D IS FOR
(DE)SERIALIZATION

STEVEN SCHVEIGHOFFER
PAYROLL!!!
PAYROLL!!!

- In the fall of 2016, our HR manager gave her notice.
- Needed someone to handle Payroll
- Who could do such a task?
PAYROLL MAN!
2 + 2 = 4
PAYROLL MAN!

2 + 2 = 4
STEPS FOR PAYROLL
1. PAYROLL TIMESHEET

- Print one at the start of every payroll
- Sorted intuitively by department
- Record hours
- Record changes to status
<table>
<thead>
<tr>
<th>PERSONNEL INFORMATION</th>
<th>RATE/HOUR</th>
<th>REGULAR HOURS</th>
<th>OVERTIME HOURS</th>
<th>OTHER (Indicate Hours or $)</th>
<th>EARNINGS (G) / DEDUCTIONS (D)</th>
<th>YEAR-TO-DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wks Worked 46</td>
<td>23.1000/hour</td>
<td>40</td>
<td>9.07</td>
<td>PTO - 84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bi-weekly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Hours Used: S 0.00V 0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YTD Wages &amp; Tips: 46,654.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**
- **Terminate:**
  - Check Salary!!

**Compl:** 1349.05

**Compl:** 185.55
2. INDIVIDUAL TIMESHEETS

- Time started and stopped each task
- Hours for each task
- Total hours for day
- Drive time (am/pm)
- Total hours for week (regular and overtime)
- Overnights
Payroll Time Record for Period: 1/8/2017 Thru 1/14/2017 Pay Period: 1 Week: 2

<table>
<thead>
<tr>
<th>Sunday 1/8/17</th>
<th>Extra Drive Job 1</th>
<th>Leave Job 1</th>
<th>Hours Job 1</th>
<th>Start Job 2</th>
<th>Leave Job 2</th>
<th>Hours Job 2</th>
<th>Start Job 3</th>
<th>Leave Job 3</th>
<th>Hours Job 3</th>
<th>Start Job 4</th>
<th>Leave Job 4</th>
<th>Hours Job 4</th>
<th>Extra Drive Time PM</th>
<th>Straight Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday 1/9/17</td>
<td>8:00 Job 1#</td>
<td>10:00 Job 2#</td>
<td>9:00 Job 3#</td>
<td>7:30 Job 4#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5 reg</td>
</tr>
<tr>
<td>Tuesday 1/10/17</td>
<td>7:30 Job 1#</td>
<td>8:30 Job 2#</td>
<td>9:00 Job 3#</td>
<td>3:30 Job 4#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7.5 reg</td>
</tr>
<tr>
<td>Wednesday 1/11/17</td>
<td>8:00 Job 1#</td>
<td>10:30 Job 2#</td>
<td>11:00 Job 3#</td>
<td>6:50 Job 4#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8 reg</td>
</tr>
<tr>
<td>Thursday 1/12/17</td>
<td>8:00 Job 1#</td>
<td>2:00 Job 2#</td>
<td>2:30 Job 3#</td>
<td>4:30 Job 4#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5 reg</td>
</tr>
<tr>
<td>Friday 1/13/17</td>
<td>8:00 Job 1#</td>
<td>5:30 Job 2#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 reg</td>
</tr>
<tr>
<td>Saturday 1/14/17</td>
<td>Job 1#</td>
<td>2:55 Job 2#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of Overnights: 1
On Call?: No

Week Regular Hours: 39.5
Week PW Hours: 0
Total PTO Hours: 0

Print Name: [Signature]

Email time sheets to imathews@nrminc.com and copy your supervisor.
All hours must be assigned to a Job # or WO #. Other time assigned as follows:
99=Drive Time / 85=Warehouse / 44=Office
# Payroll Time Record for Period: 1/29/2017 Thru 2/4/2017

## Pay Period: 3

### Week: 1

<table>
<thead>
<tr>
<th>Day</th>
<th>Start Time AM</th>
<th>Leave Time AM</th>
<th>Hours AM</th>
<th>Start Time PM</th>
<th>Leave Time PM</th>
<th>Hours PM</th>
<th>Extra Drive Time PM</th>
<th>Straight Time</th>
<th>Over Time Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/29/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/30/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/31/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/1/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/2/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/3/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Saturday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/4/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Post Name: [Signature]

Number of Overtime: (4)

Week Regular Hours: 40

Week PW Hours: 10

Total PTO Hours: 0

Total Holiday Hours: 0

Email time sheets to lmatthews@nrminc.com and copy your supervisor.

