

PRAIRIE DEV CON

CLOUD | MOBILE | WEB | DEV

EVENT SOURCING

OR, HOW I LEARNED TO STOP LAYERING AND LOVE THE EVENT LOG

Uni•nWare®



D2L
DESIRE2LEARN



SOLVERA
crafting results

imagnet

 Microsoft

online
business systems

 DMT

 Results. Guaranteed.

online

Chad Hurd
@CaffGeek

EVENT SOURCING

OR, HOW I LEARNED TO STOP LAYERING AND LOVE THE EVENT LOG

**Is your application in the gutter? Spare yourself the
grief, and strike out using Event Sourcing.**

Come on....don't be a turkey -- Adam Rehill

OUR JOURNEY

1. The Problem
2. A Familiar Story
3. The Solution
4. Some Code
5. Wrapping Up



PART 1

THE PROBLEM





DATA COMING IN

- 8 Provinces competing
- 5 Team Divisions
 - 5 people per team + coach
- 5 Singles Divisions
- 21 games
- Over 4 days

Totals 5040 individual game scores

- Names
- Averages
- Room Assignments
- Travel Information
- Practice Schedule
- Guests
- Profiles
- Lane Draws
- Hotel
- Bowling Centres
- Sponsors
- etc

DATA GOING OUT

Game 10

Sherwood lane 41-42



MB

Points: 6 1142 39 3

Jeff Bradshaw 238 8 1

Robbie Hendrickson 225 20 1

Tyler Robert 191 -25 0

David Wastle 245 14 0

Travis Manek 243 22 1



QC

Points: 2 1145 32 0

Guy Charron 182 -46 0

Yves Leblanc 207 -18 0

Garry Skene 201 -3 1

Michel Sauvé 323 91 1

Jean-Pierre Saumure 232 8 0

DATA GOING OUT











MB TOURNAMENT MEN

#	W/L	Score	vs	Opponent	Location
1	<u>L (0-8)</u>	1119	vs NL	1333	Sherwood #19
2	<u>L (3-5)</u>	1237	vs QC	1346	Sherwood #24
3	<u>L (2-6)</u>	1244	vs SO	1309	Sherwood #22
4	<u>W (8-0)</u>	1344	vs SK	1122	Sherwood #9
5	<u>L (2-6)</u>	1240	vs AB	1277	Sherwood #11
6	<u>W (7-1)</u>	1308	vs BC	1010	Sherwood #14
7	<u>W (7-1)</u>	1367	vs NO	1076	Sherwood #31
8	<u>W (7-1)</u>	1305	vs SO	1127	Sherwood #29
9	<u>L (2-6)</u>	1233	vs SK	1262	Sherwood #26
10	<u>W (6-2)</u>	1435	vs AB	1321	Sherwood #40

DATA GOING OUT

STANDINGS

Teaching Men

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Points
MB 	1006 (-97) 7 vs AB	1165 (52) 7 vs NO	979 (-124) 2 vs SO	1292 (189) 8 vs SK	1132 (29) 3 vs QC	1262 (159) 8 vs NL	1026 (-77) 5 vs BC	986 (-117) 2 vs SO	1081 (-22) 7 vs SK	1142 (39) 6 vs QC	1182 (79) 7 vs NL	1101 (-2) 2.5 vs AB	1135 (32) 7 vs BC	1127 (24) 6 vs NO	964 (-139) 5 vs QC	1128 (25) 6 vs NL	1043 (-60) 6 vs SO	1012 (-91) 3 vs SK	1008 (-95) 2 vs NO	1161 (58) 8 vs AB	1121 (18) 6 vs BC	113.5
QC 	1111 (-2) 6 vs SK	1043 (-70) 1 vs SO	1175 (62) 6 vs AB	969 (-144) 7 vs NO	1220 (107) 5 vs MB	1130 (17) 2 vs BC	1250 (137) 5 vs NL	1224 (111) 6 vs AB	1069 (-44) 7 vs NO	1145 (32) 2 vs MB	1155 (42) 2 vs BC	1261 (148) 8 vs SK	1334 (221) 8 vs NL	1111 (-2) 5 vs SO	962 (-151) 3 vs MB	1226 (113) 7.5 vs BC	1080 (-33) 7 vs AB	1066 (-47) 5 vs NO	1135 (22) 6 vs SO	1113 (0) 6 vs SK	1122 (9) 5 vs NL	109.5
BC 	1020 (33) 2 vs SO	1029 (42) 7.5 vs SK	1004 (17) 7 vs NO	987 (0) 7 vs AB	968 (-19) 6 vs NL	1012 (25) 6 vs QC	883 (-104) 3 vs MB	1167 (180) 7 vs NO	967 (-20) 6 vs AB	980 (-7) 2 vs NL	1045 (58) 6 vs QC	974 (-13) 6 vs SO	932 (-55) 1 vs MB	1043 (56) 7 vs SK	1011 (24) 6 vs NL	939 (-48) 0.5 vs QC	1072 (85) 8 vs NO	1009 (22) 6 vs AB	1028 (41) 6 vs SK	961 (-26) 6 vs SO	939 (-48) 2 vs MB	108
SO 	1236 (165) 6 vs BC	1091 (20) 7 vs QC	986 (-85) 6 vs MB	957 (-114) 6 vs NL	1072 (1) 2 vs AB	1247 (176) 6 vs SK	1122 (51) 7 vs NO	1119 (48) 6 vs MB	1179 (108) 7 vs NL	1102 (31) 5 vs AB	1137 (66) 7 vs SK	1018 (-53) 2 vs BC	1022 (-49) 7 vs NO	1003 (-68) 3 vs QC	1073 (2) 6 vs AB	1240 (169) 7 vs SK	981 (-90) 2 vs MB	1057 (-14) 3.5 vs NL	1063 (-8) 2 vs QC	1027 (-44) 2 vs BC	1073 (2) 6 vs NO	105.5
NL 	1116 (2) 7 vs NO	990 (-124) 5 vs AB	1109 (-5) 6 vs SK	951 (-163) 2 vs SO	1074 (-40) 2 vs BC	1052 (-62) 0 vs MB	1114 (0) 3 vs QC	1119 (5) 3 vs SK	1008 (-106) 1 vs SO	1149 (35) 6 vs BC	993 (-121) 1 vs MB	1112 (-2) 7 vs NO	954 (-160) 0 vs QC	1151 (37) 6 vs AB	1022 (-92) 2 vs BC	972 (-142) 2 vs MB	895 (-219) 1 vs SK	1131 (17) 4.5 vs SO	1124 (10) 5 vs AB	1249 (135) 7 vs NO	1064 (-50) 3 vs QC	73.5
SK 	947 (-66) 2 vs QC	928 (-85) 0.5 vs BC	984 (-29) 2 vs NL	1053 (40) 0 vs MB	1151 (138) 7 vs NO	1094 (81) 2 vs SO	950 (-63) 5 vs AB	1030 (17) 5 vs NL	825 (-188) 1 vs MB	964 (-49) 6 vs NO	977 (-36) 1 vs SO	841 (-172) 0 vs QC	906 (-107) 2 vs AB	940 (-73) 1 vs BC	1062 (49) 5 vs NO	1006 (-7) 1 vs SO	991 (-22) 7 vs NL	957 (-56) 5 vs MB	962 (-51) 2 vs BC	965 (-48) 2 vs QC	954 (-59) 5.5 vs AB	62
AB 	943 (-196) 1 vs MB	1009 (-130) 3 vs NL	1197 (58) 2 vs QC	918 (-221) 1 vs BC	1247 (108) 6 vs SO	1008 (-131) 2 vs NO	1022 (-117) 3 vs SK	1033 (-106) 2 vs QC	1051 (-88) 2 vs BC	1161 (22) 3 vs SO	1045 (-94) 1 vs NO	1188 (49) 5.5 vs MB	1133 (-6) 6 vs SK	1095 (-44) 2 vs NL	1080 (-59) 2 vs SO	1044 (-95) 6 vs NO	1011 (-128) 1 vs QC	1073 (-66) 2 vs BC	1083 (-56) 3 vs NL	967 (-172) 0 vs MB	1046 (-93) 2.5 vs SK	56
NO 	979 (-116) 1 vs NL	971 (-124) 1 vs MB	985 (-110) 1 vs BC	871 (-224) 1 vs QC	1050 (-45) 1 vs SK	992 (-103) 6 vs AB	1081 (-14) 1 vs SO	996 (-99) 1 vs BC	930 (-165) 1 vs QC	1009 (-86) 2 vs SK	1062 (-33) 7 vs AB	937 (-158) 1 vs NL	1027 (-68) 1 vs SO	935 (-160) 2 vs MB	1137 (42) 3 vs SK	909 (-186) 2 vs AB	1071 (-24) 0 vs BC	1002 (-93) 3 vs QC	1062 (-33) 6 vs MB	1000 (-95) 1 vs NL	942 (-153) 2 vs SO	44







PART 2

A FAMILIAR STORY





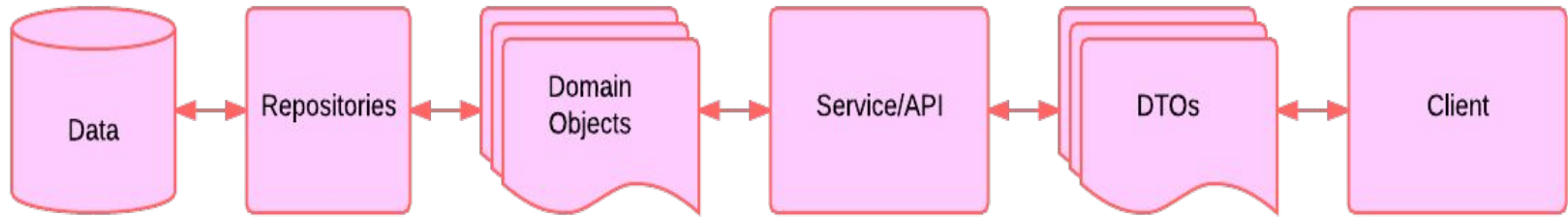








THE LAYERED APPROACH







TRADEOFFS

1. Models are often used for both reading & writing
2. We lost the context of why the change happened
3. We can't look into the past
4. Scaling our app

WHY?

```
domainModel.Name = dto.Name;  
  
domainModel.Age = dto.Age;  
  
domainModel.Gender = dto.Gender;  
  
domainModel.Address = dto.Address;  
  
domainModel.City = dto.City;  
  
domainModel.Province = dto.Province;
```









Reading is more
important than writing.

