

Substrate上的 智能合约





Maggie Dong

Software developer @ParityTech

Background:

- NFT protocol
- Dev Advocate
- Ethereum DApp Development
- Substrate Development

什么是智能合约

部署并运行在区块链之上的、“无法”更改逻辑的程序

Substrate上的contract模块

Contract模块为Substrate
runtime提供了 **部署** 和 **运行**
WebAssembly合约的方法

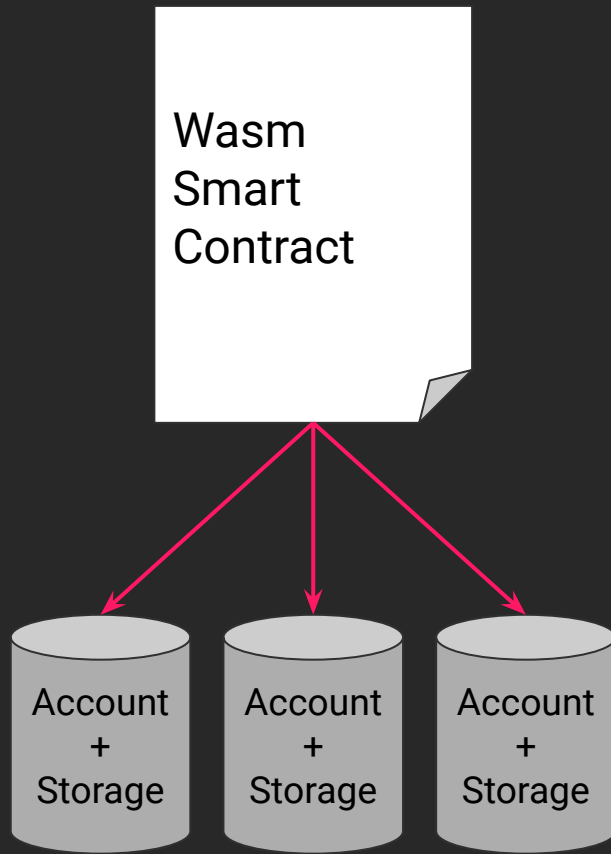
- 基于账户模型
 - 余额
 - 合约代码
 - 存储
- Gas Fees
- 可回滚的交易
- Storage Rent
- Substrate 类型

合约部署

两步

1. 把Code放到链上 **put_code**
2. Code 实例化 **instantiate**

可以减少不必要的冗余代码



调用合约

当调用合约时，会去获取相关代码并且执行

调用合约可以：

- 修改合约账户下的存储
- 创建新的合约
- 调用其他合约
- 调用runtime方法

隔离的执行环境

当调用合约时, 会创建一个**新的执行环境**:

- 创建一个wasm实例
- 它有短暂/独立的内存
- 执行环境是被隔离的
- 如果失败了可以回滚状态
- 当全部执行成功后, 才去修改substrate的存储

Parity ink!

用Rust的方式来构建智能合约

Ink! 的组成



nk!的结构

Language

真正的eDSL层, 为用户提供写代码的语法/方式

Model

中间层

Core

和contract模块交互的核心功能层

ink!如何和runtime交互

ink!



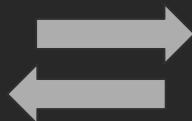
`ext_address`

`ext_caller`

`ext_call`

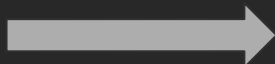
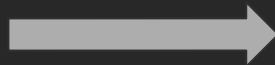
`ext_create`

`...`



ink! metadata

ink!



