist deswegen gerade für umfangreiche, langlebige Softwareprodukte ein kostenintensives Risiko.

Nun gibt es unzählige Frameworks im Web-Bereich, die Ihnen dabei helfen (sollen), der beschriebenen Probleme Herr zu werden. Und, ganz klar: eine Reihe von ihnen lindern zumindest die Schmerzen - ohne das Problem aber bei der Wurzel zu packen

Traditionelle Frameworks verhüllen Browserinkompatibilitäten unterhalb einer Zugriffsschicht und bügeln diese entsprechend aus. Und dies bedeutet wiederum für Sie: auch in Ihrer Software, die z.B. schon beim Kunden im Einsatz ist, müssen Sie permanent den neuesten Frameworkstand verwenden, um die Browser der Anwender adäquat zu unterstützen. - Ihre Software kommt also nicht zu der Ruhe, die sie verdient!

DIE RISC-METHODE

Die RISC Methode ist eine Umkehrung des HTML Denkens: sie versucht **nicht**, HTML in einfacherer, kompatiblerer Weise zu verpacken. Sondern sie setzt unabhängig von HTML bei der Fragestellung an, welche Grund-Elemente man eigentlich braucht, um komplexe Oberflächen aufzubauen.

Welches sind sozusagen die Nukleus-Komponenten, die eine Rendering-Umgebung zur Verfügung stellen muss, um darauf aufbauend mit Hilfe einer Programmiersprache alle gängigen funktionalen Komponenten - vom Button bis hin zum komplexen Grid - effektiv und flexibel zusammenzubauen?

Innerhalb der RISC Methode ist die Antwort auf diese Frage, dass zwei grundlegende Elementarten substanziell ausreichend sind:

- „Rechtecke“: die Rendering-Umgebung muss Rechteckbereiche zeichnen können. Und zwar Rechtecke mit Textinhalt und mit Hintergrundgestaltung.

- „Texteingaben“: die Rendering-Umgebung muss Texteingabefelder beherrschen: einzeilig, mehrzeilig - mit entsprechender Cursor-/Selektionssteuerung.

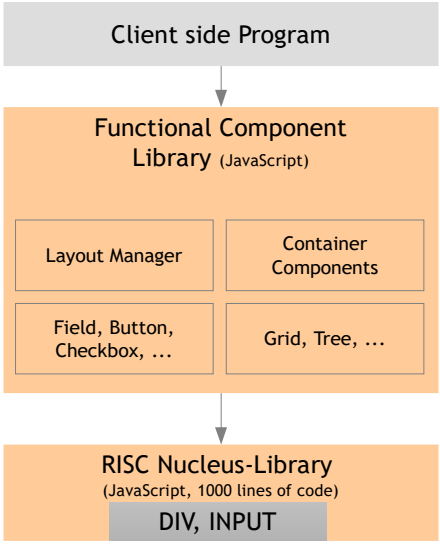
Seitens des Layoutings wird von der Rendering-Umgebung nur erwartet, dass die Elemente absolut positioniert werden können (x, y, width, height, (z)) und dass die Elemente ineinandergeschachtelt werden können.

Kurz: es wird eine minimale „Nukleus-Rendering-Umgebung“ definiert, um darauf aufbauend dann echte, funktionale Komponenten zu entwickeln.

DAS RISC WEB FRAMEWORK - „SCHMALSPURNUTZUNG“ DES BROWSERS

Diese RISC Methode wird nun auf HTML abgebildet. Hierbei bietet der Browser entsprechende Elemente (DIV für Rechteckbereiche, INPUT für Eingaben), die genau den einfachen Anforderungen gerecht werden, die an sie gestellt werden: absolute Positionierbarkeit, Hintergrundgestaltung. Und der Browser bietet mit JavaScript eine Programmiersprache und -umgebung, über die diese Elemente zu Komponenten aggregiert werden können.

Innerhalb der RISC Methode wird also der Browser und seine HTML/CSS-Technologien nur in äußerst begrenzter, sehr einfacher Art und Weise genutzt.



Die Rolle des Browsers ist die einer „Rendering-Execution-Engine“. Layouting-Entscheidungen werden nicht durch ihn getroffen sondern durch die programmierte Ebene oberhalb der Nukleus-Elemente. Der Browser weiß, **dass** ein DIV an eine bestimmte Koordinate gezeichnet wird - er weiß aber nicht **warum**.

Für manche (und uns am Anfang auch!) überraschend ist, dass über die RISC Methode erstellte Web Oberflächen innerhalb der heutigen Browser eine hervorragende Performance haben! Das Zeichnen von Rechtecken auch in sehr großer Zahl ist für den Browser geradezu eine Unterforderung. JavaScript als Umgebung für den Ablauf der darüber gelagerten funktionalen Komponenten wird mittlerweile mit einer beeindruckenden Performance ausgeführt.