Roberto Bolaño



SUM UP

Tradeoffs

- Read vs Write
- Capturing Why
- The Past
- Scaling
- Read != Writing

PART 3

THE SOLUTION



KEY TERMS

Commands

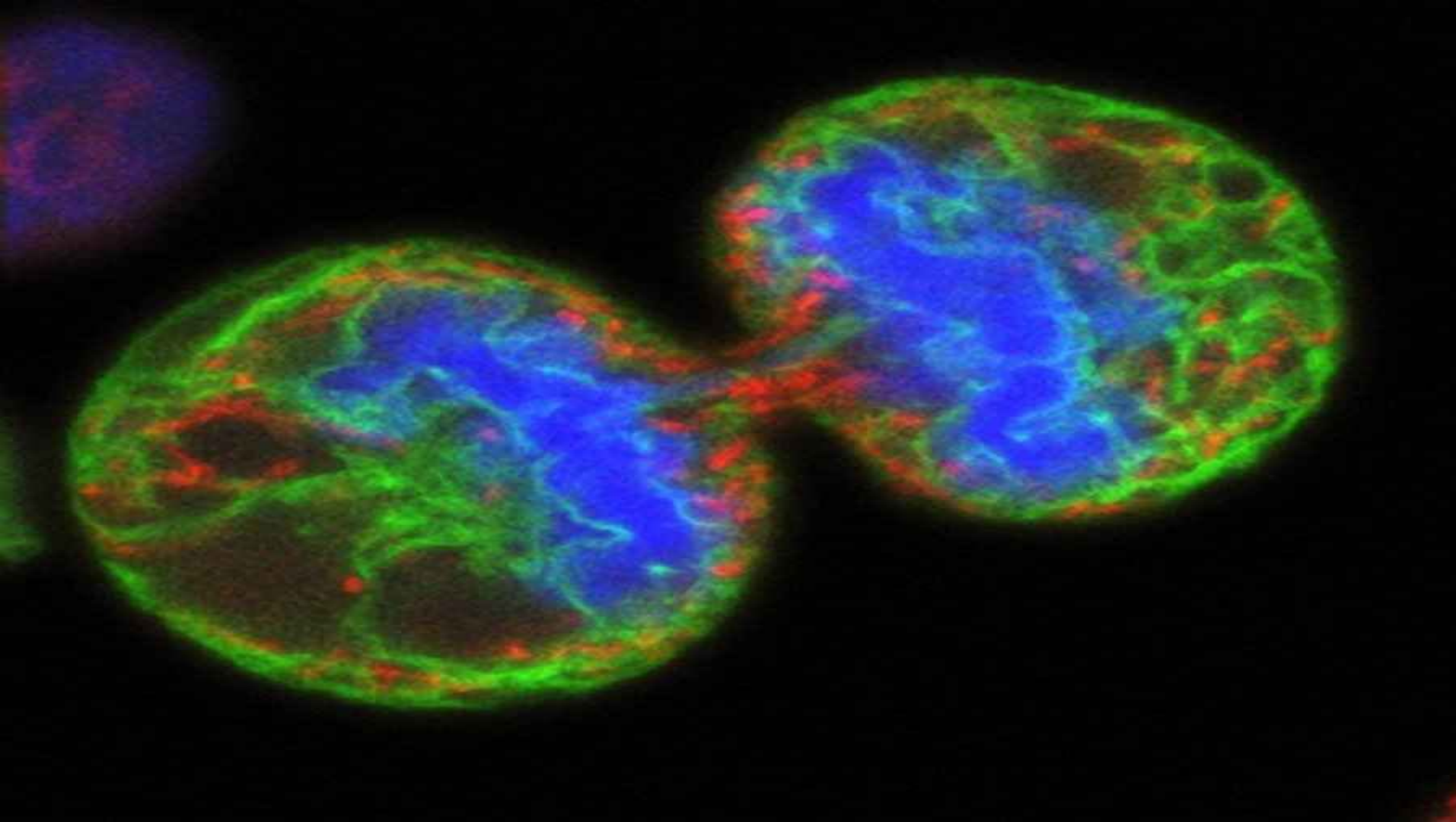
Events

Command Handlers

Aggregates

Read Models (Event Handlers)





WHAT DO WE NEED TO BUILD THIS

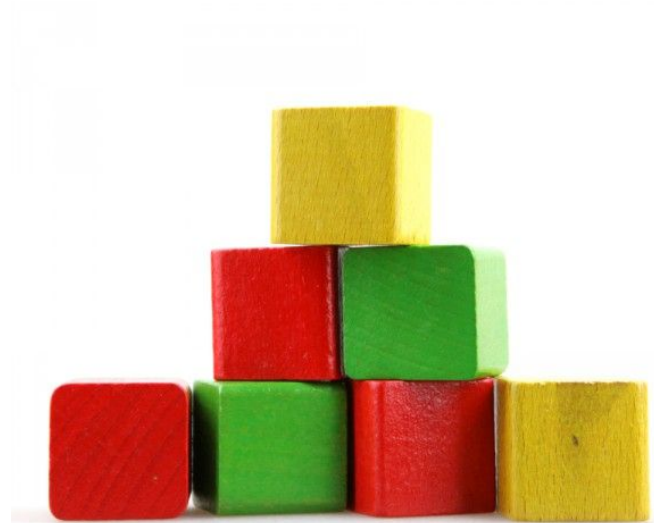
Command Handlers

Event Storage

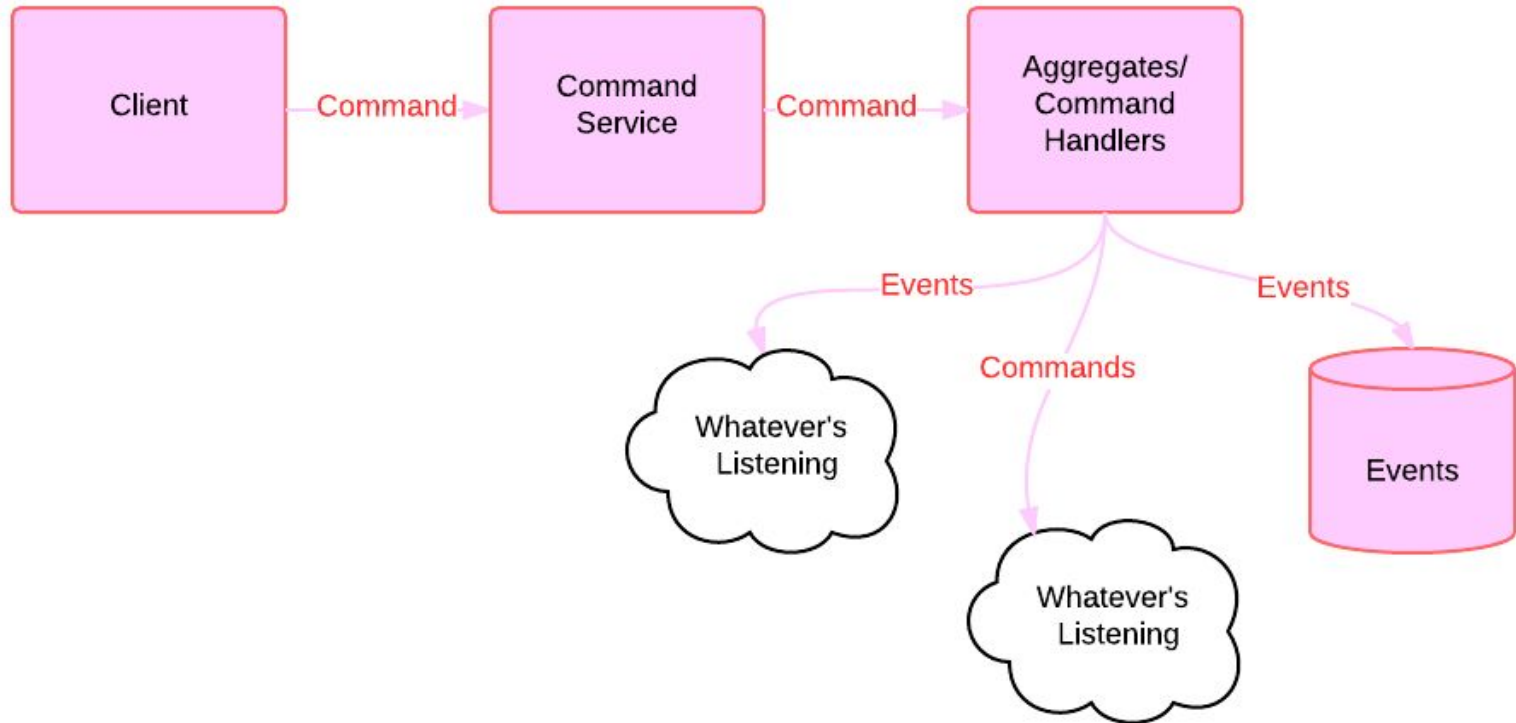
Event Handlers

Read Model Storage

Orchestration



THE WRITE SIDE



DATA COMING IN

Back

Game 21

MB Seniors

Patrick Walker (215)	2	208
Russell Knight (234)	5	217
Jean Shady (188)	1	218
Jim Anderson (215)	4	210
Tom Hinds (160)	3	109

SK Seniors

Denise Pillar (178)	2	136
Les Wardrop (193)	1	231
Jo-Ann Paxman (200)	3	202
Sheldon Kraus (187)	5	229
Larry Boehr (188)	4	179