Polkadot JS UI



Development
version 191
#461

et. al

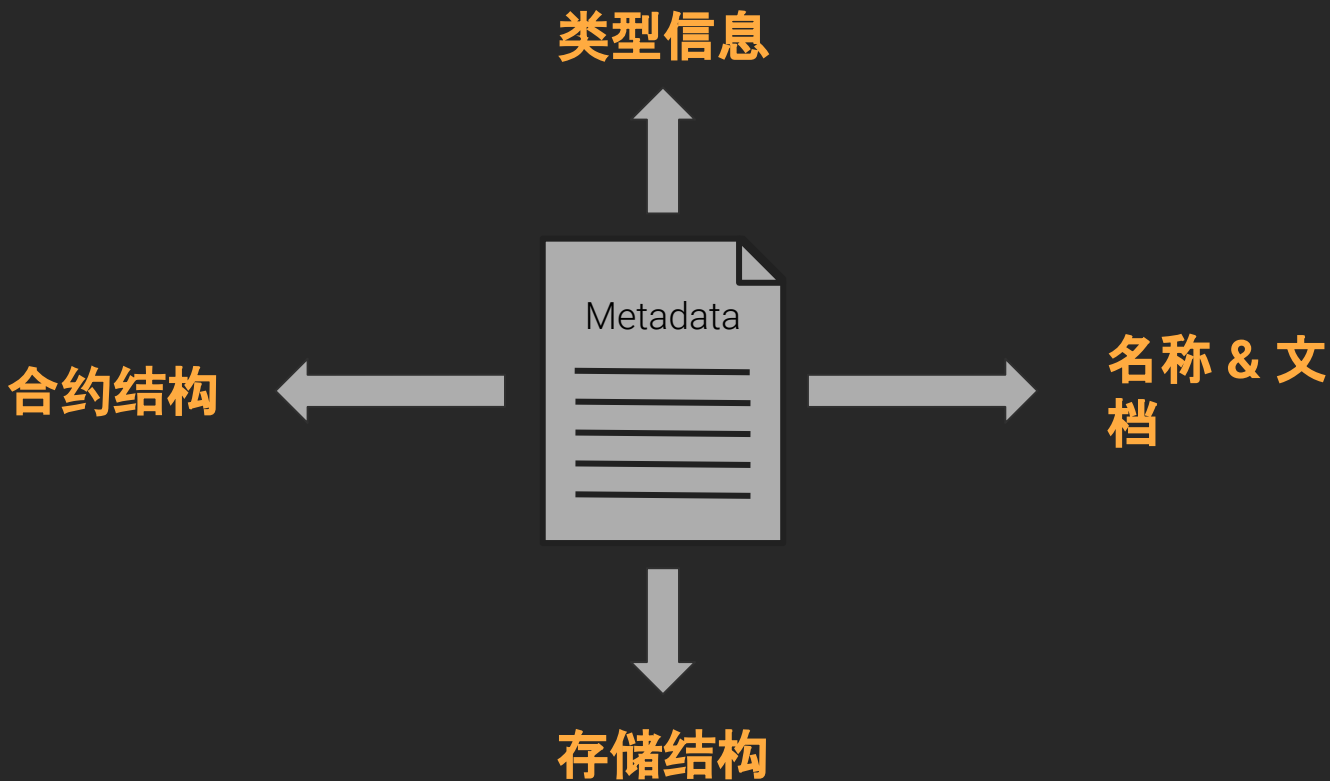
ink! metadata

```
1 {
2   "registry": {
3     "strings": {
4       "StorageAndEnv",
5       "erc20",
6       "ink_private",
7       "ink_storage",
8       "storage",
9       "Storage",
10      "total_supply",
11      "value",
12      "ink_core",
13      "storage",
14      "value",
15      "cell",
16      "sync_cell",
17      "key",
18      "key",
19      "balances",
20      "HashMap",
21      "collections",
22    }
  }
}

765 {
766   "docs": []
767 },
768 {
769   "name": 56,
770   "selector": "[\\0*26\\", "\\0*88\\", "\\0*10\\", "\\0*64\\"]",
771   "mutates": false,
772   "args": [
773     {
774       "name": 57,
775       "type": {
776         "ty": 10,
777         "display_name": [
778           22
779         ]
780       }
781     }
782   ],
783   "return_type": {
784     "ty": 4,
785     "display_name": [
786       22
787     ]
788   }
789 }

88 {
89   "custom.namespace": {
90     2,
91     2,
92     3,
93     4,
94     5,
95     6,
96     7,
97     8,
98     9,
99     10,
100    11,
101    12,
102    13,
103    14,
104    15,
105    16,
106    17,
107    18,
108    19,
109    20,
110    21,
111    22,
112    23,
113    24,
114    25,
115    26,
116    27,
117    28,
118    29,
119    30,
120    31,
121    32,
122    33,
123    34,
124    35,
125    36,
126    37,
127    38,
128    39,
129    40,
130    41,
131    42,
132    43,
133    44,
134    45,
135    46,
136    47,
137    48,
138    49,
139    50,
140    51,
141    52,
142    53,
143    54,
144    55,
145    56,
146    57,
147    58,
148    59,
149    60,
150    61,
151    62,
152    63,
153    64,
154    65,
155    66,
156    67,
157    68,
158    69,
159    70,
160    71,
161    72,
162    73,
163    74,
164    75,
165    76,
166    77,
167    78,
168    79,
169    80,
170    81,
171    82,
172    83,
173    84,
174    85,
175    86,
176    87,
177    88,
178    89,
179    90,
180    91,
181    92,
182    93,
183    94,
184    95,
185    96,
186    97,
187    98,
188    99,
189    200,
190    201,
191    202,
192    203,
193    204,
194    205,
195    206,
196    207,
197    208,
198    209,
199    210,
200    211,
201    212,
202    213,
203    214,
204    215,
205    216,
206    217,
207    218,
208    219,
209    220,
210    221,
211    222,
212    223,
213    224,
214    225,
215    226,
216    227,
217    228,
218    229,
219    230,
220    231,
221    232,
222    233,
223    234,
224    235,
225    236,
226    237,
227    238,
228    239,
229    240,
230    241,
231    242,
232    243,
233    244,
234    245,
235    246,
236    247,
237    248,
238    249,
239    250,
240    251,
241    252,
242    253,
243    254,
244    255,
245    256,
246    257,
247    258,
248    259,
249    260,
250    261,
251    262,
252    263,
253    264,
254    265,
255    266,
256    267,
257    268,
258    269,
259    270,
260    271,
261    272,
262    273,
263    274,
264    275,
265    276,
266    277,
267    278,
268    279,
269    280,
270    281,
271    282,
272    283,
273    284,
274    285,
275    286,
276    287,
277    288,
278    289,
279    290,
280    291,
281    292,
282    293,
283    294,
284    295,
285    296,
286    297,
287    298,
288    299,
289    300,
290    301,
291    302,
292    303,
293    304,
294    305,
295    306,
296    307,
297    308,
298    309,
299    310,
300    311,
301    312,
302    313,
303    314,
304    315,
305    316,
306    317,
307    318,
308    319,
309    320,
310    321,
311    322,
312    323,
313    324,
314    325,
315    326,
316    327,
317    328,
318    329,
319    330,
320    331,
321    332,
322    333,
323    334,
324    335,
325    336,
326    337,
327    338,
328    339,
329    340,
330    341,
331    342,
332    343,
333    344,
334    345,
335    346,
336    347,
337    348,
338    349,
339    350,
340    351,
341    352,
342    353,
343    354,
344    355,
345    356,
346    357,
347    358,
348    359,
349    360,
350    361,
351    362,
352    363,
353    364,
354    365,
355    366,
356    367,
357    368,
358    369,
359    370,
360    371,
361    372,
362    373,
363    374,
364    375,
365    376,
366    377,
367    378,
368    379,
369    380,
370    381,
371    382,
372    383,
373    384,
374    385,
375    386,
376    387,
377    388,
378    389,
379    390,
380    391,
381    392,
382    393,
383    394,
384    395,
385    396,
386    397,
387    398,
388    399,
389    400,
390    401,
391    402,
392    403,
393    404,
394    405,
395    406,
396    407,
397    408,
398    409,
399    410,
400    411,
401    412,
402    413,
403    414,
404    415,
405    416,
406    417,
407    418,
408    419,
409    420,
410    421,
411    422,
412    423,
413    424,
414    425,
415    426,
416    427,
417    428,
418    429,
419    430,
420    431,
421    432,
422    433,
423    434,
424    435,
425    436,
426    437,
427    438,
428    439,
429    440,
430    441,
431    442,
432    443,
