1. 클로저 소개

1.1 클로저란 무엇인가?

클로저는 새 프로그래밍 언어로서, 리습을 새롭게 디자인했다. 리습은 오래된 프로그래밍 언어중에 하나로 여전히 사용되고 있다. (두번째로 오래된 언어로, 포트란이 첫번째이다.) 어째서 이런 오래된 기술과 관련있는 것을 사람들은 배우려고 하는 것일까? 리습을 이미 알고 있는 사람들에게는 당연하지만, 이런 배경지식 없이는 설명하기 어렵다. 이번 챕터는 이런 배경지식을 전달하고자 한다.

우선 언어의 탄생 동기부터 살펴볼 것이다. 그러면 왜 리습이 탄생했는지를 이해하게 될 것이다. 그러러면, 이 언어의 이상한 겉모양을 가진 문법을 설명해보자. 리습 문법의 생소함은 종종 사람들이 언어에 흥미를 잃게 만든다. 그러므로, 문법 이면의 이유를 이해하는 것이 중요하다. 일단 이해하고 난 이후에는 이어서 3가지 메인 주제를 살펴볼 것이다. 첫번째는 무엇이 리습을 특별하게 만들고, 리습의 형태로부터 클로저가 어떤 이점을 가져갔는지에 대해서이다. 두번째는 함수형 언어라는 것이 클로저에 어떤 의미인지를 설명한다. 마지막으로, 자바 가상머신에서 동작하는 클로저의 이점에 대해 토의할 것이다.

이번 챕터의 끝부분에서 클로저라는 언어를 가지고 무엇이 가능한지 이해하게 될 것이다. 이번 챕터에는 몇몇 예제 코드가 있는데 주제에서 말하고 있는 것에 대한 코드이다. 다른 언어와 비교할때 클로저 코드가 상당히 다르므로, 코드 예제를 가볍게 넘어가도 된다. 다음의 몇몇 챕터들에서 각 컨셉의 세부적 모습을 더 살펴볼이기 때문에 안심해도 된다. 이번 챕터의 목적은 책의 다른 부분에 앞서 미리 무엇이 있는지 당신에게 주지시켜 주는 것이다. 새로우면서도 오래된 놀라운 언어로 흠뻑 빠져보자.

1.1.1 클로저 – 리습의 환생

누군가 리습이 세계에서 가장 파워풀한 언어라고 말할 때 많은 사람들이 동의합니다. (심지어 그렇게 말하는 사람들은 우쭐하기 좋아하는 리습 광신도라고 여긴다) 어떤 다른 언어가 이와 유사한 주장을 하고, 그러고도 고개를 끄덕일 수 있을까? C++? 자바? 루비?

많은 사람들은 리습을 죽은 언어이고 더 이상 사용하지 않는 언어라고 생각합니다. 그러면서도, 사람들은 몇몇 최첨단 소프트웨어 시스템이 다양한 도메인에서 사용되고 있다는 소리를 듣습니다. 나사의 패스파인더 미션-계획 소프트웨어, 헤지 펀드의 트레이드 알고리즘, 항공 예약, 데이터 마이닝, 자연어 처리, 전문가 시스템, 바이오인포메틱스, 로보틱스, 전자 디자인 자동화 등등.

리습은 어둠의 기술이라는 명성을 가진다. 실제로, 몇몇 성공적인 스타트업에서 비밀무기로서 언급되어져 왔습니다. 이런 모든 것은 그럴만한 좋은 이유가 있습니다. 이 챕터에서 이 미스터리 함들을 클로저라고 불리우는 새로운 리습을 가지고 설명해보겠습니다. 이 새로운 언어는 실무적인 리습일뿐 아니라 실용적인 패러다임을 포함하여 효율성을 추가하였습니다. 실용적인 패러다임은 동시성의 의미를 새로 새우고, 자바 가상 머신 위에서 돌아간다는 점입니다.

끝으로 클로저가 도전적인 집합 문제를 해결하기 위한 넓은 집합 도메인에 사용된다는 것을 듣고 놀라실 겁니다. 매우 큰 범위의 텍스트 보관, 데이터 마이닝 시스템, 웹을 심도 있게 이해하기 위한 시멘틱 웹 구현 기술, 패키지 모델링과 통계, 인공지능 가격 최적화, 항공 지연 예측, 보험에서 날씨 기반 사건 예측, 로보틱스, 가상 세계 구동 등등.

이제 우리는 클로저 언어의 각각의 장점들을 깊게 탐험할 것입니다.

1.1.2 어떻게 도착할 것인가

리습은 프로그래밍 언어로서, 1958년 존 매카시에 의해 디자인되었습니다. 존 매카시는 튜링상을 탄 수학자 입니다. 인공지능의 분야를 위해 처음 디자인 되었으며, 특히 (숫자 대신에) 심볼릭 표현을 처리하기 위해서 였고, 이런 심볼의 리스트로 표현되었습니다. 리습은 또한 함수 추상화를 통해 복잡성을 다루는 용도로 디자인 되었습니다. 이 말은, 함수가 1급 객체이며, 값들처럼 던져지고 서로 구성될 수 있습니다. 이 점이 자바, 루비와 다른 점입니다. 마지막으로, 리습은 “코드 is 데이터”라는 이데올로기를 가집니다. 즉 코드는 쉽게 데이터로서 다뤄질 수 있습니다. 앞으로 보게 되겠지만, 이러한 특징들의 조합으로 인해 놀라운 수준의 표현이 가능합니다.

그러나 몇 년동안 상황이 리습에 대한 음모를 품었습니다. 70~80 년대 인공지능 연구에 많은 돈이 투자되었지만, 결국 기대했던 것을 가져오지 못했습니다. 리습은 인공지능 분야와 관련성을 가졌기 때문에 인공지능 붐이 끝났을때, 그 언어의 인기도 시들해졌습니다. 많은 사람들이 리습의 인공지능 분야에서 실패를 탓했습니다. 이런 오명은 쉽게 지워지기 힘들었습니다.

