

Programming the web like a game

Applying ECS at mobile.de.

Sergiusz Urbaniak

"Old" public search

The screenshot shows the mobile.de search results page for a car search. The search criteria are:

- Preis von: Beliebig bis: Beliebig
- EZ von: Beliebig bis: Beliebig
- km von: Beliebig bis: Beliebig
- Land: Beliebig
- PLZ: Beliebig Umkreis: Kein
- Gewerbe, Ex-/Import: Nicht anzeigen
- Beschädigte Fahrzeuge: Nicht anzeigen

The results table shows one entry:

	Audi 100 Avant Top-Inserat	Händler	EZ 06/1983	210.000 km	450 EUR
<input type="checkbox"/>	Limousine / Gebrauchtfahrzeug HU 10/2014...	MHNNA AUTOEXPORT, 32791 Lage	Tel. +49 (0)5231 601917	Zur Händler-Homepage	

Envisioned public search

The screenshot shows the mobile.de search interface for a Volkswagen Golf. The search parameters are set to: Marke, Modell, Variante Volkswagen Golf; EZ von 2008 bis Beliebig; KM von Beliebig bis 100.000; Preis von Beliebig bis 20000 €. The results page displays three items:

- Volkswagen Golf VI 1.4 Trendline Sitzbez.Climatic Top-Insert**: Limousine / Gebrauchtfahrzeug, Händler EZ 07/2009, 71.000 km, 8.950 EUR.
- Volkswagen VW Golf Automatik ohne TÜV**: DE - 10781 Schöneberg, Privat EZ 01/1996, 90.000 km, 500 EUR.
- Volkswagen Golf CL**: DE - 13088 Berlin, Limousine / Gebrauchtfahrzeug, Schaltgetriebe, 51 kW (69 PS), Benzin.

New public search (aka PubSe)

suchen.mobile.de/fz/search.html (<http://suchen.mobile.de/fz/search.html>)

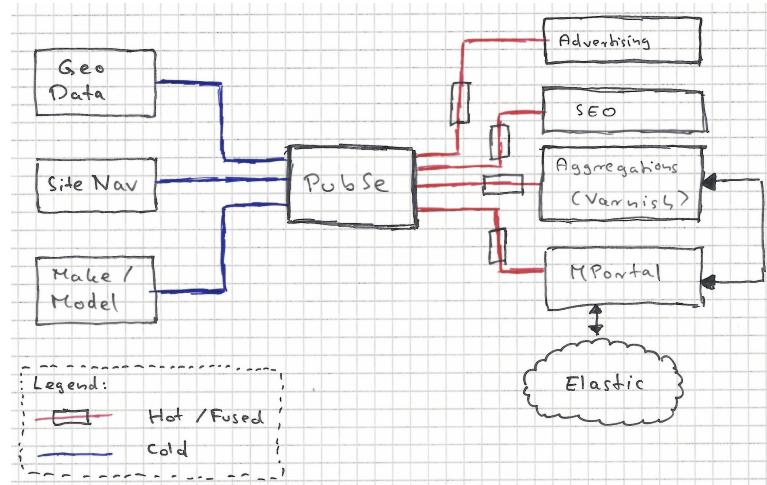
Goals:

- Break the monolith
- Sane service architecture (say microservices one more time...)
- Stack "not too far" from the existing one
- Move towards a twelve-factor-app [1]

Biggest goal: KISS (Keep it Super Simple)

[1] <http://12factor.net/> (<http://12factor.net/>)

System boundaries



Most of the "logic" is outside PubSe, how hard can it be?

Attempt #1

KISS (Keep it Super Simple)

1. Controller → Service → RemoteSearch
2. Soy ← Map<String, ?> ← JSON

Internal "service" classes:

- **SearchService**: Maps request parameter and invokes our search backend
- **RefDataService**: Loads reference data
- **AggregationService**: Aggregates data
- **Magellan**: Renders the "magellan"
- **Dutchman**: Renders the "dutchman"
- **PaginationViewModelBuilder**: Builds paging model
- ...