Next

```
namespace MBACNationals.Scores.Commands
{
    public class SaveMatchResult
    {
        public Guid Id { get; set; }
        public Team Home { get; set; }
        public Team Away { get; set; }

        public class Team
        {
            public Guid Id { get; set; }
            public Bowler[] Bowlers { get; set; }
        }

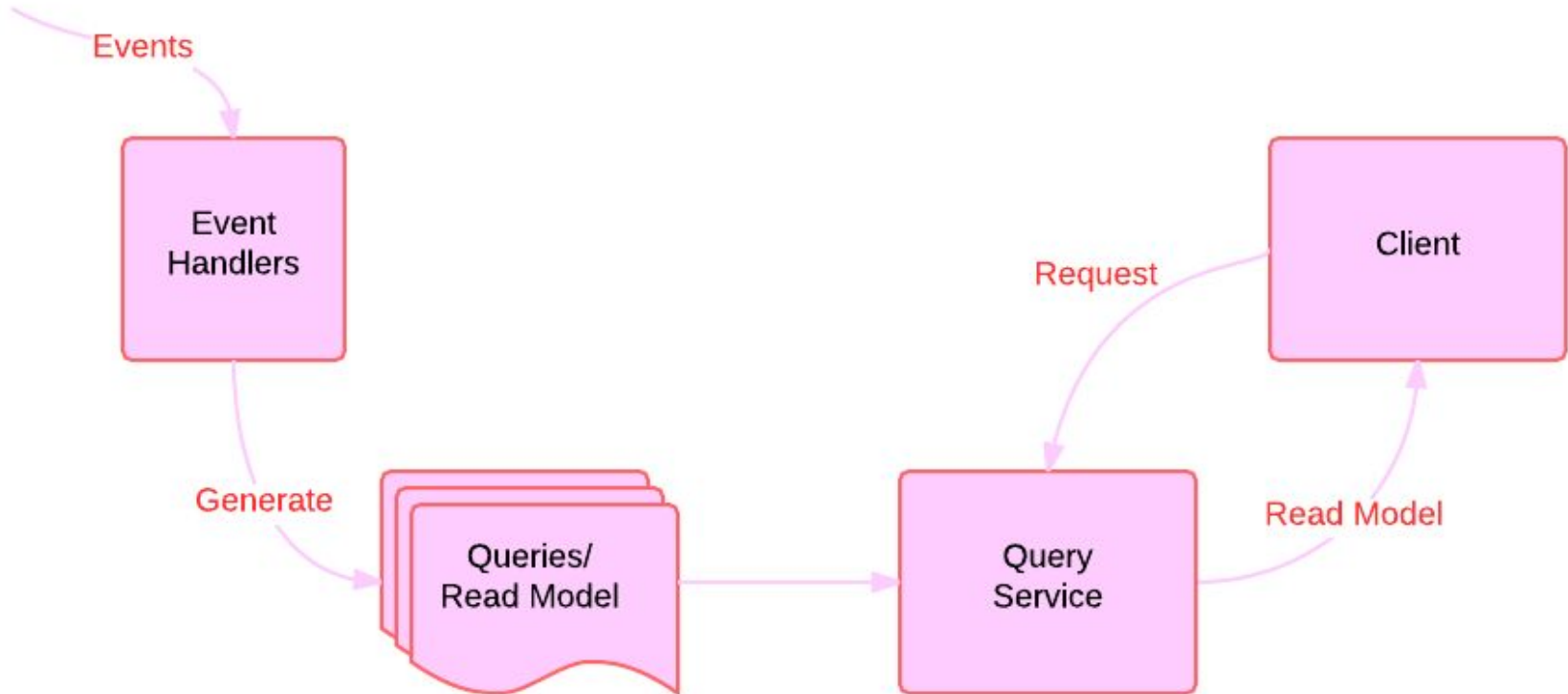
        public class Bowler
        {
            public Guid Id { get; set; }
            public int Position { get; set; }
            public int Score { get; set; }
        }
    }
}
```

RESULTING EVENTS

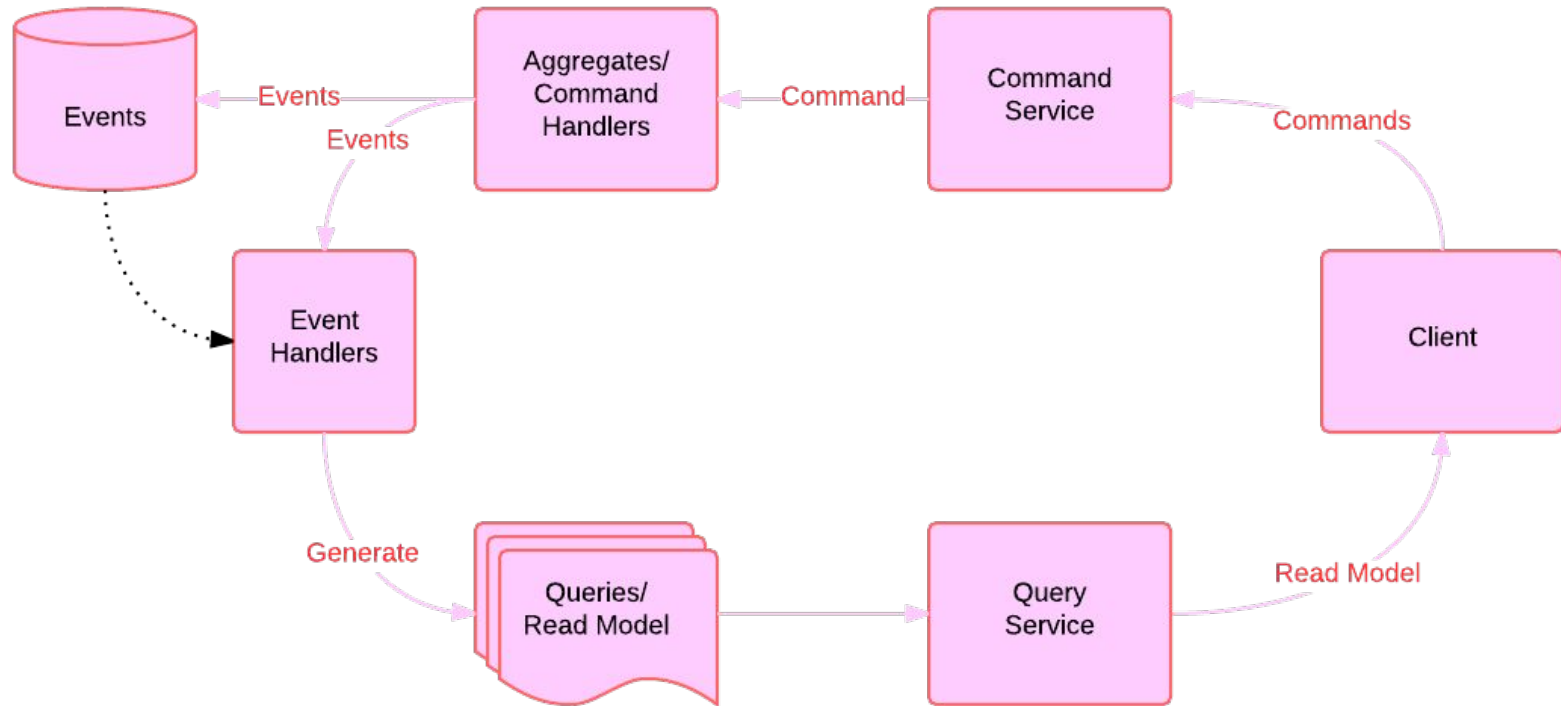
```
<MatchCompleted>
  <Id>6A69EBA4-331C-1C42-953C-CCA7A8D2814C</Id>
  <Division>Seniors</Division>
  <Home>MB</Home>
  <Away>SK</Away>
</MatchCompleted>

<ParticipantGameCompleted>
  <Id>6A69EBA4-331C-1C42-953C-CCA7A8D2814C</Id>
  <Participant>Russell Knight</Participant>
  <Score>243</Score>
</ParticipantGameCompleted>
```


THE READ SIDE

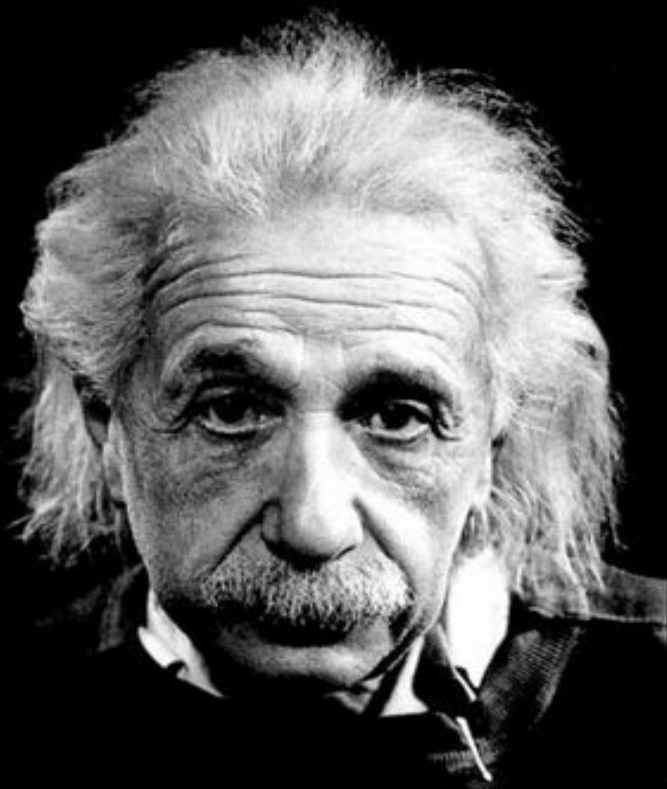


EVENT SOURCING



“Everything should be made
as simple as possible,
but not simpler.”

