Chapter 1: Why Domain-Driven Design?
**System 1**
- Fast
- Implicit
- What you see is all there is
- Effortless
- Uncontrolled
- No self-awareness
- Assessing situation
- Delivering updates

**System 2**
- Slow
- Explicit
- Logical and Sceptical
- Effortful
- Deliberately controlled
- Self-awareness
- Getting new information
- Making decisions

---

![Graph](image)
Chapter 2: Language and Context

THE WORLD SEEN BY AN “OBJECT-ORIENTED” PROGRAMMER

Living Space Separation Decorator
Privacy Manager Delegate
Visitor Monitor Interface
Entertainment Provider Singleton
Thirst Quencher Container
Indoor Session Initializer
Multi butt Supporter
Entertainment Provider View Controller

WE ARE GLAD TO FINALLY REACHING AN AGREEMENT!

AHA, BY VISUALIZING OUR IDEAS WE CAME TO A CONCLUSION!
First day not being at work

☐ Came to work but had to leave

When the leave is registered

Came back to work at

☐ Came back to work after lunch

Submit for approval  Close and cancel
Chapter 3: EventStorming

Select a payee → Enter the amount → Confirm the payment → Sign the payment

Customer paid by card

Order confirmed

Payment order created → Payment order signed → Payment order approved → Payer account debited → Payee account credited → Incoming payment recorded → Payment matched with a bill → Bill marked as paid

Time
Time

Classified ad published
Once upon a time

??? Malicious Content detection

Ad submitted for review

Ad approved

Ad published

Ad rejected

Happily ever after
Notation

Domain event:

Hotspot:

External system:

Something happened

Problem!

That's not us
Chapter 4: Designing the Model

Account balance $500 → Deposit in ATM $200 → Account debited for $200 → Account balance $700
Activate Classified Ad

Classified Ad Activated

Classified Ad:
- Title
- Text
- Price
- Images

Activate Classified Ad

Classified Ad Activated

Remove Classified Ad

Classified Ad Removed
Ad created
Ad description updated
Ad renamed
Ad got new category
Ad sell price updated
Picture added to an ad
Picture removed from an ad
Category removed from ad
Ad sent for review
Ad requested to be published
Ad activated
Ad expiration scheduled
Ad marked as sold
Ad removed
Ad rejected
Ad deactivated
Chapter 5: Implementing the Model
JANE
SELLING FOR $100

For sale:
Ikea bed
Model Stockholm,
160 cm wide,
210 cm long.

JOHN
SELLING FOR $80

For sale:
Ikea bed
Model Stockholm,
160 cm wide,
210 cm long.

Generate equality members
Select members to participate in equality operations

- _amount:double

Description
No description available

- Overload equality operators

Comparand type check: Exactly the same type as ‘this’
Title must not be empty

Text must not be empty

Price must not be specified

Ad requested to be published

Publish ad
Create an ad

Ad created

Update ad sell price

Ad sell price updated

Change the ad title

Ad title changed

Update the ad text

Ad text updated

Ad requested to be published

Request to publish the ad

SELLER

USER
Chapter 6: Acting with Commands

Hosting environment: Development
Content root path: /home/alexey/github/ddd-book/Marketplace/bin/Debug/netcoreapp2.1
Now listening on: http://localhost:5000
Application started. Press Ctrl+C to shut down.

ClassifiedAds

ClassifiedAdsCommandsApi

POST /ad

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>string($uid) (query)</td>
</tr>
<tr>
<td>OwnerId</td>
<td>string($uid) (query)</td>
</tr>
</tbody>
</table>

Responses

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
</tbody>
</table>

Response content type: application/json
Chapter 7: Consistency Boundary
Chapter 8: Aggregate Persistence

![Image of DevOps output showing database logs and status messages]

- Built with love by Hibernate Rhinos and awesome contributors!
- Using GC in server concurrent mode retaining memory from the OS.
- Source Code (git repo): https://github.com/ravendb/ravendb
- Server available on: http://7e9156d8b289:8080
- Server started, listening to requests...
- TIP: type 'help' to list the available commands.
- End of standard input detected, switching to server mode...
### ClassifiedAds Commands API

#### POST /ad

#### PUT /ad/name

#### PUT /ad/text

#### PUT /ad/price

#### PUT /ad/publish
### POST /ad

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Id</td>
</tr>
<tr>
<td>OwnerId</td>
</tr>
</tbody>
</table>

