Machine Learning, Blockchain and Eclipse Scout
Eclipse Scout
What is Eclipse Scout?

Business Application Framework

- Based on Java and HTML5
- Multi Device support, Modular Apps, ...

Framework Goals

- Long Term Strategy (enterprise apps live > 10 years)
- Boosts Productivity (producing software in Switzerland ...)
- Easy to learn (new team members productive in 1-2 weeks)
Eclipse Scout Application Model

Java Application Model

- Clean Business Code
- High Maintainability

```java
@Order(10)
public class FirstNameField extends AbstractStringField {
    @Override
    protected String getConfigurerLabel() {
        return TEXTS.get("FirstName");
    }
}
```
Eclipse Scout HTML5 Rendering

Current Standards

- HTML5, CSS3, JavaScript
- Flexible Styling
- Theming
Eclipse Scout Commercial Application
Eclipse Scout Hello World
Machine Learning
Deeplearning4j demo
What’s behind the Hype?

Machine learning (ML) actually starts to work ...
Many robust ML libraries / frameworks to choose from

**Innovation driven by Open Source**

- People able to replicate results
- Open Source software
- Open Data Collections
- Open Publications (arXiv)
<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td>Personality judgement via social media 2014, conversational speech recognition 2016.</td>
</tr>
</tbody>
</table>
Deeplearning4j

Deep Learning Library

➔ Java (most are Python)
➔ Good documentation
➔ Open Source

Features

➔ Full GPU support
➔ Distributed deep learning
➔ Runs with Hadoop + Spark

https://github.com/deeplearning4j/deeplearning4j
The ML «Hello World» Recognition of handwritten digits

Proc. of the IEEE, November 1998

Fig. 2. Architecture of LeNet-5, a Convolutional Network whose weights are constrained to be identical.

1998 Gradient-based Learning for Document Recognition, Y. LeCun

```java
public static MultilayerConfiguration configuration() {
    return new NeuralNetConfiguration.Builder()
        .seed(SEED).weightInit(WeightInit.XAVER)
        .iterations(NUM_ITERATIONS)
        .regularization(true).l2(0.0005).learningRate(.01)
        .optimizationAlgorithm(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
        .updater(Updater.NESTEROVS).momentum(0.9)
        .list()
        .layer(0, new ConvolutionLayer.Builder(5, 5)
            .stride(1, 1)
            .nIn(NUM_CHANNELS)
            .nOut(26)
            .activation(Activation.IDENTITY)
            .build())
        .layer(1, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
            .kernelSize(2, 2)
            .stride(2, 2)
            .build())
        .layer(2, new ConvolutionLayer.Builder(5, 5).stride(1, 1)
            .nOut(50)
            .activation(Activation.IDENTITY)
            .build())
        .layer(3, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
            .kernelSize(2, 2)
            .stride(2, 2)
            .build())
        .layer(4, new DenseLayer.Builder()
            .activation(Activation.RELU)
            .nOut(500)
            .build())
        .layer(5, new OutputLayer.Builder(lossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
            .activation(Activation.SOFTMAX)
            .nOut(NUM_OUTPUTS)
            .build())
        .setInputType(InputType.convolutionalFlat(28, 28, 1))
        .backprop(true)
        .pretrain(false).build();
```
Blockchain
web3j demo with Ethereum
Blockchain «Micro-Intro»

**Blockchain**
- **Bitcoin** here since 2009
- Cheap, fast, reliable, efficient (compare with T+3 and high fees)
- **Ethereum** adds smart contracts
- Increasing amounts of money ...

**Main Challenges**
- Scalability
- Privacy
Smart Contracts

What is a Ethereum Smart Contract?

- Piece of (byte) code
- Is executed by the Ethereum Virtual Machine (EVM)
- Has an owner
- Has a life cycle

Example: «Truly» autonomous cars
«Truly» Autonomous Cars

Smart contract: To order car to transport people (by paying to contract)
Smart contract: Car pays for energy/services
Ethereum and Application Integration

Ethereum Client
Geth/TestRPC/...