433    444,
434    445,
435    446,
436    447,
437    448,
438    449,
439    450,
440    451,
441    452,
442    453,
443    454,
444    455,
445    456,
446    457,
447    458,
448    459,
449    460,
450    461,
451    462,
452    463,
453    464,
454    465,
455    466,
456    467,
457    468,
458    469,
459    470,
460    471,
461    472,
462    473,
463    474,
464    475,
465    476,
466    477,
467    478,
468    479,
469    480,
470    481,
471    482,
472    483,
473    484,
474    485,
475    486,
476    487,
477    488,
478    489,
479    490,
480    491,
481    492,
482    493,
483    494,
484    495,
485    496,
486    497,
487    498,
488    499,
489    500,
490    501,
491    502,
492    503,
493    504,
494    505,
495    506,
496    507,
497    508,
498    509,
499    510,
500    511,
501    512,
502    513,
503    514,
504    515,
505    516,
506    517,
507    518,
508    519,
509    520,
510    521,
511    522,
512    523,
513    524,
514    525,
515    526,
516    527,
517    528,
518    529,
519    530,
520    531,
521    532,
522    533,
523    534,
524    535,
525    536,
526    537,
527    538,
528    539,
529    540,
530    541,
531    542,
532    543,
533    544,
534    545,
535    546,
536    547,
537    548,
538    549,
539    550,
540    551,
541    552,
542    553,
543    554,
544    555,
545    556,
546    557,
547    558,
548    559,
549    560,
550    561,
551    562,
552    563,
553    564,
554    565,
555    566,
556    567,
557    568,
558    569,
559    570,
560    571,
561    572,
562    573,
563    574,
564    575,
565    576,
566    577,
567    578,
568    579,
569    580,
570    581,
571    582,
572    583,
573    584,
574    585,
575    586,
576    587,
577    588,
578    589,
579    590,
580    591,
581    592,
582    593,
583    594,
584    595,
585    596,
586    597,
587    598,
588    599,
589    600,
590    601,
591    602,
592    603,
593    604,
594    605,
595    606,
596    607,
597    608,
598    609,
599    610,
600    611,
601    612,
602    613,
603    614,
604    615,
605    616,
606    617,
607    618,
608    619,
609    620,
610    621,
611    622,
612    623,
613    624,
614    625,
615    626,
616    627,
617    628,
618    629,
619    630,
620    631,
621    632,
622    633,
623    634,
624    635,
625    636,
626    637,
627    638,
628    639,
629    640,
630    641,
631    642,
632    643,
633    644,
634    645,
635    646,
636    647,
637    648,
638    649,
639    650,
640    651,
641    652,
642    653,
643    654,
644    655,
645    656,
646    657,
647    658,
648    659,
649    660,
650    661,
651    662,
652    663,
653    664,
654    665,
655    666,
656    667,
657    668,
658    669,
659    670,
660    671,
661    672,
662    673,
663    674,
664    675,
665    676,
666    677,
667    678,
668    679,
669    680,
670    681,
671    682,
672    683,
673    684,
674    685,
675    686,
676    687,
677    688,
678    689,
679    690,
680    691,
681    692,
682    693,
683    694,
684    695,
685    696,
686    697,
687    698,
688    699,
689    700,
690    701,
691    702,
692    703,
693    704,
694    705,
695    706,
696    707,
697    708,
698    709,
699    710,
700    711,
701    712,
702    713,
703    714,
704    715,
705    716,
706    717,
707    718,
708    719,
709    720,
710    721,
711    722,
712    723,
713    724,
714    725,
715    726,
716    727,
717    728,
718    729,
719    730,
720    731,
721    732,
722    733,
723    734,
724    735,
725    736,
726    737,
727    738,
728    739,
729    740,
730    741,
731    742,
732    743,
733    744,
734    745,
735    746,
736    747,
737    748,
738    749,
739    750,
740    751,
741    752,
742    753,
743    754,
744    755,
745    756,
746    757,
747    758,
748    759,
749    760,
750    761,
751    762,
752    763,
753    764,
754    765,
755    766,
756    767,
757    768,
758    769,
759    770,
760    771,
761    772,
762    773,
763    774,
764    775,
765    776,
766    777,
767    778,
768    779,
769    780,
770    781,
771    782,
772    783,
773    784,
774    785,
775    786,
776    787,
777    788,
778    789,
779    790,
780    791,
781    792,
782    793,
783    794,
784    795,
785    796,
786    797,
787    798,
788    799,
789    800,
800    801,
801    802,
802    803,
803    804,
804    805,
805    806,
806    807,
807    808,
808    809,
809    810,
810    811,
811    812,
812    813,
813    814,
814    815,
815    816,
816    817,
817    818,
818    819,
819    820,
820    821,
821    822,
822    823,
823    824,
824    825,
825    826,
826    827,
827    828,
828    829,
829    830,
830    831,
831    832,
832    833,
833    834,
834    835,
835    836,
836    837,
837    838,
838    839,
839    840,
840    841,
841    842,
842    843,
843    844,
844    845,
845    846,
846    847,
847    848,
848    849,
849    850,
850    851,
851    852,
852    853,
853    854,
854    855,
855    856,
856    857,
857    858,
858    859,