**Execute**

**Clear**

**Responses**

Response content type: application/json

**Curl**

```bash
curl -X POST "http://localhost:5000/ad?id=302790d5-735e-445e-a042-b5891ad3cf1f&ownerId=83508629-d2ee-4798-9ac5-b5bbc3a57731" -H "accept: application/json"
```

**Request URL**

http://localhost:5000/ad?id=302790d5-735e-445e-a042-b5891ad3cf1f&ownerId=83508629-d2ee-4798-9ac5-b5bbc3a57731

**Server response**

<table>
<thead>
<tr>
<th>Code</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Error: Bad Request</td>
</tr>
</tbody>
</table>

Response body

```json
{}
```

```csharp
Download
```
PUT /ad/name

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>302790d5-735e-445e-a042-b5891ad3cf1f</td>
</tr>
<tr>
<td>Title</td>
<td>Green sofa</td>
</tr>
</tbody>
</table>

Responses

Curl

curl -X PUT "http://localhost:5000/ad/name?Id=302790d5-735e-445e-a042-b5891ad3cf1f&Title=GreenSofa" -H "accept: application/json"

Request URL

http://localhost:5000/ad/name?Id=302790d5-735e-445e-a042-b5891ad3cf1f&Title=GreenSofa
Chapter 9: CQRS - The Read Side
Chapter 10: Event Sourcing
1. Read
2. When
3. Apply
4. Commit
Event Stream 'ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17'

<table>
<thead>
<tr>
<th>Event #</th>
<th>Name</th>
<th>Type</th>
<th>Created Date</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdPublished</td>
<td>2018-10-18 22:51:02</td>
<td>JSON</td>
</tr>
<tr>
<td>4</td>
<td>4@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdSentForReview</td>
<td>2018-10-18 22:50:50</td>
<td>JSON</td>
</tr>
<tr>
<td>3</td>
<td>3@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdPriceUpdated</td>
<td>2018-10-18 22:50:43</td>
<td>JSON</td>
</tr>
<tr>
<td>2</td>
<td>2@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdTextUpdated</td>
<td>2018-10-18 22:50:25</td>
<td>JSON</td>
</tr>
<tr>
<td>1</td>
<td>1@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdTitleChanged</td>
<td>2018-10-18 22:40:26</td>
<td>JSON</td>
</tr>
<tr>
<td>0</td>
<td>0@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdCreated</td>
<td>2018-10-18 22:40:09</td>
<td>JSON</td>
</tr>
</tbody>
</table>

1@ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17

<table>
<thead>
<tr>
<th>No.</th>
<th>Stream</th>
<th>Type</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ClassifiedAd-556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>ClassifiedAdTitleChanged</td>
<td>2018-10-18 22:40:26</td>
</tr>
</tbody>
</table>

Data

```
{
  "Id": "556bc798-bacc-4bb8-a55b-50144add4f17",
  "Title": "Wooden table"
}
```

Metadata

```
{
  "ClrType": "Marketplace.Domain.ClassifiedAd.Events+ClassifiedAdTitleChanged, Marketplace.Domain, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"
}
```

Internal data

```
EventId 3d94e782-b002-4923-8b79-0ce0d1d5e363
```
Chapter 11: Projections and Queries
Event stream

1. Ad created
   -> Title: empty
   -> Status: inactive

2. Ad renamed
   -> Title: Green sofa
   -> Status: inactive

3. Ad published
   -> Title: empty
   -> Status: published

4. Ad renamed
   -> Title: Red sofa
   -> Status: active

5. Ad deactivated
   -> Title: Red sofa
   -> Status: inactive

"My Ads" projection
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserId</td>
<td>8dd8c5c6-6edf-4e42-ac9e-a232ea445b76</td>
</tr>
<tr>
<td>FullName</td>
<td>Jane Austen</td>
</tr>
<tr>
<td>DisplayName</td>
<td>JustPrejudice</td>
</tr>
</tbody>
</table>
### Event Stream '$all'

<table>
<thead>
<tr>
<th>Event #</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>74777</td>
<td>74777@$stats-0.0.0.0:2113</td>
</tr>
<tr>
<td>74776</td>
<td>74776@$stats-0.0.0.0:2113</td>
</tr>
<tr>
<td>74775</td>
<td>74775@$stats-0.0.0.0:2113</td>
</tr>
<tr>
<td>74774</td>
<td>74774@$stats-0.0.0.0:2113</td>
</tr>
<tr>
<td>735</td>
<td>735@$projections-$master</td>
</tr>
<tr>
<td>734</td>
<td>734@$projections-$master</td>
</tr>
<tr>
<td>733</td>
<td>733@$projections-$master</td>
</tr>
<tr>
<td>732</td>
<td>732@$projections-$master</td>
</tr>
<tr>
<td>0</td>
<td>0@$ce-UserProfile</td>
</tr>
<tr>
<td>0</td>
<td>0@$Set-UserRegistered</td>
</tr>
<tr>
<td>0</td>
<td>0@$Category-UserProfile</td>
</tr>
<tr>
<td>0</td>
<td>0@$Current-UserProfile</td>
</tr>
<tr>
<td>0</td>
<td>0@$Set-UserRegistered</td>
</tr>
</tbody>
</table>
### 0@UpcastedClassifiedAdEvents

