EWPA as analysis framework

- **EWPA** is intended to provide only a simple and easy-to-maintain environment where develop analysis code
  - decoupling data access to common inserter tools (unique and well-defined selections)
  - decoupling common analysis code (e.g. track matching)
  - allowing for UserData handling (add additional properties to event/particle)
  - allowing to save analysis results to POOL D2PD for later use and resource optimizations
  - allowing to develop independently different part of analysis (Tool1, Tool2, ...), test them separately and finally run them in sequence in a unique job
  - ...
- How many effort to port code in EWPA ?
  - just run **ewpa_makeClass** EWMyAnalysisTool
  - create the skeleton of class .h, .cxx, and jobOptions
  - cut and past your code
  - change loops to interact with EWEventLibrary (probably "slimming" them a lot ...)
- For more see **EWPA twiki** page ...

EWPA as analysis framework

- **EWPA** provides
  - ★ **EWInserters** to read and select particles form AOD/DnPD
  - ★ **EWAssociator** to match particles of different types
  - ★ **EWOverlapper** to remove overlap
  - ★ **EWStatManager** to easy get statistical features (histos, n-tuples, fits, ...)
  - ★ **EWEventLibrary** and **EWEventObject** to handle event and particle information
- I don't want to go in details: only few words on information handling
  - ★ **EWEventObject** is a data-class to record:
    - particle pointer (Muon*) with a label (EWPA::MuidCBMuon)
    - along with any desired particle-level UserData (tags, isolation, matches, overlaps, ...)
    - and inter-particle links (matchers, overlappers, ...)
  - ★ **EWEventLibrary** is a data-class to record:
    - EWEventObjectCollections with a label (EWPA::MuidCBMuon)
    - along with any desired event-level UserData (selections, run information, ...)
    - plus TrigDecisionTool and MissingET objects
  - ★ EWEventLibrary provides also methods for loops

EWPA as analysis framework

- **EWPA** provides
  - ★ **EWInserters** to read and select particles form AOD/DnPD
  - ★ **EWAssociator** to match particles of different types
  - ★ **EWOverlapper** to remove overlap
  - ★ **EWStatManager** to easy get statistical features (histos, n-tuples, fits, ...)
  - ★ **EWEventLibrary** and **EWEventObject** to handle event and particle information
- I don't want to go in details: only few words on information handling
  - ★ **EWEventObject** is a data-class to record:
    - particle pointer (Muon*) with a label (EWPA::MuidCBMuon)
    - along with any desired particle-level UserData (tags, isolation, matches, overlaps, ...)
    - and inter-particle links (matchers, overlappers, ...)
  - ★ **EWEventLibrary** is a data-class to record:
    - EWEventObjectCollections with a label (EWPA::MuidCBMuon)
    - along with any desired event-level UserData (selections, run information, ...)
    - plus TrigDecisionTool and MissingET objects
  - ★ EWEventLibrary provides also methods for loops, just 2 example in next slides ...

Example1: a simple loop

```
 const ParticleJetContainer* jetContainer;
 sc = m_storeGate->retrieve(jetContainer, m_jetContainerName);
 if (sc.isFailure() || !jetContainer) {
    log << MSG::WARNING << "No AOD Jet Container found !" << endreq;
    return StatusCode::SUCCESS;
  }
 ParticleJetContainer::const_iterator it = jetContainer->begin();
 for ( ; it != jetContainer->end() ; ++it ) {

   double pt = (*it)->pt();

}
```

Analysis Skeleton based

Example1: a simple loop

```
 const ParticleJetContainer* jetContainer;
 sc = m_storeGate->retrieve(jetContainer, m_jetContainerName);
 if (sc.isFailure() || !jetContainer) {
    log << MSG::WARNING << "No AOD Jet Container found !" << endreq;
    return StatusCode::SUCCESS;
  }
 ParticleJetContainer::const_iterator it = jetContainer->begin();
 for ( ; it != jetContainer->end() ; ++it ) {

   double pt = (*it)->pt();

}
```

Analysis Skeleton based

```
EWEventObject* jet;
m_EWevtlib->resetCounter(EWPA::JetCone7H1TW);
while(m_EWevtlib->getNext(EWPA::JetCone7H1TW, jet)) {

  double pt = jet->pt();

}
```