859    860,
860    861,
861    862,
862    863,
863    864,
864    865,
865    866,
866    867,
867    868,
868    869,
869    870,
870    871,
871    872,
872    873,
873    874,
874    875,
875    876,
876    877,
877    878,
878    879,
879    880,
880    881,
881    882,
882    883,
883    884,
884    885,
885    886,
886    887,
887    888,
888    889,
889    890,
890    891,
891    892,
892    893,
893    894,
894    895,
895    896,
896    897,
897    898,
898    899,
899    900,
900    901,
901    902,
902    903,
903    904,
904    905,
905    906,
906    907,
907    908,
908    909,
909    910,
910    911,
911    912,
912    913,
913    914,
914    915,
915    916,
916    917,
917    918,
918    919,
919    920,
920    921,
921    922,
922    923,
923    924,
924    925,
925    926,
926    927,
927    928,
928    929,
929    930,
930    931,
931    932,
932    933,
933    934,
934    935,
935    936,
936    937,
937    938,
938    939,
939    940,
940    941,
941    942,
942    943,
943    944,
944    945,
945    946,
946    947,
947    948,
948    949,
949    950,
950    951,
951    952,
952    953,
953    954,
954    955,
955    956,
956    957,
957    958,
958    959,
959    960,
960    961,
961    962,
962    963,
963    964,
964    965,
965    966,
966    967,
967    968,
968    969,
969    970,
970    971,
971    972,
972    973,
973    974,
974    975,
975    976,
976    977,
977    978,
978    979,
979    980,
980    981,
981    982,
982    983,
983    984,
984    985,
985    986,
986    987,
987    988,
988    989,
989    990,
990    991,
991    992,
992    993,
993    994,
994    995,
995    996,
996    997,
997    998,
998    999,
999    1000,
1000   1001,
1001   1002,
1002   1003,
1003   1004,
1004   1005,
1005   1006,
1006   1007,
1007   1008,
1008   1009,
1009   1010,
1010   1011,
1011   1012,
1012   1013,
1013   1014,
1014   1015,
1015   1016,
1016   1017,
1017   1018,
1018   1019,
1019   1020,
1020   1021,
1021   1022,
1022   1023,
1023   1024,
1024   1025,
1025   1026,
1026   1027,
1027   1028,
1028   1029,
1029   1030,
1030   1031,
1031   1032,
1032   1033,
1033   1034,
1034   1035,
1035   1036,
1036   1037,
1037   1038,
1038   1039,
1039   1040,
1040   1041,
1041   1042,
1042   1043,
1043   1044,
1044   1045,
1045   1046,
1046   1047,
1047   1048,
1048   1049,
1049   1050,
1050   1051,
1051   1052,
1052   1053,
1053   1054,
1054   1055,
1055   1056,
1056   1057,
1057   1058,
1058   1059,
1059   1060,
1060   1061,
1061   1062,
1062   1063,
1063   1064,
1064   1065,
1065   1066,
1066   1067,
1067   1068,
1068   1069,
1069   1070,
1070   1071,
1071   1072,
1072   1073,
1073   1074,
1074   1075,
1075   1076,
1076   1077,
1077   1078,
1078   1079,
1079   1080,
1080   1081,
1081   1082,
1082   1083,
1083   1084,
1084   1085,
1085   1086,
1086   1087,
1087   1088,
1088   1089,
1089   1090,
1090   1091,
1091   1092,
1092   1093,
1093   1094,
1094   1095,
1095   1096,
1096   1097,
1097   1098,
1098   1099,
1099   1100,
1100   1101,
1101   1102,
1102   1103,
1103   1104,
1104   1105,
1105   1106,
1106   1107,
1107   1108,
1108   1109,
1109   1110,
1110   1111,
1111   1112,
1112   1113,
1113   1114,
1114   1115,
1115   1116,
1116   1117,
1117   1118,
1118   1119,
1119   1120,
1120   1121,
1121   1122,
1122   1123,
1123   1124,
1124   1125,
1125   1126,
1126   1127,
1127   1128,
1128   1129,
1129   1130,
1130   1131,
1131   1132,
1132   1133,
1133   1134,
1134   1135,
1135   1136,
1136   1137,
1137   1138,
1138   1139,
1139   1140,
1140   1141,
1141   1142,
1142   1143,
1143   1144,
1144   1145,
1145   1146,
1146   1147,
1147   1148,
1148   1149,
1149   1150,
1150   1151,
1151   1152,
1152   1153,
1153   1154,
1154   1155,
1155   1156,
1156   1157,
1157   1158,
1158   1159,
1159   1160,
1160   1161,
1161   1162,
1162   1163,
1163   1164,
1164   1165,
1165   1166,
1166   1167,
1167   1168,
1168   1169,
1169   1170,
1170   1171,
1171   1172,
1172   1173,
1173   1174,
1174   1175,
1175   1176,
1176   1177,
1177   1178,
1178   1179,
1179   1180,
1180   1181,
1181   1182,
1182   1183,
1183   1184,
1184   1185,
1185   1186,
1186   1187,
1187   1188,
1188   1189,
1189   1190,
1190   1191,
1191   1192,
1192   1193,
1193   1194,
1194   1195,
1195   1196,
1196   1197,
1197   1198,
1198   1199,
1199   1200,
1200   1201,
1201   1202,
1202   1203,
1203   1204,
1204   1205,
1205   1206,
1206   1207,
1207   1208,
1208   1209,
1209   1210,
1210   1211,
1211   1212,
1212   1213,
1213   1214,
1214   1215,
1215   1216,
1216   1217,
1217   1218,
1218   1219,
1219   1220,
1220   1221,
1221   1222,
1222   1223,
1223   1224,
1224   1225,
1225   1226,
1226   1227,
1227   1228,
1228   1229,
1229   1230,
1230   1231,
1231   1232,
1232   1233,
1233   1234,
1234   1235,
1235   1236,
1236   1237,
1237   1238,
1238   1239,
1239   1240,
1240   1241,
1241   1242,
1242   1243,
1243   1244,
1244   1245,
1245   1246,
1246   1247,
1247   1248,
1248   1249,
1249   1250,
1250   1251,
1251   1252,
1252   1253,
```