Albert Einstein

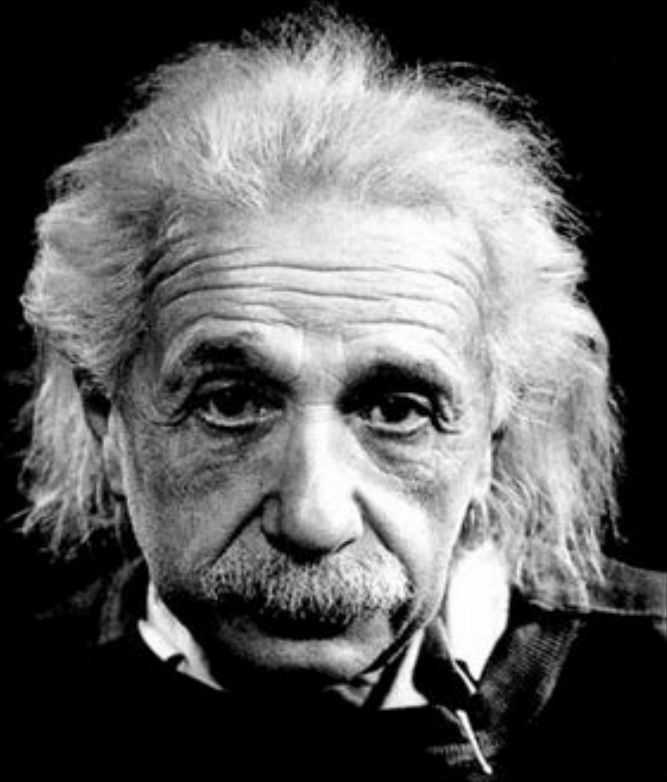


“Everything should be made
as simple as possible,
but not simpler.”

Albert Einstein

...what he actually said was

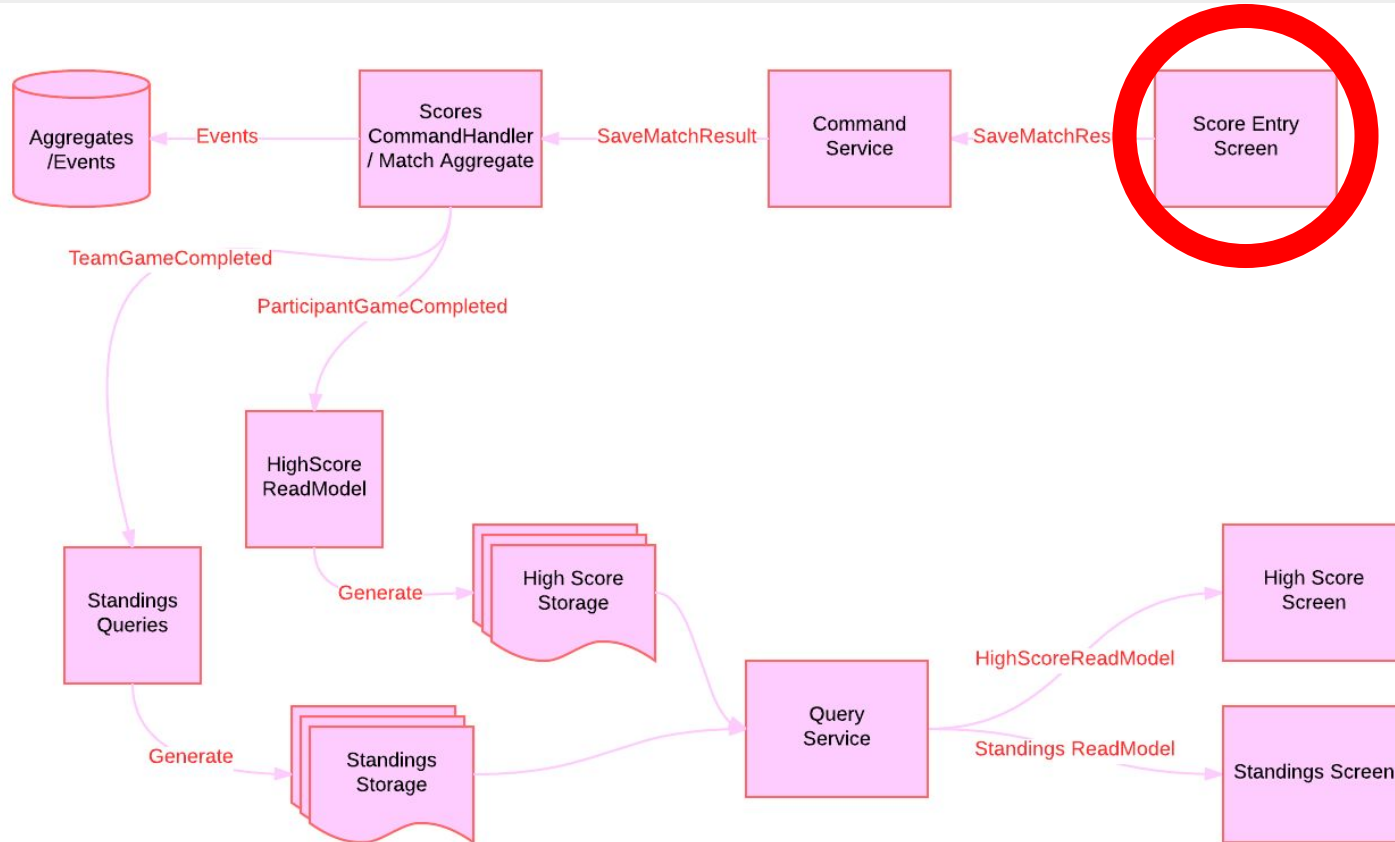
***“It can scarcely be denied that the
supreme goal of all theory is to
make the irreducible basic elements
as simple and as few as possible
without having to surrender the
adequate representation of a single
datum of experience.”***



PART 4

SOME CODE

ENTERING A SCORE



WE ENTER THE SCORES, ISSUING A COMMAND

[Back](#)

Game 21

MB Seniors

Patrick Walker (215)	2	208
Russell Knight (234)	5	217
Jean Shady (188)	1	218
Jim Anderson (215)	4	210
Tom Hinds (160)	3	109

SK Seniors

Denise Pillar (178)	2	136
Les Wardrop (193)	1	231
Jo-Ann Paxman (200)	3	202
Sheldon Kraus (187)	5	229
Larry Boehr (188)	4	179

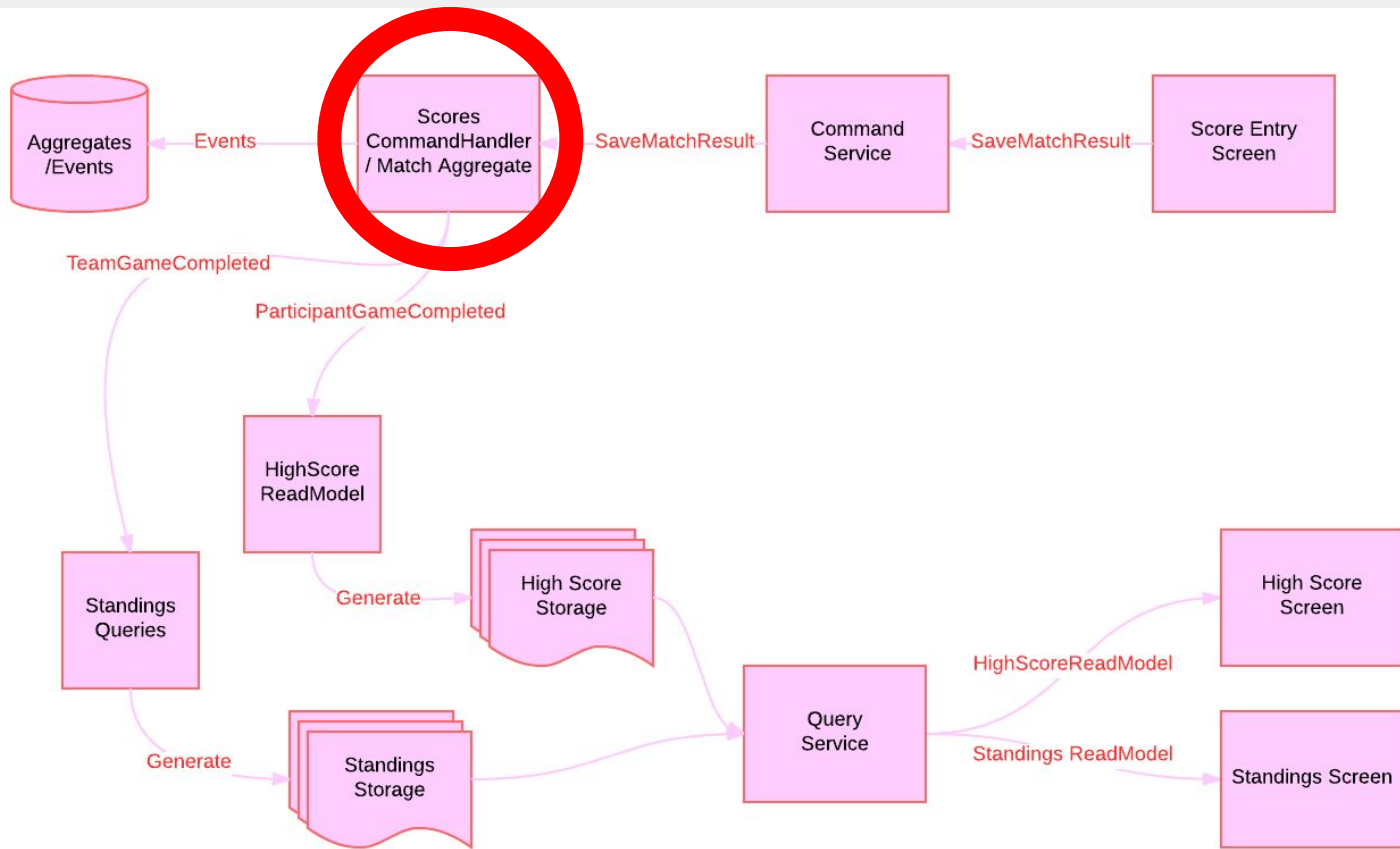
Next

```
namespace MBACNationals.Scores.Commands
{
    public class SaveMatchResult
    {
        public Guid Id { get; set; }
        public Team Home { get; set; }
        public Team Away { get; set; }

        public class Team
        {
            public Guid Id { get; set; }
            public Bowler[] Bowlers { get; set; }
        }

        public class Bowler
        {
            public Guid Id { get; set; }
            public int Position { get; set; }
            public int Score { get; set; }
        }
    }
}
```