<table>
<thead>
<tr>
<th>No</th>
<th>Stream</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UpcastedClassifiedAdEvents</td>
<td>ClassifiedAdPublished</td>
</tr>
</tbody>
</table>

#### Data
```

{  
  "Id": "556bc798-bacc-4bb8-a55b-50144add4f17",  
  "OwnerId": "8dd8c5c6-6ed6-4e42-ac9e-a232ea445b76",  
  "SellersPhotoUrl": "https://www.biography.com/.image/t_share/MTE1ODA0OTcxNzQ2ODcxMzA5/jane-austen-9192819-1-102.jpg"  
  "ApprovedBy": "3d5bdc6f-cb2a-4eae-8099-1991a727d19a"  
}
```

#### Metadata
```

{
  "ClrType": "Marketplace.Projections.ClassifiedAdUpcastedEvents+V1+ClassifiedAdPublished, Marketplace, Version=1.0.0.
}
```

#### Internal data

| EventId | 2ee4ca39-71ef-45ec-8f2a-77bc692d322 |

### Marketplace_Chapter11

#### SELECTION

- New document
- Delete
- Copy

<table>
<thead>
<tr>
<th>Id</th>
<th>Change Vector</th>
<th>Last Modified</th>
<th>Collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>readmodels</td>
<td>A:18</td>
<td>2018 December 2nd, 9:46 PM</td>
<td>Checkpoints</td>
</tr>
<tr>
<td>556bc798-bacc-4bb8-a55b-50144add4f17</td>
<td>A:16</td>
<td>2018 December 2nd, 9:46 PM</td>
<td>ClassifiedAdDetails</td>
</tr>
<tr>
<td>8dd8c5c6-6ed6-4e42-ac9e-a232ea445b76</td>
<td>A:12</td>
<td>2018 December 2nd, 9:46 PM</td>
<td>UserDetails</td>
</tr>
</tbody>
</table>
Chapter 12: Bounded Context

**Diagram:**

- **Product**
  - + ProductId
  - + Description
  - + Price

- **Order**
  - + OrderId
  - + CustomerId
  - + Status

- **Customer**
  - + CustomerId
  - + FullName
  - + PhoneNumber

- **OrderLine**
  - + OrderLineId
  - + OrderId
  - + ProductId
  - + Quantity
Ad title:
Green sofa

Description:
Solid sofa of green color.
Used, but still in good condition.

Images:

Add an image

Price:
200

Next step
Ad created

Ad renamed

Ad description updated

Picture added to an ad

Ad sell price updated
User | User | User | Admin
---|---|---|---
Confirm order | Submit payment | Send ad for review (publish) | Approve ad
Order confirmed | Payment approved | Ad sent for review | Ad approved

Awesome Marketplace
http://marketplace.whatever/ad/3423442ad665/#

Your classified ad will look like this when published:

![Green sofa](image)

200,-

Get more attention, choose one of the options:
- Top of the search results for 7 days (€5)
- Top of the search results for 14 days (€8)
- I don't want more visibility for my ad (free)

[Proceed]
New Classified Ad
Fill out the details and then click the Add button

<table>
<thead>
<tr>
<th>Title</th>
<th>Green sofa</th>
</tr>
</thead>
</table>

| Description | Selling my beautiful sofa, which I spent a lot of time sitting and thinking before I start to write! Excellent condition, but used a lot. |

| Price | 260 |
Learn to solve complex business problems by understanding users better, finding the right problem to solve, and building lean event-driven systems to give your customers what they really want.

- eBook: €29.99
- Printed: €37.99
- In stock
- Order now and get it in three days
Learn to solve complex business problems by understanding users better, finding the right problem to solve, and building lean event-driven systems to give your customers what they really want.

Composition API (Backend For Frontend)

- API
  - Recommendation
- API
  - Pricing
- API
  - Inventory
- API
  - Book catalog
Learn to solve complex business problems by understanding users better, finding the right problem to solve, and building lean event-driven systems to give your customers what they really want.