많은 리습들은 이와 같은 초기 시대에 태어났씁니다. 그리고 많은 리습들은 잊혀졌습니다. 몇 개가 오늘까지 사용되고 있습니다. 커먼리습, CMUCL, 알레그로 CL. 어떤 컴퓨터 대학교는 스킴 언어를 상당히 적합한 교육 언어로 사용하고 있습니다.

자바 가상머신 위에 리습을 올리려는 시도가 몇번 있었습니다. j스킴, 카와, ABCL, 등등.

다양한 이유로 이들은 결코 유명해지지 못했습니다. 클로저는 자바가상머신 위에 리습을 환생시키려는 마지막 시도 입니다. 리치 히키가 만들었습니다. 마침내, 클로저는 살아남은 리습이 될 수도 있습니다. 실제로 리습의 미래와 동적 언어의 미래가 될 수도 있습니다.

이 섹션을 클로저를 가르치키 위한 책이라는 접근법으로 감싸서 설명할 것입니다. 그리고나서 리습 유형의 언어를 배우는 대부분의 사람들이 처음으로 뛰어드는 부분으로 들어갈 것입니다.

바로 – 문법 syntax 입니다.

1.1.3 이 책은 어떤 방법으로 클로저를 설명하는가

이책의 철학은 두개의 큰 기둥 위에 높여 있습니다. 클로저의 첫번째 원리를 강조하고, 이런 원리 이해를 위해 실습하는 단계입니다. 이런 컨셉을 설명하기 위한 많은 예제 코드들을 보게될 것입니다. 클로저 언어는 생각하는 새로운 방법을 요구하며, 이 것은 이미 익숙한 방법과 다를 것입니다. 우린 이런 점을 리습의 기초, 함수형 프로그래밍, 클로저에서 제공하는 많은 기능들을 통해 설명할 것입니다.

이책을 훑어보면 오늘날 발전중인 다양한 시스템에 적용 가능한 많은 예제 코드들을 담아놓았다는 것을 볼 수 있습니다. 일단 언어 자체의 특징으로 거슬러 시작해서 TTD, 데이터 저장, 웹서비스 등등 현실세계의 문제를 설명할 것입니다. 메시지 시스템에 사용을 통해 클로저 어플리케이션의 스케일을 살펴볼 것이며, 심지어 작은 분산 컴퓨팅 프레임워크를 만들어 볼 것입니다. 하둡, HBase 같은 기술을 통해 빅 데이터를 처리하는데 클로저를 사용하는 것을 보여줄 것입니다. 또한 클로저에서 도메인-특화 언어를 만드는 것을 볼 것입니다.

이런 점을 염두해두면, 클로저를 탐험할 준비가 끝난 것입니다. 첫번째 단계는 코드를 처음본 사람들이 자주 묻는 질문입니다. 클로저 코드는 왜 그런 모양인가요?

1.2 클로저 문법 이해하기

대개 새로운 언어를 배우는 대부분의 사람들은 첫번째로 문법을 생각합니다. 문법은 언어들을 다르게 보이게 만드는 것 중 하나입니다. 실제로, 그 모양이 사람들로 하여금 어떤 언어를 좋아하기도 싫어하기도 하게 만드는 이유가 됩니다.

그러나, 문법은 껍데기에 지나지 않습니다. 세부적인 문법, 언어가 프로그래머에게 각 생성자는 어떤 모양인지 강요하는 규칙등은 중요한 것이 아닙니다. 컴파일러는 abstract syntax tree 라고 불리는 데이터 구조를 생성하는데 언어의 상세한 문법대로 쓰여진 코드들을 파싱해서 만듭니다. 소스코드는 AST 가 만들어진 이후에는 버려집니다. 시멘틱 분석과 같은 모든 다른 번역 작업들은 단지 AST 를 필요로 할 뿐입니다. 그러므로 세부적인 문법은 기본적으로 사람들을 위한 것이라고 말할 수 있습니다. 사람의 편의성이 중요치 않다고 말하는 것이 아니라, 문법으로는 컴퓨터 언어가 무엇이 가능한지를 얻을 수 없다는 것입니다. (1.3.1 섹션에서 이 주제를 다시 다룹니다)

클로저는 배우기에 매우 단순한 언어입니다. 문법이라는 측면에서 거의 배울 것이 없습니다. 다음의 클로저 코드를 살펴보세요.

(add 3 5 7)

이 표현식은 add 라는 함수를 호출하고 3개의 숫자를 인자로 넘기고 있습니다. 함수는 항상 이러한 방식으로 불려집니다. 함수 이름이 처음에 나타나고 이어서 나오는 것은 인자입니다. 클로저 문법의 일반적 규칙들을 살펴봅시다.

클로저 표현식은 심볼이거나 시퀀스 일 수 있습니다. 예를들면, 앞서 본 표현식은 4개의 심볼을 가진 리스트로 구성되었습니다. (add, 3, 5, 7) . 13과 + 와 같은 하나의 심볼을 가진 표현식도 가능합니다.

만약 표현식이 시퀀스라면, 비었을 수도 있고, 다른 심볼이나 시퀀스를 포함할 수도 있습니다. 심볼이란 프로그램 텍스트에 나타나는 것일 뿐입니다. 지금까지 배운 것이 전부입니다!! 이 규칙의 재귀적 속성을 주의 하세요. 클로저 리스트는 다른 리스트를 가질 수 있다는 점 때문에 (필연적으로 중첩되는) 복잡한 표현식도 가능합니다. 더욱이,