PROCESSING THE COMMAND



```
public class ScoresCommandHandlers :
    IHandleCommand<SaveMatchResult, MatchAggregate>,
    IHandleCommand<SaveMatch, MatchAggregate>,
    IHandleCommand<CreateStepladderMatch, StepladderMatchAggregate>,
    IHandleCommand<UpdateStepladderMatch, StepladderMatchAggregate>,
    IHandleCommand<DeleteStepladderMatch, StepladderMatchAggregate>
{
    private MessageDispatcher _dispatcher;
    private ICommandQueries CommandQueries;

    public ScoresCommandHandlers(ICommandQueries commandQueries, MessageDispatcher dispatcher)
    {
        CommandQueries = commandQueries;
        _dispatcher = dispatcher;
    }

    public IEnumerable Handle(Func<Guid, MatchAggregate> al, SaveMatchResult command)
    {
        var agg = al(command.Id);

        if (agg.IsPOA)
        {
            //Single might no longer be first bowler if they are replaced
            var awaySingleParticipant = CommandQueries.GetTeamParticipants(command.Away.Id)
                .Where(x => !x.IsReplaced)
                .OrderBy(x => x.QualifyingPosition)
                .FirstOrDefault() ?? new CommandQueries.Participant();

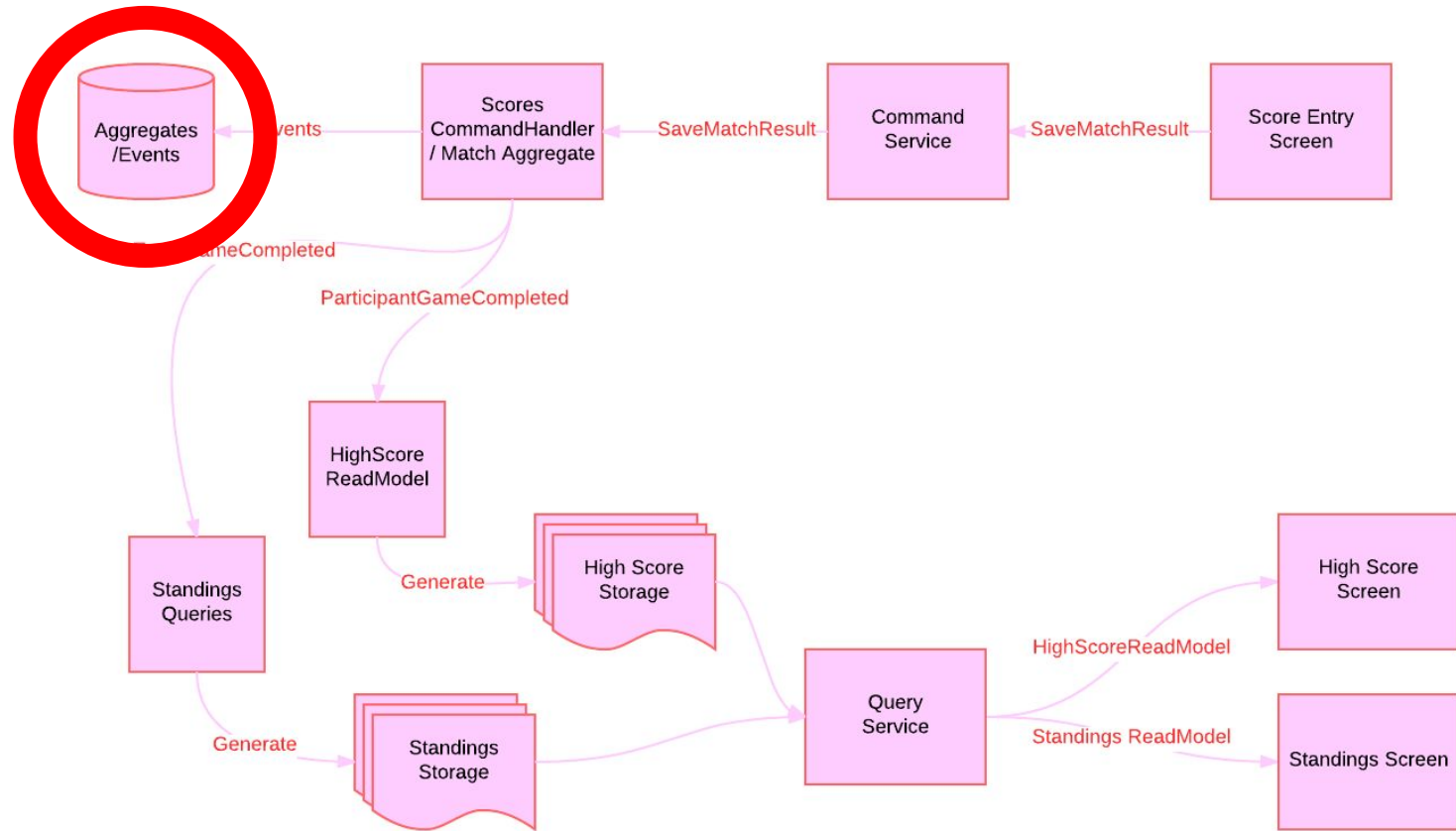
            var awayBowler = command.Away.Bowlers.FirstOrDefault(x => x.Id == awaySingleParticipant.Id)
                ?? command.Away.Bowlers.First(); // Never used to have QualifyingPosition

            var homeSingleParticipant = CommandQueries.GetTeamParticipants(command.Home.Id)
                .Where(x => !x.IsReplaced)
                .OrderBy(x => x.QualifyingPosition)
                .FirstOrDefault() ?? new CommandQueries.Participant();
        }
    }
}
```

RESULTING EVENTS

```
<MatchCompleted>  
  <Id>6A69EBA4-331C-1C42-953C-CCA7A8D2814C</Id>  
  <Division>Seniors</Division>  
  <Home>MB</Home>  
  <Away>SK</Away>  
</MatchCompleted>  
  
<ParticipantGameCompleted>  
  <Id>6A69EBA4-331C-1C42-953C-CCA7A8D2814C</Id>  
  <Participant>Russell Knight</Participant>  
  <Score>243</Score>  
</ParticipantGameCompleted>
```


STORE THE EVENTS



THE EVENT STORE

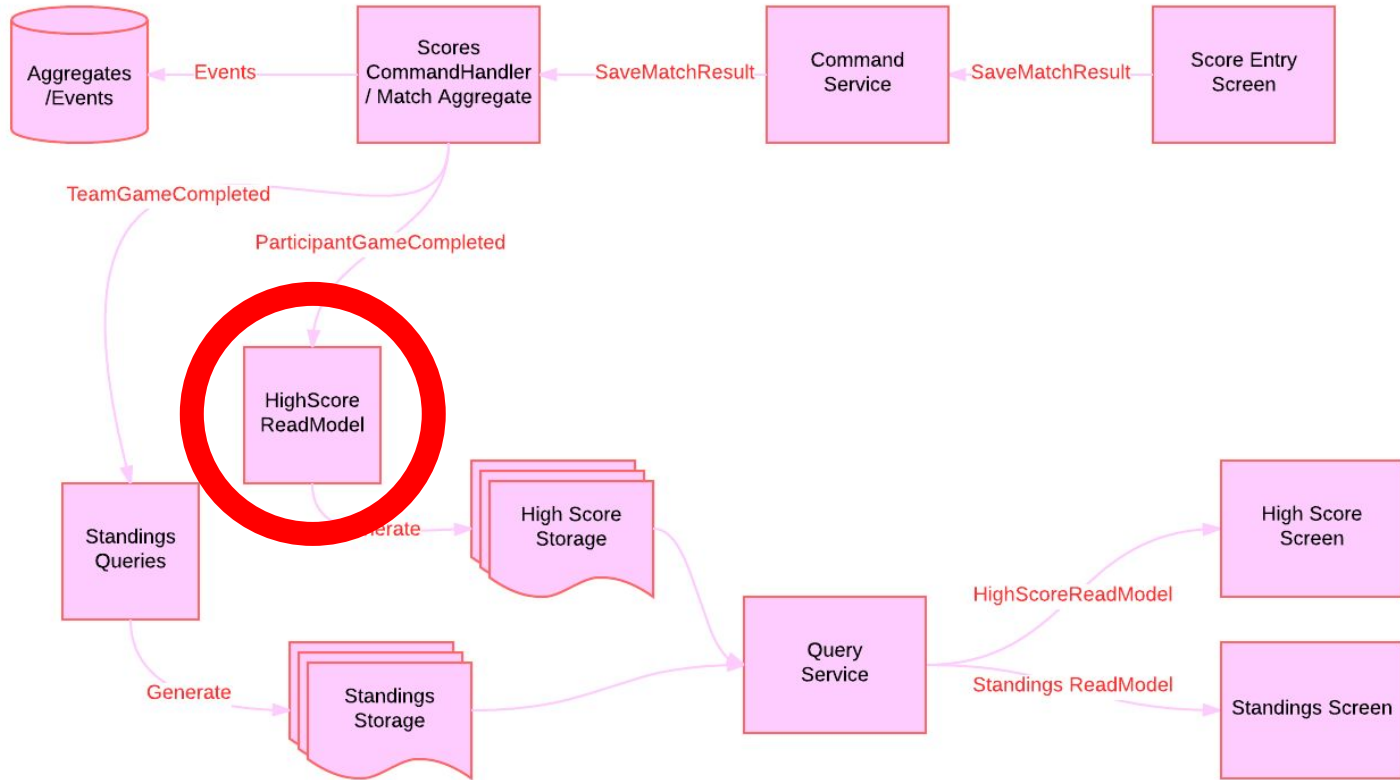
Aggregates

Id	uniqueidentifier
Type	varchar(max)