Problems with attempt #1

- Services have too many responsibilities, have too many LOC
- Concrete concerns (i.e. the "logic" for handling minPrice/maxPrice parameter) are spread all over the place

search.html?...&minPrice=5000&maxPrice=10000 (<http://suchen.mobile.de/fz/search.html?isSearchRequest=true&minPrice=5000&maxPrice=10000>)

minPrice=5000&maxPrice=10000)

```
▼ public-search-germany-webapp (14 occurrences)
  ▼ pubse.component.dutchman (2 occurrences)
    ▼ Dutchman.java (2 occurrences)
      ▼ of(RequestParameters, JsonObject, MakeModel18nResolver, Locale) (2 occurrences)
        (43: 23) d.single("minPrice", d::formatPrice);
        (44: 23) d.single("maxPrice", d::formatPrice);

  ▼ pubse.component.sorting (1 occurrence)
    ▼ SortingParams.java (1 occurrence)
      (32: 34) .put("searchNetGrossPrice", "p") // default for trucks api handles net/cross price

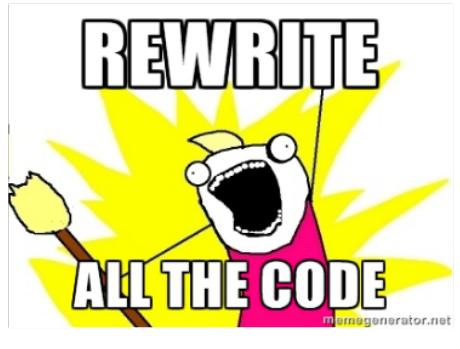
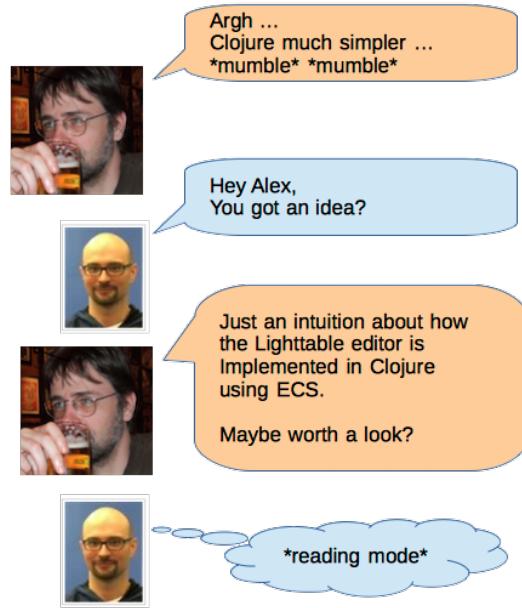
  ▶ pubse.ecs.entities (6 occurrences)
  ▼ pubse.service (5 occurrences)
    ▼ AggregationService.java (4 occurrences)
      ▼ formData(RequestParameters, JsonObject, JsonObject, Locale) (2 occurrences)
        (122: 36) .put(comboBox("minPrice", p, ref))
        (123: 36) .put(comboBox("maxPrice", p, ref))
      ▼ normalizeParameters(RequestParameters) (2 occurrences)
        (89: 33) switchValuesIf(np, "minPrice", "maxPrice", (min, max) -> parseInt(min) > parseInt(max));
        (89: 45) switchValuesIf(np, "minPrice", "maxPrice", (min, max) -> parseInt(min) > parseInt(max));
```

PubSea team@retro



Back at the desk

Light Table: Chris Granger - The IDE as a value (<http://www.chris-granger.com/2013/01/24/the-ide-as-data/>)



Programming is hard, let's talk about games!

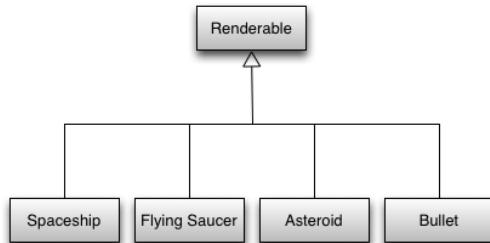


Source [www.arcade-museum.com](http://www.arcade-museum.com/game_detail.php?game_id=7146) (http://www.arcade-museum.com/game_detail.php?game_id=7146)

Ingredients

- Spaceship, Flying Saucer
- Asteroids, Bullet
- Static objects, Force Field

A classical OO model

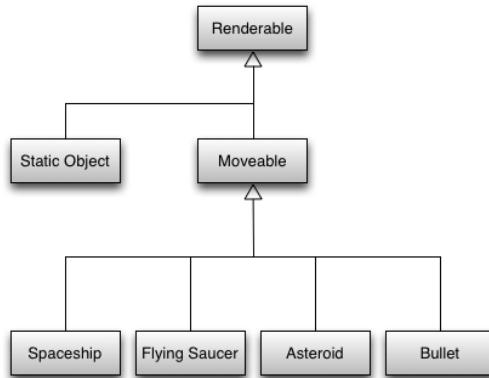


Source [Richard Lord - What is an entity framework](http://www.richardlord.net/blog/what-is-an-entity-framework) (<http://www.richardlord.net/blog/what-is-an-entity-framework>)

Nice, but ...