SUBSTANZIELLE PROBLEMLÖSUNG UND KONTROLLGEWINN

Die RISC Methode stellt eine substanzielle, Architektur-bedingte Lösung der in der Einleitung angesprochenen Probleme dar:

- **Browser-/Devicekompatibilität**: der Browser wird nur auf minimale Weise genutzt - zum Zeichnen absolut postionierter Rechteckbereiche und zum Verwalten von Eingabefeldern. Diese Funktionen sind so grundlegend, dass sie von allen Browsern per se erfüllt werden. - Und selbst wenn es mal zu Problemen kommen sollte: die betreffende Nukleus-Library ist sehr klein (<1000 JavaScript Zeilen) und kann auf einfach Weise ausgetauscht/angepasst werden, ohne Auswirkungen auf die darüber liegenden funktionalen Komponenten.

  Aufwände im Bereich der permanenten Wartung von Web Benutzeroberflächen können also komplett gestrichen werden.

- **Layout-Flexibilität**: Layout Entscheidungen werden oberhalb der Browser-Execution-Ebene getroffen - innerhalb der funktionalen Komponenten. Hier sind der Kreativität keinerlei Grenzen gesetzt - sowohl seitens der Gestaltung von Benutzer-Controls als auch seitens der Funktionalität von Layout-Managern.

- **Komplexität**: die Komplexität liegt nicht mehr in einem Konglomerat aus HTML-Elementen und CSS-Styling, sondern sie wird klar auf Ebene der funktionalen Komponenten verwaltet. Dies bedeutet insbesondere: die Komplexität ist steuerbar und entspricht direkt der Komplexität der betreffenden Komponente.

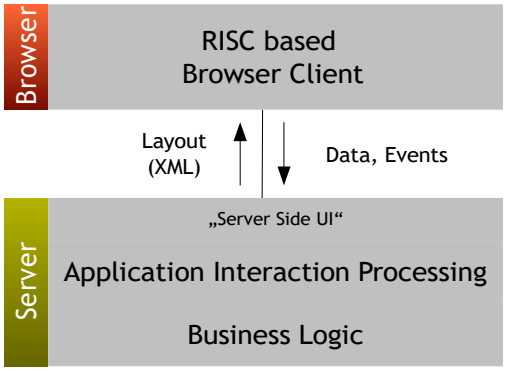
Über die RISC Methode kommt man zurück zu langfristig kontrollierbaren Szenarien. Nicht der Browser taktet die Maintenance-Zyklen und -Aufwände Ihrer Anwendung, sondern Sie gewinnen wieder die Taktungshoheit für Ihr Softwareprodukt zurück!

CAPTAINCASA ENTERPRISE CLIENT RISC

Der CaptainCasa Enterprise Client ist eine Rich Client Technologie zur Erstellung und zum Betrieb der Oberfläche langlebiger, umfangreicher, Server-basierter Geschäftsanwendungen.

SERVER CENTRIC UI

Der CaptainCasa Enterprise Client folgt dem sog. „Server-centric UI“-Prinzip.

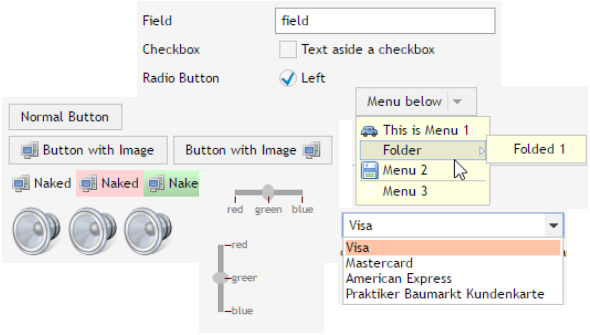


Die Entwicklung und Verarbeitung von Dialogen findet rein auf der Serverseite statt. Dadurch kann sie direkt auf die dort laufende Anwendungslogik zugreifen.

Sowohl die Kommunikation des aktuellen Layouts vom Server zum Client wie auch die Kommunikation von Daten und Ereignisse zurück ist optimiert. Es werden in beide Richtungen nur Änderungen übertragen. Roundtrips zwischen Client und Server sind deswegen ausnahmslos „lightweight“-Roundtrips mit geringem Datenvolumen.

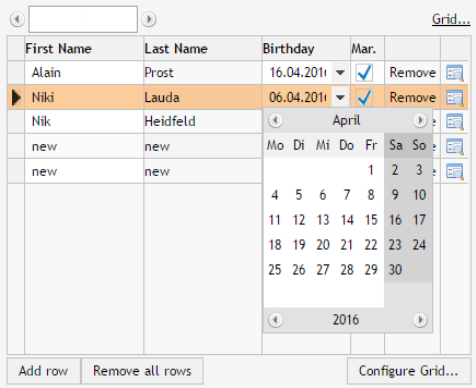
RISC BASED BROWSER CLIENT

Der Client des CaptainCasa Enterprise Client ist ein reiner HTML Client, der nach dem beschriebenen RISC Prinzip aufgebaut ist. In ihm wird eine überaus große Anzahl von grafischen Komponenten zur Verfügung gestellt.