Events

AggregateId	uniqueidentifier
SequenceNumber	int
Type	nvarchar(max)
Body	nvarchar(max)
Timestamp	datetime2

BUILDING OUR READ MODELS



```
public void Handle(ParticipantGameCompleted e)
{
    var divisionName = SimplifyDivision(e.Division);
    var year = "2014";
    if (Matches.ContainsKey(e.Id))
    {
        year = Matches[e.Id];
    }

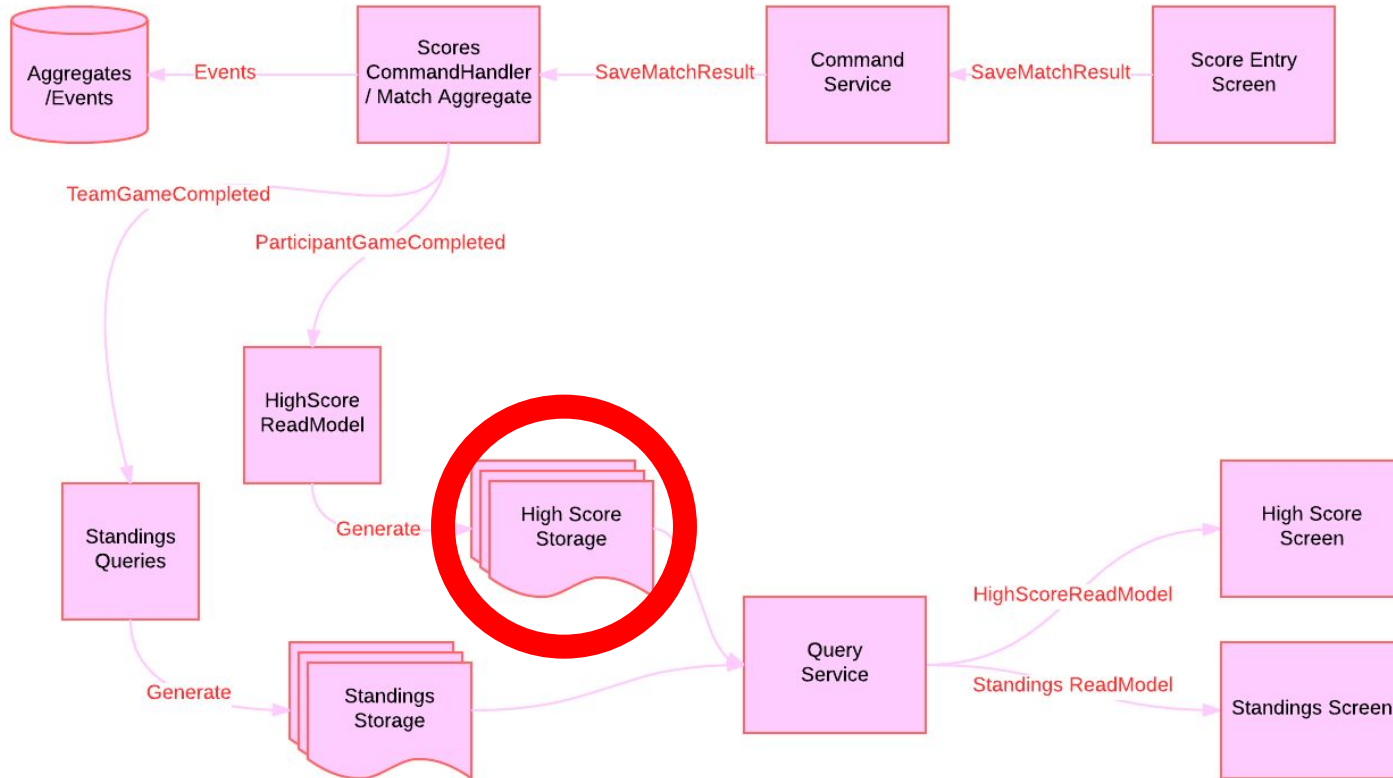
    var division = Divisions.SingleOrDefault(x => x.Name == divisionName && x.Year == year);
    if (division == null) ...

    division.Scores.RemoveAll(x => x.MatchId == e.Id && x.ParticipantId == e.ParticipantId);


    if (divisionName.Equals("Tournament", StringComparison.OrdinalIgnoreCase) && e.Score < 275) return;
    if (!divisionName.Equals("Tournament", StringComparison.OrdinalIgnoreCase) && e.POA < 75) return;

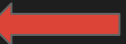
    division.Scores.Add(new Score
    {
        Gender = e.Gender,
        MatchId = e.Id,
        Name = e.Name,
        ParticipantId = e.ParticipantId,
        POA = e.POA,
        Scratch = e.Score,
        Year = year
    });
}
```

PERSISTING THE READ MODEL

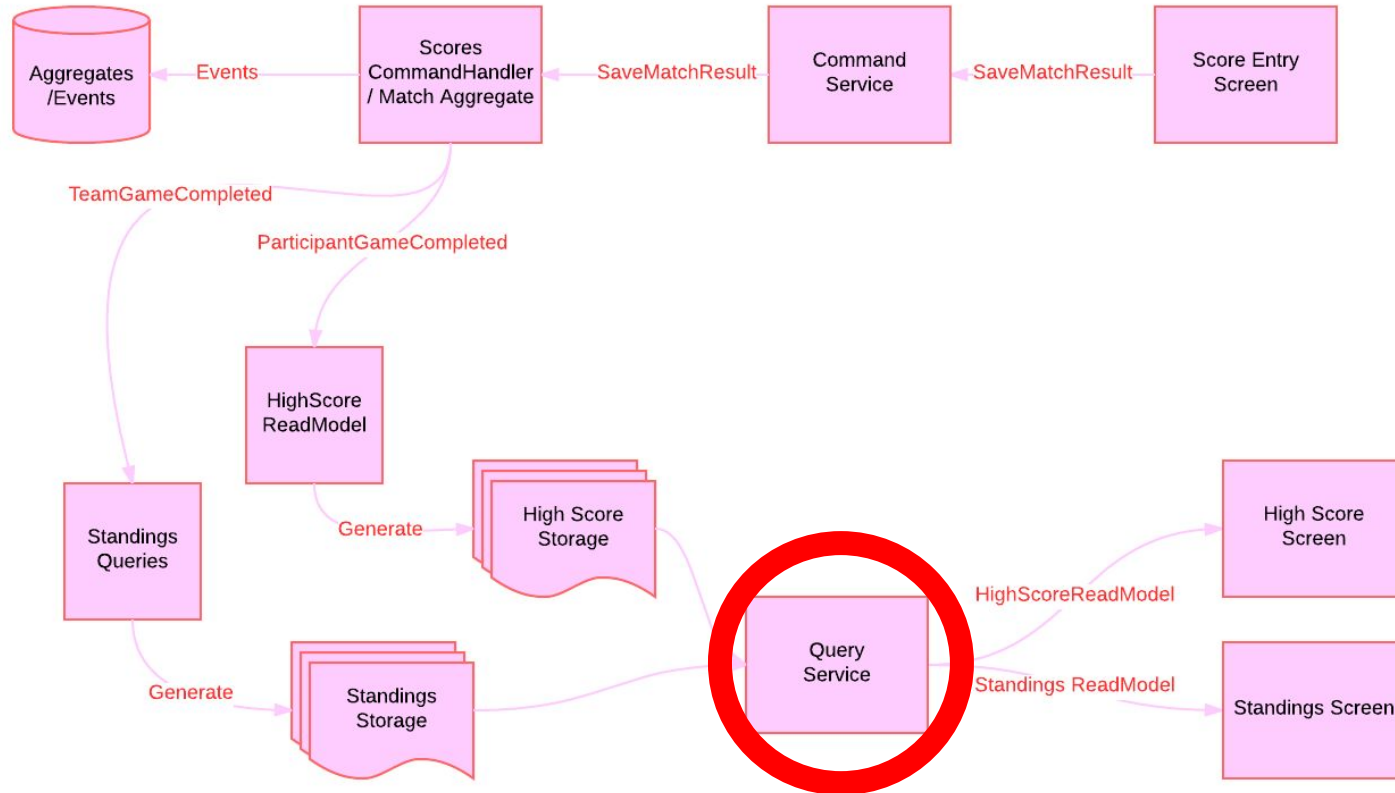



```
private void PublishEvent(object e)
{
    var eventType = e.GetType();
    if (eventSubscribers.ContainsKey(eventType))
        lock (eventPubLock)
            foreach (var sub in eventSubscribers[eventType])
            {
                sub(e);
                dynamic target = sub.Target;
                var y = target.subscriber as IReadModel;
                if (y != null) y.Save(); ←
            }
}
```

```
public static void Save(dynamic readModel)
{
    var jsonModel = JsonConvert.SerializeObject(readModel); 
    var container = GetContainer();
    CloudBlockBlob modelBlob = container.GetBlockBlobReference(readModel.GetType().Name);
    modelBlob.UploadText(jsonModel);
}
```

```
public static T Load<T>()
    where T : new()
{
    var container = GetContainer();
    CloudBlockBlob modelBlob = container.GetBlockBlobReference(typeof(T).Name);
    if (modelBlob.Exists())
    {
        var text = modelBlob.DownloadText();
        return JsonConvert.DeserializeObject<T>(text); 
    }
    else
    {
        return new T();
    }
}
```