- the spaceship, flying saucer and asteroid are all movable too!
- Plus there are static objects

A classical OO model

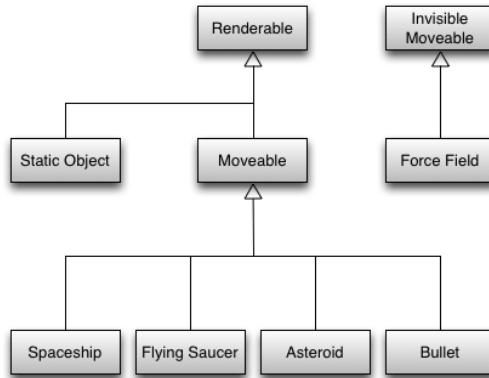


Source [Richard Lord - What is an entity framework](http://www.richardlord.net/blog/what-is-an-entity-framework) (<http://www.richardlord.net/blog/what-is-an-entity-framework>)

Nice, but ...

- What if we want an invisible game object? (Moveable but not Renderable)

A classical based OO model



Source [Richard Lord - What is an entity framework](http://www.richardlord.net/blog/what-is-an-entity-framework) (<http://www.richardlord.net/blog/what-is-an-entity-framework>)

- Our type hierarchy becomes complex
- We're missing multiple inheritance

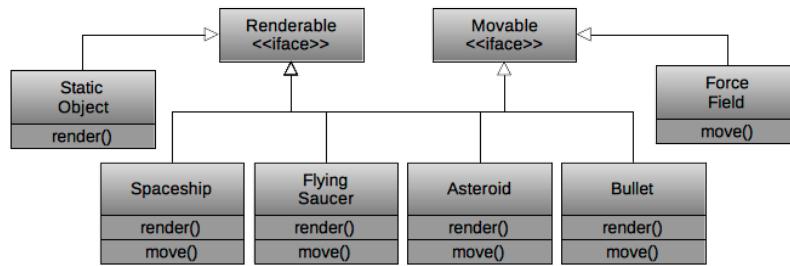
Hah, we have interfaces!

Problem:

- render() methods are redundant

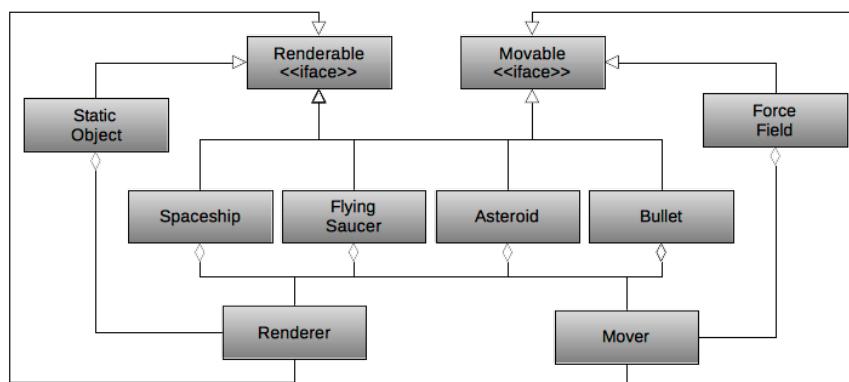
Solution:

- Favor composition over inheritance



An interface based OO model

Hmm ... that's quite a lot of arrows ...

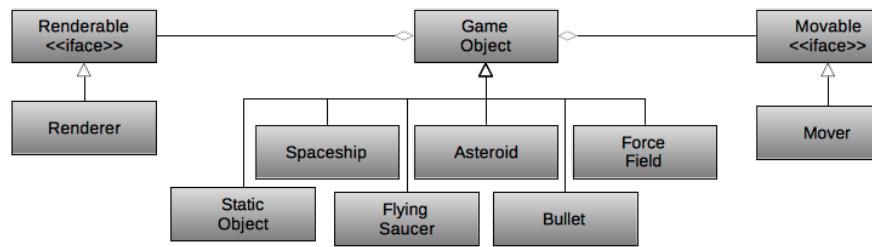


Patterns, patterns everywhere

Let's use bridges!