Komponenten-Beispiele

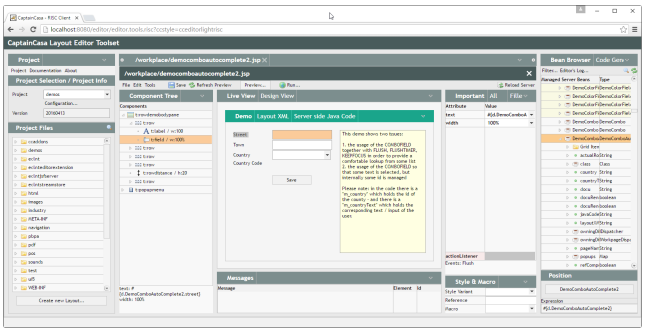
Der RISC basierte Komponentenumfang umfasst: klassische Komponenten (Button, Feld, Combo Box, …), flexible Grids (bis hin zum Pivot Grid), flexible Trees, verschiedenste Containertypen, Layout Manager, adaptive Layout Manager, Upload/Download Komponenten – und vieles mehr.



Beispiel für Grid

ENTWICKLUNG MIT CAPTAINCASA ENTEPRISE CLIENT

Die Entwicklung der Dialoge findet rein auf dem Server statt: die Dialog werden innerhalb eines komfortablen WYSIWYG-Editors entwickelt und per Java an die bestehende Logik auf dem Server angebunden.



Unsere eigene Toolumgebung ist (natürlich!) selbst mit dem CaptainCasa Enterprise Client geschrieben – und läuft deswegen komplett im Browser. Sie ist deswegen ein sehr schönes Anschauungsobjekt, wie flexibel, stabil und performant komplexe Szenarien über den RISC Ansatz im Browser ablaufen!

CAPTAINCASA COMMUNITY

CaptainCasa ist eine offene Community mittelständischen Softwarehäuser aus Deutschland, der Schweiz, Österreich der Niederlande und Belgien.

Die Community wurde 2007 gegründet – mit einem Java-basierten Client, zunächst auf Java Swing dann auf JavaFX Ebene. Nun gibt es den über die RISC Methode entwickelten Web Client. Die Umstellung von Java- auf RISC-Client erfolgt für die Anwendungen weitestgehend geräuschlos, da das Protokoll zwischen der Client- und der Server-Verarbeitung gleich geblieben ist.

Die Community ist über ein Forum miteinander verbunden und trifft sich einmal jährlich zum Community Treffen in Heidelberg.

Die CaptainCasa GmbH ist die „legal entity“ hinter der Community. Über die CaptainCasa GmbH wird der Enterprise Client vertrieben.

VERFÜGBARKEIT

CaptainCasa Enterprise Client RISC wurde innerhalb der Community seit Q2/2016 freigegeben. Eine offizielle Freigabe erfolgt zum Ende von Q2/2016.

Ach ja... - und eines noch:

WARUM DER NAME „RISC METHODE“?

Der Name „RISC Methode“ ist eine Anlehnung an die Prozessorentwicklung in den 80er und 90er Jahren. Zunächst gab es bei Prozessoren das Bestreben, möglichst viele Befehle direkt über eine entsprechende Hardware-Abbildung innerhalb des Prozessors zur Verfügung zu stellen. Man sprach von „CISC“-Prozessoren („complex instruction set“). Dies hatte negative Auswirkung auf die Komplexität des Prozessors und auf die zu erreichende Taktgeschwindigkeit. - Deswegen entwickelte sich die „RISC“-Architektur, („reduced instruction set“) in der der Prozessor nur noch grundlegendste logische Operationen ausführt – diese aber mit hoher Geschwindigkeit. Die Algorithmik z.B. einer Multiplikation wurde somit nicht mehr in der Prozessorhardware abgebildet sondern im Programm, das durch den Prozessor abgearbeitet wurde. - Heute folgen alle Prozessoren im Kern dem RISC-Ansatz.

Die Parallelen zum Browser sind natürlich offensichtlich: auch dieser entwickelte sich chronologisch von einem Text-Renderer zu einer hochkomplexen Umgebung. Die RISC Methode beschränkt diese Komplexität wieder zurück auf einfachste Grundelemente und verlagert die Algorithmik aus der Kern-Browser-Verarbeitung heraus in eine programmierte Ebene davor.

CaptainCasa GmbH
Hindemithweg 13
D- 69245 Bammental
http://www.CaptainCasa.com
info@CaptainCasa.com