QUERYING FOR DATA

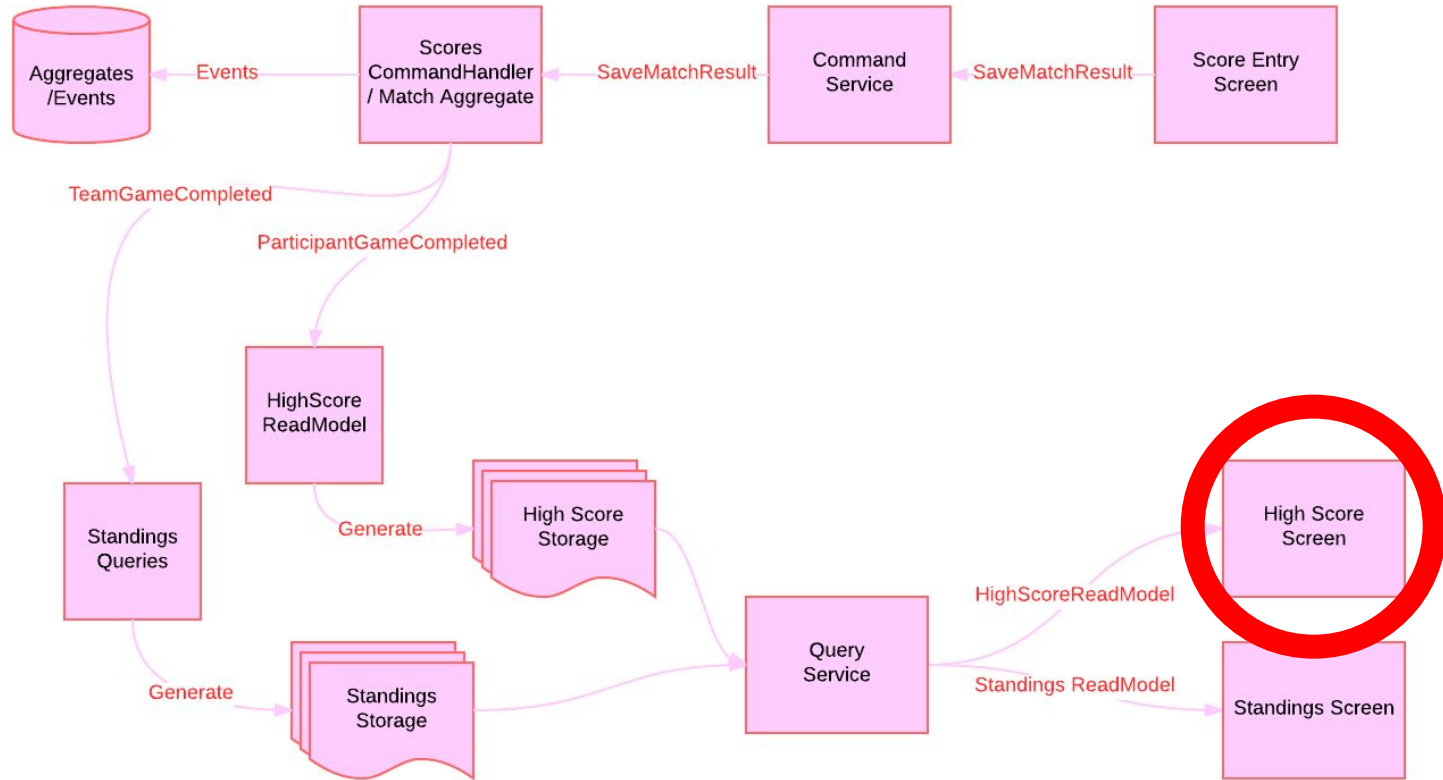


```
[HttpGet]
[OutputCache(NoStore = true, Duration = 0, VaryByParam = "None")]
public JsonResult Participant(Guid participantId)
{
    var participant = Domain.ParticipantScoreQueries.GetParticipant(participantId);
    return Json(participant, JsonRequestBehavior.AllowGet);
}
```

```
[HttpGet]
[OutputCache(NoStore = true, Duration = 0, VaryByParam = "None")]
public JsonResult Team(Guid teamId)
{
    var team = Domain.TeamScoreQueries.GetTeam(teamId);
    return Json(team, JsonRequestBehavior.AllowGet);
}
```

```
[HttpGet]
[OutputCache(NoStore = true, Duration = 0, VaryByParam = "None")]
public JsonResult HighScores(string division, int year)
{
    var highScores = Domain.HighScoreQueries.GetDivision(division, year);
    return Json(highScores, JsonRequestBehavior.AllowGet);
}
```

THE READ MODEL IS RETURNED



HIGH SCORES

TOURNAMENT DIVISION

MEN

- 448 - Kyle Young
- 410 - Gary Baird
- 398 - Stu Ryan

WOMEN

- 393 - Shauna Pirie-Laisnez
- 374 - Jennifer Baker
- 373 - Karen Armstrong

TEACHING DIVISION

MEN

- 390 - Brett Hendrickson
- 385 - Walter Fernets
- 375 - Rock durand
- +184 - Walter Fernets
- +156 - Rock durand
- +141 - Dave Lamarsh

WOMEN

- 375 - Jennifer Sauvé
- 326 - Marie-Pascale Croteau
- 321 - Jennifer Collier
- +183 - Jennifer Sauvé
- +133 - Jennifer Collier
- +125 - Sharon Farrell

SENIORS

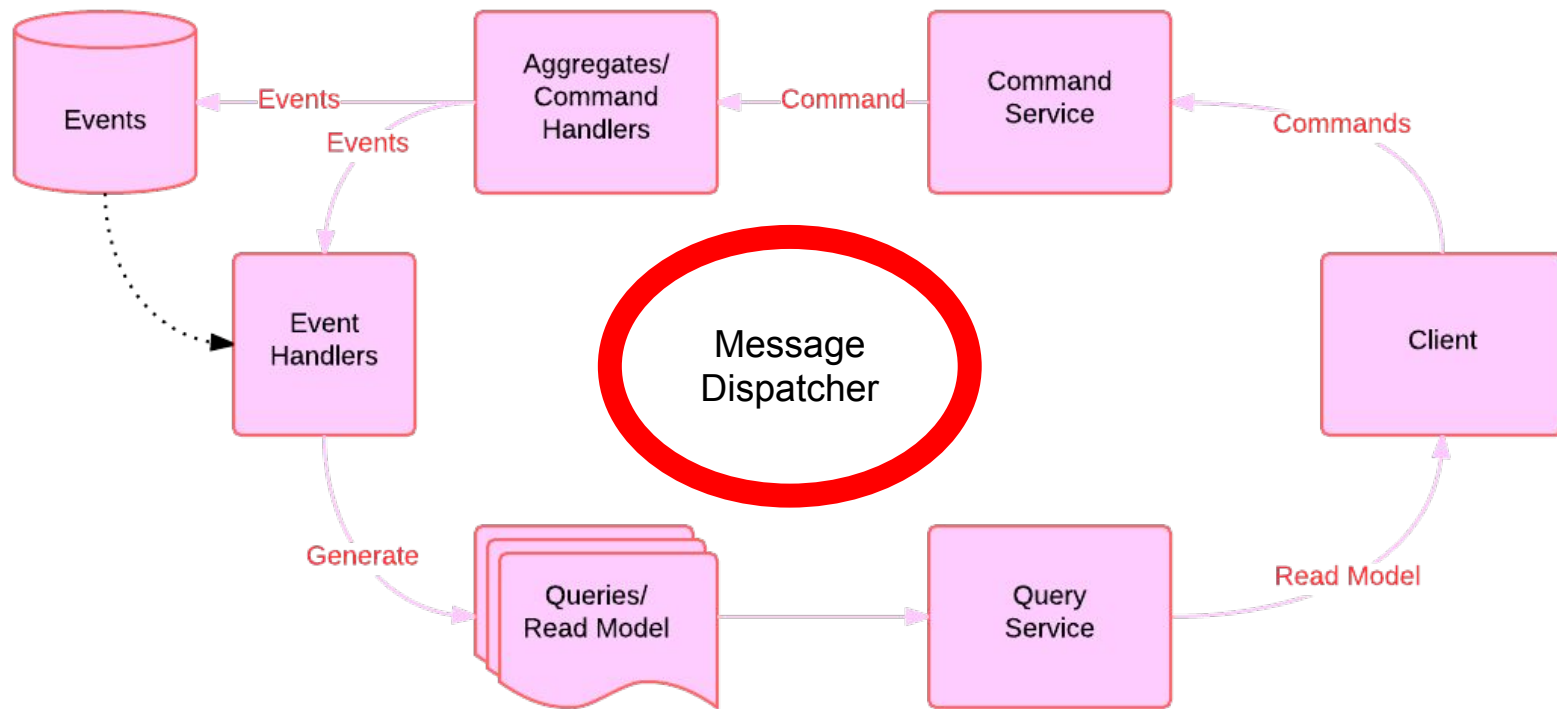
MEN

- 358 - Greg Gigliuk
- 344 - Jean-Pierre Saumure
- 341 - Jim Thorpe
- +127 - Baxter Vincent
- +118 - Jean-Pierre Saumure
- +109 - Greg Gigliuk

WOMEN

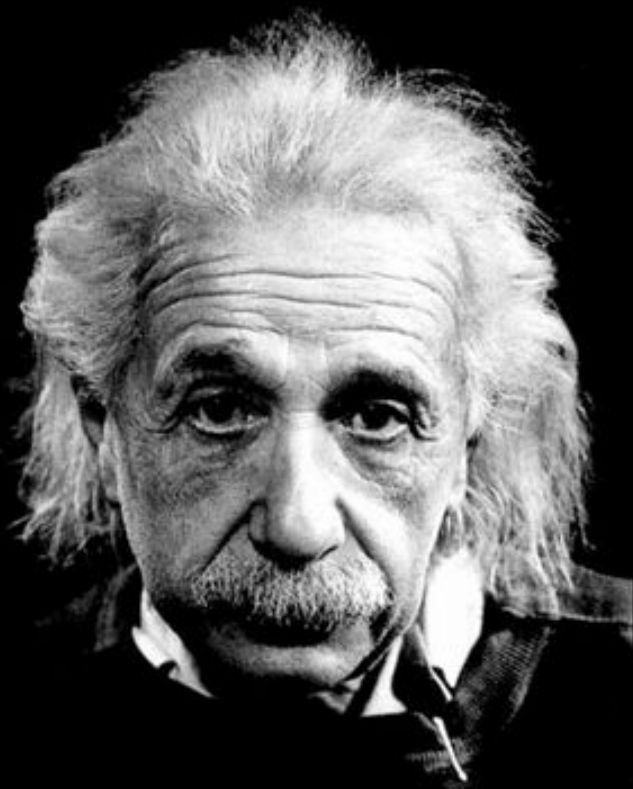
- 310 - Linda Ryan
- 288 - Linda Ryan
- 284 - Suzanne Layman
- +102 - Linda Ryan
- +91 - Suzanne Layman
- +88 - Louise Stevenson

FRAMEWORK



“Everything should be made
as simple as possible,
but not simpler.”