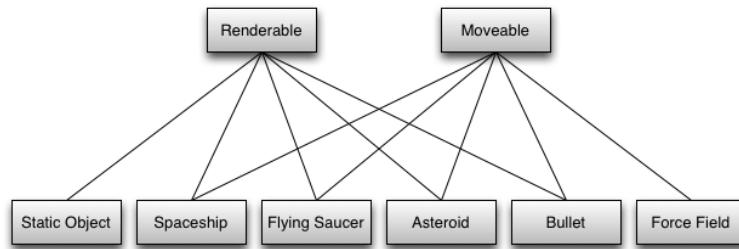
"decouple an abstraction from its implementation so that the two can vary independently"

But wait .. we have **three** hierarchies now!



A fresh start ...

- Stop shuffling around hierarchies ...
- Renderable & Moveable are **not** superclasses
- Renderable & Moveable are simply data structures (**components**)
- The spaceship, asteroids, flying saucers, the bullet are **entities**



Source [Richard Lord - What is an entity framework](http://www.richardlord.net/blog/what-is-an-entity-framework) (<http://www.richardlord.net/blog/what-is-an-entity-framework>)

Components, Entities

```
struct Moveable {  
    Integer x, y  
    Float velocity  
}
```

```
struct Renderable {  
    Bitmap mesh  
}
```

```
struct Entity {  
    UUID id  
}
```

World

Who stores the association between a **component** and an **entity**?

→ the **world**

The world should be considered the "database" of our engine:

- Add new entities
- Query for entities having components

Examples:

- Hello world, give me all entities having a Renderable component
- Hello world, create a new entity having a Moveable component
- Hello world, please update a given entity with a new Renderable component instance

Interfaces/Classes

```
public interface World {  
    Entity newEntity(Component... cs);  
  
    <C extends Component> void createOrUpdate(Entity e, C component);  
  
    <C extends Component> Optional<C> component(Entity e, Class<C> clazz);  
  
    <C extends Component, T> Stream<T> map(Class<C> clazz, BiFunction<Entity, C, T> func);  
}
```

```
public final class Entity {  
    Entity() {}  
}
```

```
public interface Component {}
```

```
public interface Sys {  
    void apply(World w);  
}
```

Systems

Who speaks to the **world**?

→ **systems**

Systems implement the logic for an application. Given the asteroid example we can identify the following systems:

- **InputSystem**

1. Reads input devices (keyboard, mouse, joystick)
2. Updates all entities having a Moveable component

- **RenderSystem**

1. Iterates over all Renderable entities and renders them on the screen

Systems

```
class InputSystem {
    function apply(World w) {
        // get input from joystick
        newX, newY = ...

        w.forEach(Moveable, (entity, movable) -> {
            w.update(entity, new Moveable(newX, newY))
        })
    }
}

class RenderSystem {
    function apply(World w) {
        w.forEach(Renderable, (entity, renderable) -> {
            graphicsCard.display(renderable.mesh)
        })
    }
}
```

The big picture

	Components									
	Position	Sprite	Camera	Animation	Shape	RigidBody	Controls	Enemy	Hero	Bullet
Systems	Red	Red	Red							
		Orange		Orange						
	Yellow				Yellow	Yellow				
	Green						Green			
	Blue							Blue		
	Purple							Dark Blue		Purple
	Purple							Magenta		Purple

Famous ECS engines

The game industry made a big move towards ECS ~14 years ago. Famous ECS engines:



Artemis ECS (<https://github.com/junkdog/artemis-odb>)

PubSe entities?

The screenshot shows a search interface for vehicle entities. On the left, there is a sidebar with filters for 'Fahrzeugzustand' (Neu), 'Marke, Modell, Variante' (Beliebig), 'Erstzulassung' (with dropdowns for 'von' and 'bis'), 'Kilometer' (with dropdowns for 'von' and 'bis'), and 'Preis (€)' (with dropdowns for 'von' and 'bis'). Below these filters is a button labeled '147.394 Treffer' with a right-pointing arrow. At the bottom is a button labeled 'Email search request'. To the right of the sidebar, there are several entity types listed: Condition, MakeModel, FirstRegistration, Mileage, and Price. Each entity type has a corresponding table with fields like 'Kraftstoffart' (Fuel), 'Leistung' (Power), 'Standort' (AmbitAddress), 'Fahrzeugtyp' (AmbitCountry), 'Getriebe' (Category), and 'Transmission'. Each table row includes a 'Beliebig' value and a 'ändern' (change) link.