ink! metadata



Ink! 语法的发展轨迹

```
contract! {  
    #![env = DefaultSrmlTypes]  
  
    struct Flipper {  
        value: storage::Value<bool>,  
    }  
  
    impl Deploy for Flipper {  
        fn deploy(&mut self) {  
            self.value.set(false)  
        }  
    }  
  
    impl Flipper {  
        pub(external) fn flip(&mut self) {  
            *self.value = !*self.value;  
        }  
    }  
}
```

```
#[ink::contract(version = "0.1.0")]  
mod flipper {  
    #[ink(storage)]  
    struct Flipper {  
        value: storage::Value<bool>,  
    }  
  
    impl Flipper {  
        #[ink(constructor)]  
        fn new(&mut self) {  
            self.value.set(false)  
        }  
  
        #[ink(message)]  
        fn flip(&mut self) {  
            *self.value = !*self.value;  
        }  
    }  
}
```

曾经的ink! 语法



```
contract! {  
    #![env = DefaultSrmlTypes]
```

ink!

```
}
```



现在的ink! 语法



```
#[ink::contract(version = "0.1.0")]  
mod flipper {
```

ink! + 

```
}
```



修改的好处 - Tooling

```
#[ink::contract(version = "0.1.0")]
mod flipper {
    #[ink(storage)]
    struct Flipper {
        value: storage::Value<bool>,
    }

    impl Flipper {
        #[ink(constructor)] fn new(&mut
self) {
            self.value.set(false)
        }

        #[ink(message)]
        fn flip(&mut self) {
            *self.value
            = !*self.value; }
    }
}
```

修改的好处 - Tooling

```
#[ink::contract(version = "0.1.0")]
mod flipper {
    #[ink(storage)]
    struct Flipper {
        value: storage::Value<bool>,
    }

    impl Flipper {
        #[ink(constructor)]
        fn new(&mut self) {
            self.value.set(false)
        }

        #[ink(message)]
        fn flip(&mut self) {
            *self.value = !*self.value;
        }
    }
}
```



rustfmt

如何写ink!

```
#![cfg_attr(not(feature = "std"), no_std)]
use ink_lang as ink;

#[ink::contract(version = "0.1.0")]
mod mycontract {
    use ink_core::storage;

    #[ink(storage)]
    struct MyContract {
        ...
    }

    #[ink(event)]
    struct MyContract {
    }

    impl MyContract {
        #[ink(constructor)]
        fn new(&mut self, initial_supply: Balance) {
        }

        #[ink(message)]
        fn call_a(&self) -> Balance {
        }

        #[ink(message)]
        fn call_b(&self, owner: AccountId) -> Balance {
        }
    }
}
```

ink! Header

```
# [ink::contract (  
    version = "0.1.0",  
    env = DefaultSrmlTypes,  
)]  
mod contract {  
    ...  
}
```

ink! Header

```
# [ink::contract(  
    version = "0.1.0",  
    env = DefaultSrmlTypes,  
)]  
mod contract {  
    ...  
}
```

ink! storage

ink!声明

```
#[ink(storage)]  
struct Erc20 {  
    total_supply: storage::Value<Balance>,  
    balances: storage::HashMap<AccountId, Balance>,  
    allowances: storage::HashMap<  
        (AccountId, AccountId), // Key  
        Balance                 // Value  
    >,  
}
```

ink! events

`#[ink(event)]`

```
struct Transfer {  
    #[ink(topic)] from: Option<AccountId>,  
    #[ink(topic)] to: Option<AccountId>,  
    value: Balance,  
}
```

定义

USAGE

```
self.env().emit_event(Transfer {  
    from: Some(from),  
    to: Some(to),  
    value,  
});
```


ink! constructors

```
#[ink(constructor)]
```

```
fn new(&mut self, initial_supply: Balance) {  
    let caller = self.env().caller();  
    self.total_supply.set(initial_supply);  
    self.balances.insert(caller, initial_supply);  
}
```

ink! messages

```
# [ink (message) ]
```

```
fn total_supply(&self) -> Balance {  
    self.total_supply.get()  
}
```

ink! errors

```
error: #[ink(constructor)] functions must have a &mut self receiver
--> src/lib.rs:52:16
```

```
|
52 |         fn new(&self, initial_supply: Balance) {
|               ^^^^^^
```