Ethereum Peer-to-Peer Network

JavaScript
web3

Java
web3j

Interface
HTTP://localhost:8545
JSON-RPC
Web3j

Library to interact with Ethereum (its peer-to-peer clients)
- Java
- Good documentation
- Open Source

Features
- JSON-RPC client API implementation
- Command line tools to generate Java contract wrappers

https://github.com/web3j/web3j
Smart Contracts Life Cycle

Deploying and using Smart Contracts
1. Write contract in high level language (eg. Solidity)
2. Compile contract to EVM byte-code
3. Pack byte code into a contract creation TX and sent to the network
4. The TX gets its own contract address
5. Invoke contract methods
contract greeter {

    /* Owner of this contract */
    address owner;

    /* Configurable greeting */
    string greeting;

    /* Constructor runs when contract is deployed */
    function greeter(string _greeting) public {
        owner = msg.sender;
        greeting = _greeting;
    }

    /* Main function */
    function greet() constant returns (string) {
        return greeting;
    }

    /* Function to recover the funds on the contract */
    function kill() {
        if (msg.sender == owner)
            selfdestruct(owner);
    }
}
Solidity Compiler (online)

byte code (EVM) to deploy contract

Deploy script (JS)
web3j: greeter.sol → Greeter.java

From Solidity to Java Contract Class

1. Compile *greeter.sol* (e.g. using online compiler)
   → *greeter.bin, greeter.abi*

2. Create contract wrapper class (use Web3j command line tool)
   → *Greeter.java*

1. Use *Greeter.java* in your Java code
Generated Contract Wrapper

```java
/**
 * <p>Auto generated code.<br>
 * <strong>Do not modify!</strong><br>
 * Please use [link org.web3j.codegen.SolidityFunctionWrapperGenerator] to update.<br>
 * </p>
 * <p>Generated with web3j version 2.1.0.</p>
 */

public final class Greeter extends Contract {
    private static final String BINARY = "0x6006004052346100005760040516102e3383390101640528051015b60008054600160a060020a0319166c
    
    private Greeter(String contractAddress, Web3j web3j, Credentials credentials, BigInteger gasPrice, BigInteger gasLimit) {
        super(contractAddress, web3j, credentials, gasPrice, gasLimit);
    }

    private Greeter(String contractAddress, Web3j web3j, TransactionManager transactionManager, BigInteger gasPrice, BigInteger gasLimit) {
        super(contractAddress, web3j, transactionManager, gasPrice, gasLimit);
    }

    public Future<Uint256> deposits() {
        Function function = new Function("deposits",
        Arrays.<Type>asList(),
        Arrays.<TypeReference<?>>asList(new TypeReference<Uint256>() {}));
        return execute(CallSingleValueReturnAsync(function);
    }
```
Ethereum, web3j + Eclipse Scout

Trading Network Demo
- **Currency Hedging**: Buy orders and Sell orders (€ / US$)

- **Classical Business App**
  - **Identity Management** for mapping real persons ↔ Blockchain addresses
  - **User Interface**

- **Blockchain Benefits**
  - **Efficiency**: No central organization/infrastructure
  - **Trust**: Tampering-proof ledger, trust by blockchain technology
Eclipse Scout
UI (web application)

Eclipse Scout
Backend

web3

Ethereum Client
TestRPC

web3j

JDBC

PostgreSQL
### Trading Center

#### NESSELE

<table>
<thead>
<tr>
<th>Deal-Nr</th>
<th>Action</th>
<th>Organization</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Buy</td>
<td>Nestlé</td>
<td>500,000</td>
</tr>
<tr>
<td>2</td>
<td>Buy</td>
<td>Nestlé</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

```javascript
var USDEUR = OrderBook.at('0xbfc371bedaa57b0f73a596a8f6e6f84a01c6441')
undefined

USDEUR.symbol()
null

USDEUR.getNumberofBuyOrders()
0

USDEUR.getNumberofSellOrders()
0

USDEUR.matchExists()
true

USDEUR.topBuyOrder
{"String: '1'} s: 1, e: 0, c: [ 1 ]

USDEUR.topSellOrder
{"String: '3'} s: 1, e: 0, c: [ 3 ]

USDEUR.matchExists()
true

USDEUR.topBuyOrder
{"String: '2'} s: 1, e: 0, c: [ 2 ]

USDEUR.matchExists()
false
```
Thanks!

@ZimMatthias