Albert Einstein



```
public void ScanInstance(object instance)
{
    // Scan for and register handlers.
    var handlers =
        from i in instance.GetType().GetInterfaces()
        where i.IsGenericType
        where i.GetGenericTypeDefinition() == typeof(IHandleCommand<,>)
        let args = i.GetGenericArguments()
        select new
        {
            CommandType = args[0],
            AggregateType = args[1]
        };
    foreach (var h in handlers)
        this.GetType().GetMethod("AddHandlerFor")
            .MakeGenericMethod(h.CommandType, h.AggregateType)
            .Invoke(this, new object[] { instance });

    // Scan for and register subscribers.
    var subscriber =
        from i in instance.GetType().GetInterfaces()
        where i.IsGenericType
        where i.GetGenericTypeDefinition() == typeof(ISubscribeTo<>)
        select i.GetGenericArguments()[0];
    foreach (var s in subscriber)
        this.GetType().GetMethod("AddSubscriberFor")
            .MakeGenericMethod(s)
            .Invoke(this, new object[] { instance });
}
```

```
public void AddHandlerFor<TCommand, TAggregate>(IHandleCommand<TCommand, TAggregate> handler) where TAggregate : Aggregate, new()
{
    if (commandHandlers.ContainsKey(typeof(TCommand)))
        throw new Exception("Command handler already registered for " + typeof(TCommand).Name);

    commandHandlers.Add(typeof(TCommand), c =>
    {
        var agg = new TAggregate();

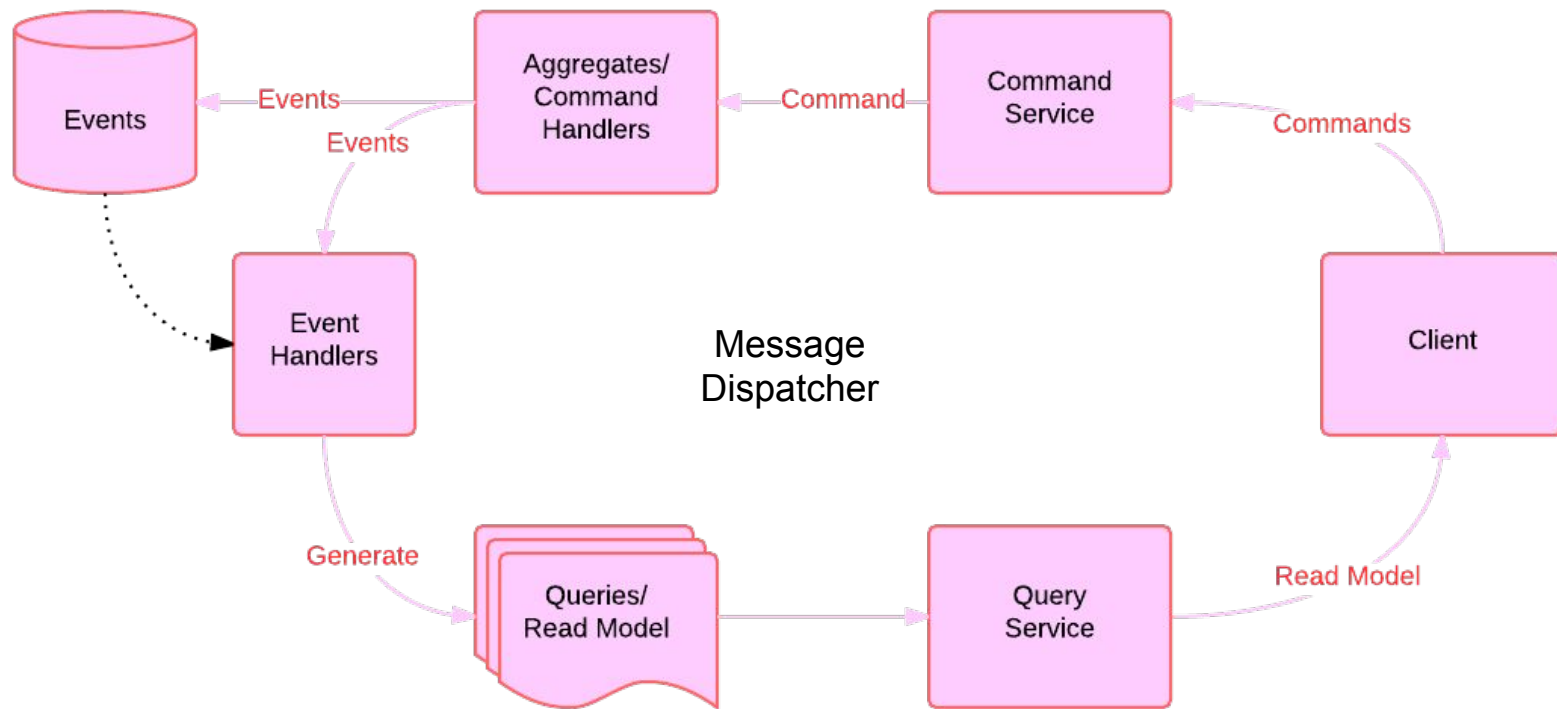
        Func<Guid, TAggregate> al = id =>
        {
            agg.Id = id;
            agg.ApplyEvents(eventStore.LoadEventsFor<TAggregate>(id));
            return agg;
        };

        // With everything set up, we invoke the command handler, collecting the events that it produces.
        var resultEvents = new ArrayList();
        foreach (var e in handler.Handle(al, (TCommand)c))
            resultEvents.Add(e);

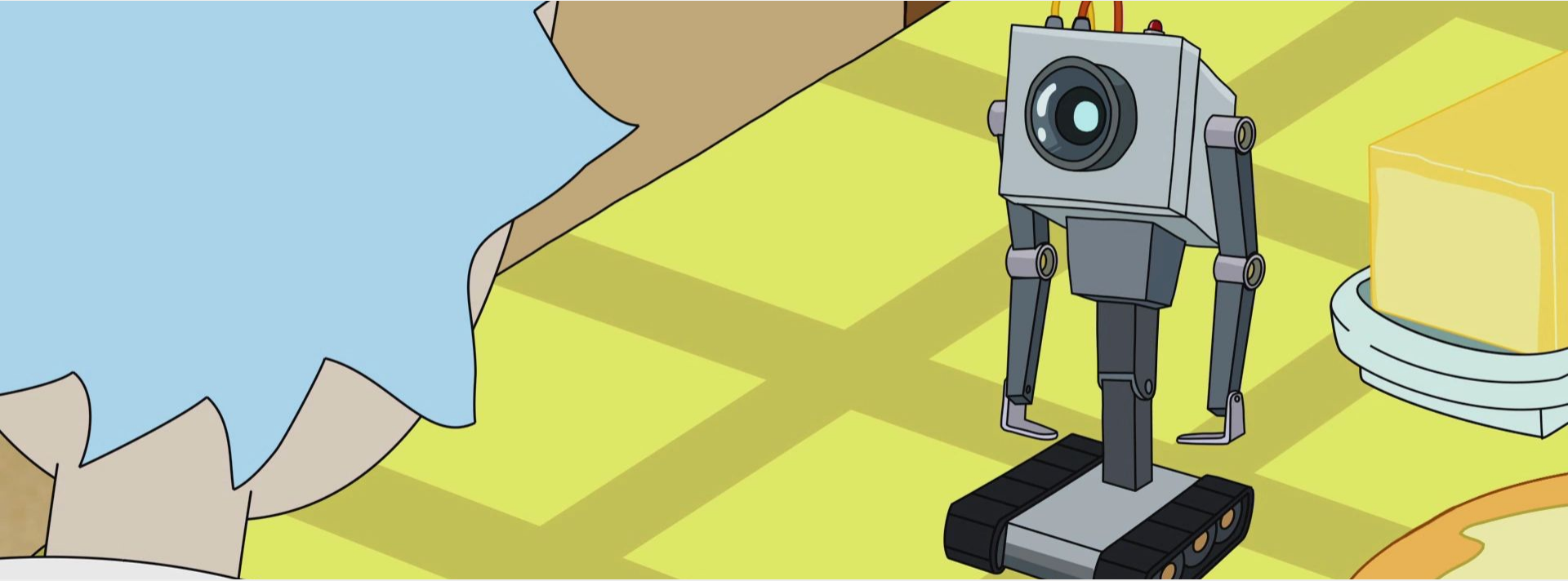
        // Store the events in the event store.
        eventStore.SaveEventsFor<TAggregate>(
            agg.Id,
            agg.EventsLoaded,
            resultEvents);

        // Publish them to all subscribers.
        foreach (var e in resultEvents)
            PublishEvent(e);
    });
}
```


FRAMEWORK



WHAT IS MY PURPOSE?



YOU PASS THE BUTTER

Another Home Improvement Project by

TERMITE™ TERRY
PEST CONTROL



949-631-7340

www.termiteterry.com

PART 5

WRAPPING UP

DISADVANTAGES TO EVENT SOURCING

Event versioning

Large aggregates

Messaging can be complex once you scale

Eventual Consistency can become a problem

ADVANTAGES

Event Store is a historical account

View state at any point in time

Read is optimized to be fast

Write is optimized to be correct

Scaling is easier

WHAT I WOULD DO DIFFERENTLY



**CHANGE
AHEAD**

My aggregates don't do much

My aggregates are not correct

CommandQueries are shared

Testing is mostly absent

WHAT DID I GET RIGHT

No Eventual Consistency

No Message Bus

Rebuilding Read Models

Choosing Event Sourcing



WHAT'S NEXT



 Results. Guaranteed.



online

Chad Hurd

@CaffGeek

Source: <https://github.com/CaffGeek/mbacnationals>