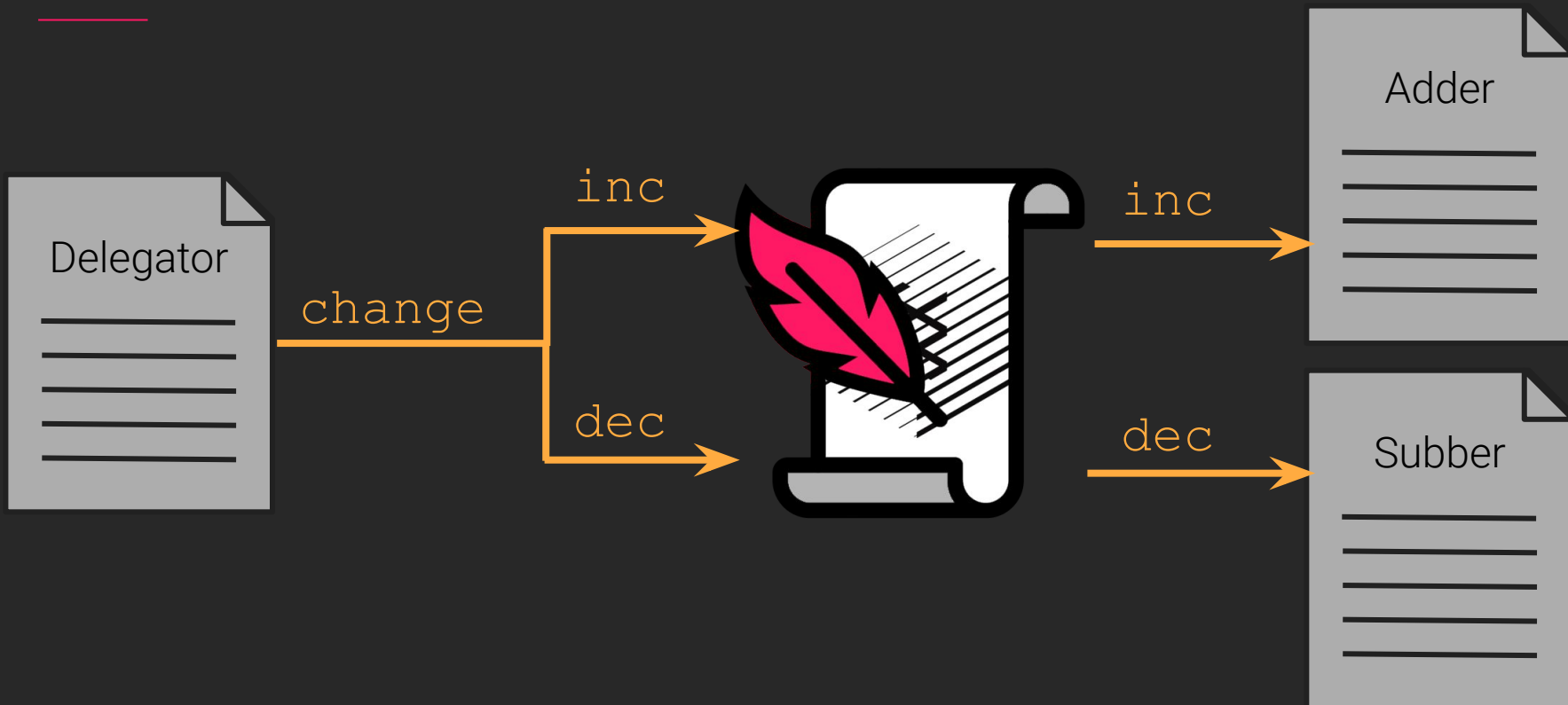
```
error: #[ink(constructor)] functions must not have a return type
--> src/lib.rs:52:52
```

```
|
52 |         fn new(&mut self, initial_supply: Balance) -> Self {
|                                                       ^^^^^^^^
```

ink! 跨合约调用

```
/// Delegates to either Adder or Subber contract.  
#[ink(message)]  
fn change(&mut self, by: i32) {  
    match self.which {  
        Which::Adder => self.adder.inc(by),  
        Which::Subber => self.subber.dec(by),  
    }  
}
```

ink! across borders



Ink! 支持的类型

存储类型

AccountId
Balance
Hash
Timestamp
Balance
BlockNumber
Call

Ink! 支持的类型

基本存储类型

Bitstash

Bitvec

Hashmap

Smallvec

Stash - 可索引的元素

Vec

Lazy/Value - 单值

复合存储类型

Struct

enum

Ink! 环境安装

Substrate的要求

```
curl https://getsubstrate.io -sSf | bash -s -- --fast  
rustup target add wasm32-unknown-unknown --toolchain stable  
rustup component add rust-src --toolchain nightly
```

安装指定版本 Substrate

```
cargo install node-cli --git  
https://github.com/paritytech/substrate.git --tag  
v2.0.0-rc4 --force
```