PubSe components?

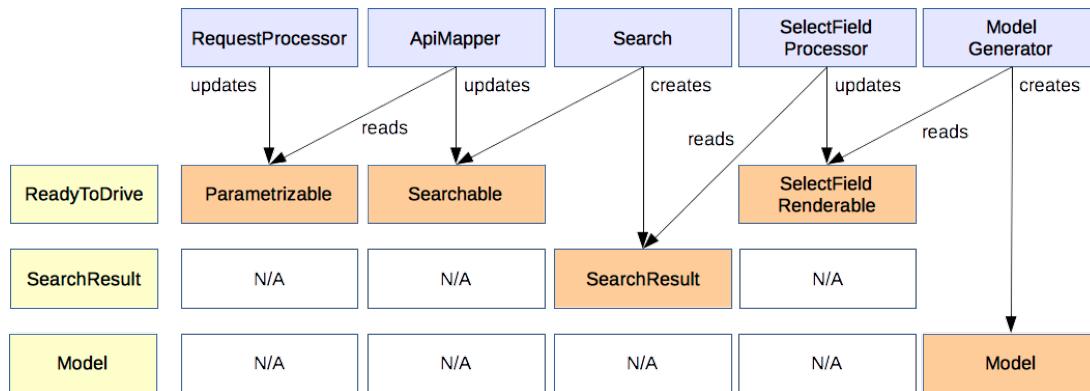
The screenshot illustrates the flow of components in a search application:

- Startseite » Meine Pkw-Suche » 1.422.280 Treffer**: The main search results page.
- mobile.de**: The logo at the top left.
- Suche speichern**: A link in the top right corner.
- The "Condition" entity has the following components:**
 - MagellanRenderable**: Represented by a red arrow pointing to the "Alle Kriterien" button.
 - DutchmanRenderable**: Represented by a red arrow pointing to the "Neu X" button.
 - SelectFieldRenderable**: Represented by a red arrow pointing to the "Gebraucht (1282067)" checkbox.
- 147.387 Treffer - Diese Fahrzeuge entsprechen folgenden Suchkriterien**: The total number of results and a summary statement.
- Fahrzeugzustand**: A dropdown menu showing "Neu (147386)" and "Gebraucht (1282067)".
- Aggregable**: A section on the right side of the search results.
- Filter options**: On the left, there are filters for "Fahrzeugzustand", "Marke, Modell, Variante", "Erstzulassung", and "Kilometer".

PubSe systems?

- **RequestProcessor**: Responsible for parsing the request in flight. Updates Parametrizable components
- **ApiMapper**: Responsible for value mapping for the mportal API. Updates Searchable components
- **Search**: Responsible for invoking the mportal API. Reads Searchable components, Creates an entity with a SearchResult component.
- **SelectFieldProcessor**: Responsible for updating SelectFieldRenderable components.
- **DutchmanProcessor**: Responsible for updating DutchmanRenderable components.
- **MagellanProcessor**: Responsible for updating MagellanRenderable components.

A partial PubSe ECS matrix



A few numbers

In PubSe we have ...

- ~40 systems
- ~50 entities
- ~50 components

A LEGO backend

One can stitch systems and/or worlds differently together.

Examples:

- **Search:** Needs the whole she-bang
- **HitCounter:** Doesn't need aggregations, so leave this system out
- **Tomcat:** Is blocking, thus can have a blocking async system
- **Netty:** Is non-blocking, thus needs a non-blocking async system
- **Logging:** A logging world can trace an event log
- **Tracing:** Every action in the world is recorded and can be inspected in a special trace view

Final thoughts

Peter Norvig:

- Objects are state data with attached behavior
- Closures/Functions are behaviors with attached state data

Rob Pike:

- Sometimes data is just data and functions are just functions

ECS falls into the former category

References

- Hall, Daniel. ECS Game Engine Design (<http://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1138&context=cpesp>)
- <http://www.richardlord.net/blog/what-is-an-entity-framework>
- <http://entity-systems.wikidot.com/>
- <http://t-machine.org/index.php/2007/09/03/entity-systems-are-the-future-of-mmog-development-part-1/>
- <http://vasir.net/blog/game-development/how-to-build-entity-component-system-in-javascript>
- <http://blog.lmorchard.com/2013/11/27/entity-component-system/>

Thank you

Sergiusz Urbaniak



Source dilbert.com (<http://dilbert.com/strip/2000-08-16>)

Thank you

Sergiusz Urbaniak