Supply Chain

adalah Smart Contract “manajemen”, dimana kita dapat menambahkan item.

```
pragma solidity ^0.6.0;
contract ItemManager{
    enum SupplyChainSteps{Created, Paid, Delivered}

    struct S_Item {
        ItemManager.SupplyChainSteps _step;
        string _identifier;
        uint _priceInWei;
    }
    mapping(uint => S_Item) public items;
    uint index;

    event SupplyChainStep(uint _itemIndex, uint _step);

    function createItem(string memory _identifier, uint _priceInWei) public {

        items[index]._priceInWei = _priceInWei;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step));
        index++;
    }

    function triggerPayment(uint _index) public payable {
        require(items[_index]._priceInWei <= msg.value, "Not fully paid");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is
further in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step));
    }

    function triggerDelivery(uint _index) public {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further
in the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step));
    }
}
```

Dengan kode diatas dimungkinkan untuk menambahkan item dan membayarnya, memindahkannya ke depan dalam supply chain dan memicu pengiriman. Tapi hanya memberikan alamat sederhana kepada pengguna untuk mengirim uang.

Item Smart Contract

Coba tambahkan Smart Contract yang lain :

```
pragma solidity ^0.6.0;

import "./ItemManager.sol";

contract Item {
    uint public priceInWei;
    uint public paidWei;
    uint public index;

    ItemManager parentContract;

    constructor(ItemManager _parentContract, uint _priceInWei, uint _index)
public {
        priceInWei = _priceInWei;
        index = _index;
        parentContract = _parentContract;
    }

    receive() external payable {
        require(msg.value == priceInWei, "We don't support partial payments");
        require(paidWei == 0, "Item is already paid!");
        paidWei += msg.value;
        (bool success, ) =
address(parentContract).call{value:msg.value}(abi.encodeWithSignature("trigger
Payment(uint256)", index));
        require(success, "Delivery did not work");
    }

    fallback () external {

    }
}
```

Perubahan Solidity

Note bahwa *call.value(msg.value)(abi.encodeWithSignature("triggerPayment(uint256)", index))* , karena perubahan pada Solidity versi 6.4 sangat direkomendasikan harus diubah menjadi *call{value:msg.value} (abi.encodeWithSignature("triggerPayment(uint256)", index))* .

Dan ubah Smart Contract Item Manager untuk menggunakan Smart Contract Item meskipun hanya Struct:

```
pragma solidity ^0.6.0;

import "./Item.sol";
```

```
contract ItemManager {
    struct S_Item {
        Item _item;
        ItemManager.SupplyChainSteps _step;
        string _identifier;
    }

    mapping(uint => S_Item) public items;
    uint index;

    enum SupplyChainSteps {Created, Paid, Delivered}

    event SupplyChainStep(uint _itemIndex, uint _step, address _address);

    function createItem(string memory _identifier, uint _priceInWei) public {
        Item item = new Item(this, _priceInWei, index);
        items[index]._item = item;
        items[index]._step = SupplyChainSteps.Created;
        items[index]._identifier = _identifier;
        emit SupplyChainStep(index, uint(items[index]._step), address(item));
        index++;
    }

    function triggerPayment(uint _index) public payable {
        Item item = items[_index]._item;
        require(address(item) == msg.sender, "Only items are allowed to update
themselves");
        require(item.priceInWei() == msg.value, "Not fully paid yet");
        require(items[_index]._step == SupplyChainSteps.Created, "Item is
further in the supply chain");
        items[_index]._step = SupplyChainSteps.Paid;
        emit SupplyChainStep(_index, uint(items[_index]._step),
address(item));
    }

    function triggerDelivery(uint _index) public {
        require(items[_index]._step == SupplyChainSteps.Paid, "Item is further
in the supply chain");
        items[_index]._step = SupplyChainSteps.Delivered;
        emit SupplyChainStep(_index, uint(items[_index]._step),
address(items[_index]._item));
    }
}
```

Dengan ini sekarang kita hanya perlu memberi pelanggan Smart Contract cerdas yang dibuat selama “creatItem” dan dia akan dapat membayar langsung dengan mengirimkan X Wei ke Smart Contract. Tapi Smart Contract belum terlalu aman. Kita butuh semacam fungsi owner.

Ownable Functionality

Biasanya kami akan menambahkan Smart Contract OpenZeppelin dengan Fungsionalitas yang dapat Dimiliki. Tetapi pada saat menulis ini dokumen mereka belum diperbarui ke solidity 0.6. Jadi, alih – alih kami akan menambahkan fungsi Ownable kami sendiri sangat mirip dengan satu dari OpenZeppelin:

Ownable.sol

```
pragma solidity ^0.6.0;

contract Ownable {
    address public _owner;

    constructor () internal {
        _owner = msg.sender;
    }

    /**
    * @dev Throws if called by any account other than the owner.
    */

    modifier onlyOwner() {
        require(isOwner(), "Ownable: caller is not the owner");
        _;
    }

    /**
    * @dev Returns true if the caller is the current owner.
    */
    function isOwner() public view returns (bool) {
        return (msg.sender == _owner);
    }
}
```

Kemudian ubah ItemManager sehingga semua fungsi, yang seharusnya dapat dieksekusi oleh "pemilik saja" memiliki pengubah yang benar.

```
pragma solidity ^0.6.0;

import "./Ownable.sol";

import "./Item.sol";
contract ItemManager is Ownable {
        //…

        function createItem(string memory _identifier, uint _priceInWei)
public onlyOwner {
    //…
        }
```

```
        function triggerPayment(uint _index) public payable {
            //…
        }
        function triggerDelivery(uint _index) public onlyOwner {
            //…
}
```

Install Truffle

Untuk menginstall Truffle, Buka terminal (powershell) di komputer.

Ketik kata berikut :

npm install -g truffle