安装cargo contract 插件

```
cargo install cargo-contract --vers 0.6.1 --force
```


如何创建ink! 项目

```
cargo contract new <your-project>
```

如何测试ink! 项目

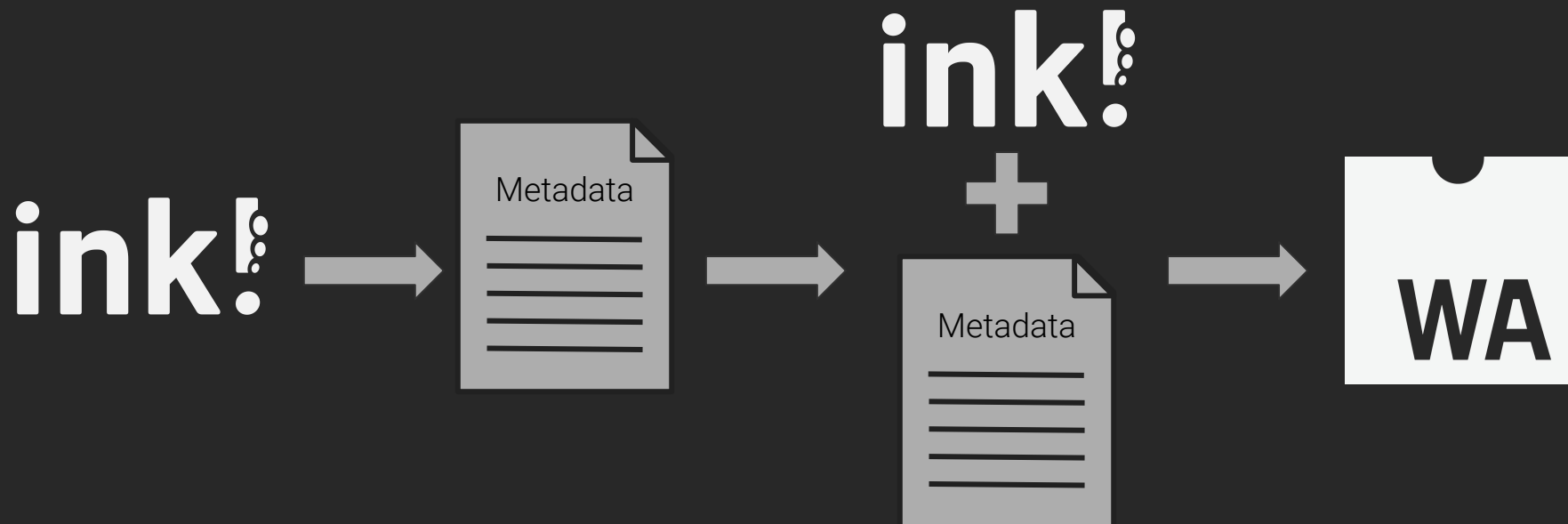
```
cargo +nightly test
```



演示

```
future<ink!>
```

Bake it; Eat it



Trait & Generics Support

```
#[ink::contract(version = "0.1.0")]
mod callback {
    #[ink(trait)]
    trait Listen {
        fn listen(&self, value: i32);
    }

    #[ink(storage)]
    struct Callback<T> {
        listener: storage::Value<T>,
    }

    ...
}
```

```
impl<T> Callback<T>
where
    T: Listen,
{
    #[ink(constructor)]
    fn new(&mut self, listener: T) {
        self.listener.set(listener)
    }

    #[ink(message)]
    fn fire(&self, value: i32) {
        self.listener.call(value)
    }
}
```

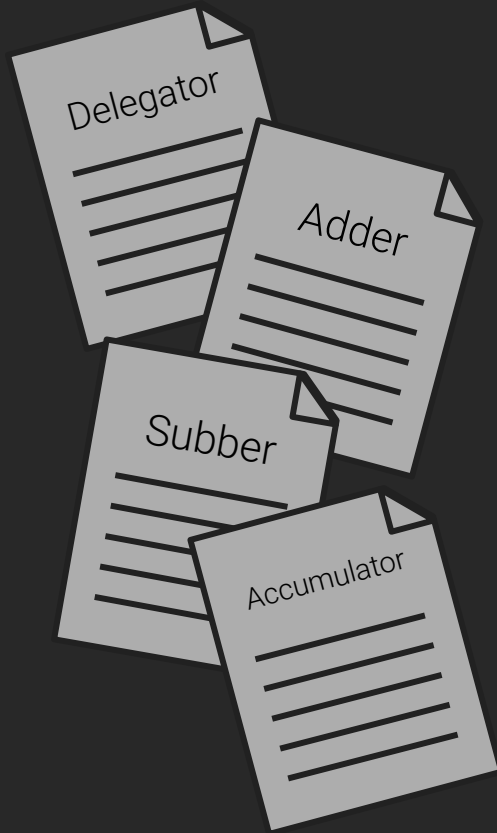
Multi-file Projects

```
#[ink::contract(version = "0.1.0")]
mod delegator {
    #[ink(storage)]
    struct Delegator { ... }

    #[ink(path = "accumulator")] mod accumulator;
    #[ink(path = "adder")] mod adder;
    #[ink(path = "subber")] mod subber;

    impl Delegator {
        #[ink(constructor)]
        fn new(&mut self, ...) { ... }

        #[ink(message)]
        fn delegate(&self, value: i32) { ... }
    }
}
```

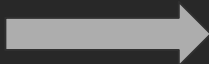


Pre- & Post Conditions

```
fn transfer(&mut self, from: AccountId, to: AccountId, value: Balance) {
    let from_balance = .. ;
    let to_balance = .. ;
    assume!(
        self.approval(from, to) >= value &&
        from_balance >= value
    );
    // do the computations that rely on the assumptions
    ensure!(
        self.approval(from, to) >= 0 &&
        self.balance_of(from) = from_balance - value &&
        self.balance_of(to) = to_balance + value
    );
}
```

Built-ink! Errors

```
#[ink(error)]  
enum Error {  
    /// Invalid message input.  
    InvalidInput,  
    /// Too few elements to process.  
    TooFewElements,  
    /// The value of the user is insufficient.  
    InsufficientValue,  
}
```