EWPA

Example2: a not so simple match

```
const TruthParticleContainer*  mcpartTES;
sc=m_storeGate->retrieve( mcpartTES, truthParticleContainerName);
if( sc.isFailure()  ||  !mcpartTES ) {
  log << MSG::WARNING << "No AOD MC truth particle container found in TDS" << endreq;
  return StatusCode::SUCCESS;
}
const MuonContainer* muonTES;
sc=m_storeGate->retrieve( muonTES, muonContainerName);
if( sc.isFailure()  ||  !muonTES ) {
    log << MSG::WARNING << "No AOD muon container found in TDS" << endreq;
    return StatusCode::SUCCESS;
}
MuonContainer::const_iterator muonItr  = muonTES->begin();
MuonContainer::const_iterator muonItrE = muonTES->end();
for (; muonItr != muonItrE; ++muonItr) {
   int index = -1;
  double deltaRMatch;
  const TruthParticleContainer * truthContainer = mcpartTES;
  bool findMatch = m_analysisTools->matchR((*muonItr),truthContainer,index,deltaRMatch,(*muonItr)->pdgId());
    if(findMatch) {
    }
    ....
}
```

Analysis Skeleton based

Example2: a not so simple match

```
const TruthParticleContainer*  mcpartTES;
sc=m_storeGate->retrieve( mcpartTES, truthParticleContainerName);
if( sc.isFailure()  ||  !mcpartTES ) {
  log << MSG::WARNING << "No AOD MC truth particle container found in TDS" << endreq;
  return StatusCode::SUCCESS;
}
const MuonContainer* muonTES;
sc=m_storeGate->retrieve( muonTES, muonContainerName);
if( sc.isFailure()  ||  !muonTES ) {
    log << MSG::WARNING << "No AOD muon container found in TDS" << endreq;
    return StatusCode::SUCCESS;
}
MuonContainer::const_iterator muonItr  = muonTES->begin();
MuonContainer::const_iterator muonItrE = muonTES->end();
for (; muonItr != muonItrE; ++muonItr) {
   int index = -1;
  double deltaRMatch;
  const TruthParticleContainer * truthContainer = mcpartTES;
  bool findMatch = m_analysisTools->matchR((*muonItr),truthContainer,index,deltaRMatch,(*muonItr)->pdgId());
    if(findMatch) {
    }
    ....
}
```

Analysis Skeleton based

```
EWEventObject* muon;
m_EWevtlib->resetCounter(EWPA::MuidCBMuon);
while(m_EWevtlib->getNext(EWPA::MuidCBMuon, muon)) {
   if(muon->isAssoPart(EWPA::TrueMuon)) {
      ...
   }
}
```

EWPA

Example2: a not so simple match

```
const TruthParticleContainer*  mcpartTES;
sc=m_storeGate->retrieve( mcpartTES, truthParticleContainerName);
if( sc.isFailure()  ||  !mcpartTES ) {
  log << MSG::WARNING << "No AOD MC truth particle container found in TDS" << endreq;
  return StatusCode::SUCCESS;
}
const MuonContainer* muonTES;
sc=m_storeGate->retrieve( muonTES, muonContainerName);
if( sc.isFailure()  ||  !muonTES ) {
    log << MSG::WARNING << "No AOD muon container found in TDS" << endreq;
    return StatusCode::SUCCESS;
}
MuonContainer::const_iterator muonItr  = muonTES->begin();
MuonContainer::const_iterator muonItrE = muonTES->end();
for (; muonItr != muonItrE; ++muonItr) {
   int index = -1;
  double deltaRMatch;
  const TruthParticleContainer * truthContainer = mcpartTES;
  bool findMatch = m_analysisTools->matchR((*muonItr),truthContainer,index,deltaRMatch,(*muonItr)->pdgId());
    if(findMatch) {
    }
    ....
}
```

Analysis Skeleton based

```
EWEventObject* muon;
m_EWevtlib->resetCounter(EWPA::MuidCBMuon);
while(m_EWevtlib->getNext(EWPA::MuidCBMuon, muon)) {
   if(muon->isAssoPart(EWPA::TrueMuon)) {
      ...
   }
}
```

matching previous calculated
with common tool

EWPA