Rust Crate Attributes

crate root

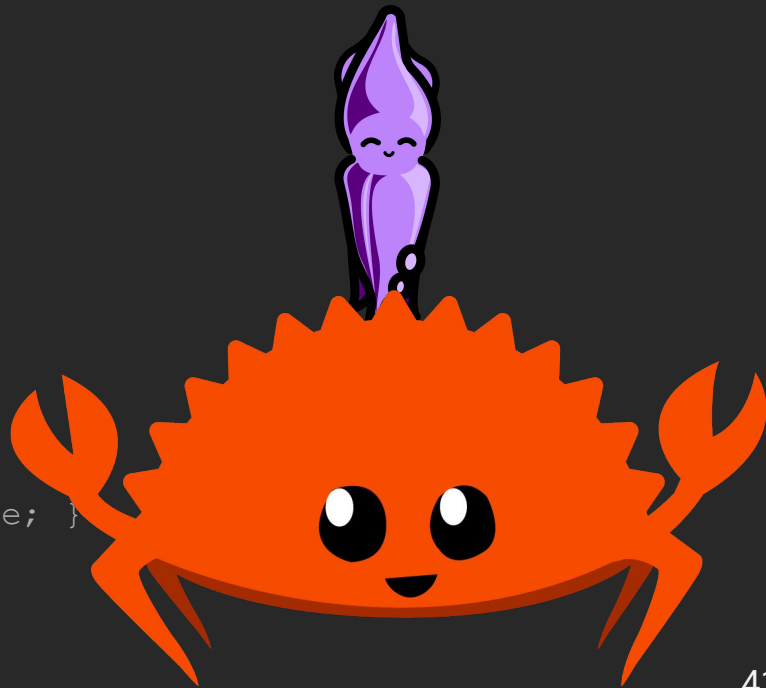
```
#![ink::contract(version = "0.1.0")]

#[ink(storage)]
struct Flipper {
    value: storage::Value<bool>,
}

impl Flipper {
    #[ink(constructor)]
    fn new(&mut self) { self.value.set(false) }

    #[ink(message)]
    fn flip(&mut self) { *self.value = !*self.value; }

    #[ink(message)]
    fn get(&self) -> bool { *self.value }
}
```



即将发布的ink!3.0

<https://github.com/paritytech/ink/pull/470>

Robbepop commented 7 days ago • edited

Member

Trait Support - New ink! Codegen

This PR implements the new ink! codegen to provide support for ink! trait implementations.

ToDo List

- ☐ Experiment if `proc-macro-error` crate is actually useful for us for error reporting in the `ink_lang_ir` crate.
- ☐ Make use of the new `ink_lang_ir` and `ink_lang_codegen` crates from `ink_lang_macro`.
- ☐ Implement the entire rest of ink! codegen:
 - ☒ `EnvTypes` trait
 - ☒ `ink! storage (#[ink(storage)]) struct`
 - ☒ `ink! event definitions`
 - ☒ event struct generation + trait impls
 - ☒ `EmitEvent` trait implementation
 - ☒ `ink! implementation blocks`
 - ☒ `ink! messages`
 - ☒ `ink! constructors`
 - ☒ `#[ink(impl)]` defined impl blocks with no ink! specific items
 - ☒ non-ink! specific impl items
 - ☐ metadata generation procedures
 - ☐ cross-calling definitions
 - ☐ dispatch routines
 - ☒ `ink! messages dispatch enum`
 - ☒ `ink! constructors dispatch enum`
 - ☒ `dispatch trait impls`
 - ☒ `dispatch entry points`
 - ☒ `DispatchUsingMode` impl
 - ☐ implement support for trait implementations in dispatch trait impls
 - ☐ All other non-ink! specific (Rust) items

Description

ink! CLI

```
cargo-contract 0.1.2
Parity Technologies <admin@parity.io>
Utilities to develop Wasm smart contracts.
```

USAGE:

```
cargo contract <SUBCOMMAND>
```

OPTIONS:

```
-h, --help      Prints help information
-V, --version    Prints version information
```

SUBCOMMANDS:

```
new      Setup and create a new smart contract.
build    Builds the smart contract.
test     Test the smart contract off-chain.
deploy   Deploy the smart contract on-chain. (Also for testing ,
help     Prints this message or the help of the given subcommand\
```



ink! playground

ink! playground

Substrate Node : not connected

COMPILE CODE

```
1  #![feature(proc_macro_hygiene)]
2  #![cfg_attr(not(feature = "std"), no_std)]
3
4  use ink_core::storage;
5  use ink_lang2 as ink;
6
7  #[ink::contract(version = "0.1.0")]
8  mod sample {
9      /// Defines the storage of your contract.
10     /// Add new fields to the below struct in order
11     /// to add new static storage fields to your contract.
12     #[ink(storage)]
13     struct Sample {
14         /// Stores a single 'bool' value on the storage.
15         value: storage::Value<bool>,
16     }
17
18     impl Sample {
19         /// Constructor that initializes the 'bool' value to the given 'init_value'.
20         #[ink(constructor)]
21         fn new(&mut self, init_value: bool) {
22             self.value.set(init_value);
23         }
24
25         /// Constructor that initializes the 'bool' value to 'false'.
26         ///
```

you can find it here:
/projects/sample/target/sample.wasm

```
[build abi]
cargo run -p abi-gen
Compiling sample v0.1.0 (/projects/sample)
Compiling abi-gen v0.1.0 (/projects/sample/.ink/abi_gen)
Finished dev [unoptimized + debuginfo] target(s) in 1.48s
Running `target/debug/abi-gen`
```

WASM

ABI