All hours must be assigned to a Job # or WO #. Other time assigned as follows:

99=Drive Time / 55=Warehouse / 44=Office

Date: 2/4/17

On Call? (4)

Supervisor Signature: [Signature]
3. PUNCHCARDS

- Office employees use RFID punchcards
- But must also manually write them down
- Shore up punch card log with written data
5. ENTER DATA

- Copy all data from the payroll timesheet to the web site
- Double check all numbers by adding on a calculator!
- Adjust anything that needs adjusting (salary, tax withholdings, etc)
6. OTHER STUFF

- Commissions
- Time off report
- Run post-payroll reports
- Print everything, put in a folder
6. OTHER STUFF

- Separate folder for each PW job
- Extract 401k report from payroll system
- Update 401k CSV file... BY HAND
- 401k loan updates
6. OH GOD IT KEEPS GOING

- Print 401k report for separate folder
- Send commission sheets to salespeople
- Certified payroll
- Print out timesheet for next payroll
TIME NEEDED TO PREPARE PAYROLL

3-4 days

TIME AVAILABLE TO PREPARE PAYROLL

2 days
THE MORAL

"You have a computer, why not use it?"

Use computers for everything they are good at
TIMECARD APP

- No more time math
- Always have it with them
- Links to jobs in our tracking system
- Replace unreliable punchcard system
PAYROLL TIMESHEET

- Use Excel to track everything instead of paper
- VBA macro to generate CSV for upload to payroll system
- No more clunky transfer, one button push
- All totals calculated instantly.
TIME TO RUN PAYROLL NOW?

4 hours
D SERIALIZATION

You have a computer, why not use it?
D SERIALIZATION

You have a computer D Compiler, why not use it?
A Simple Example

```c
import db.api;

struct Record
{
    int id;
    string name;
}

Record processRecord(Row r)
{
    Record result;
    result.id = r.getInteger("id");
    result.name = r.getString("mane");
    return result;
}
```
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("mane");
    return result;
}
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("mane");
    return result;
}
A Simple Example

```python
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("mane");
    return result;
}
```

Parse integer field
A Simple Example

```python
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("mane");
    return result;
}

Parse string field
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("name");
    return result;
}
A Simple Example

```c
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("mane");
    return result;
}
```

Does it work?
A GENERATIVE APPROACH

- static foreach
- __traits/std.traits
- static if
A Generative Approach

```javascript
import db.api;

struct Record
{
  int id;
  string name;
}

Record processRecord(Row r)
{
  Record result;
  import std.traits;
  static foreach(i, m; FieldNameTuple!Record)
  {
    __traits(getMember, result, m) =
      r.getType!(typename->__traits(getMember, result, m))(m);
  }
}
```
A Generative Approach

```plaintext
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record) {
        __traits(getMember, result, m) =
        r.getWithTyp!(typeof(__traits(getMember, result, m)))(m);
    }
}
```

Same definition
A Generative Approach

```plaintext
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record) {
        __traits(getMember, result, m) = r.getWith_type!(typeof(__traits(getMember, result, m)))(m);
    }
}
```

Initialize Record
A Generative Approach

```c
class Record
{
    int id;
    string name;
}

Record processRecord(Row r)
{
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record)
    {
        __traits(getMember, result, m) =
            r俾WithTye!(typeof(__traits(getMember, result, m)))(m);
    }
}
```

Import some generative magic
A Generative Approach

```cpp
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record) {
        __traits(getMember, result, m) = r.getType!(typeof(__traits(getMember, result, m)))(m);
    }
}

Get all the field member names
```
A Generative Approach

```cpp
struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record) {
        __traits(getMember, result, m) = r.getWithType!(typeof(__traits(getMember, result, m)))(m);
    }
}
```

Assign the field to...
A Generative Approach

```cpp
import db.api;

struct Record {
  int id;
  string name;
}

Record processRecord(Row r) {
  Record result;
  import std.traits;
  static foreach(i, m; FieldNameTuple!Record) {
    __traits(getMember, result, m) = r.getType!(typeof(__traits(getMember, result, m)))(m);
  }
}
```

The exact conversion expected
A Generative Approach

```rust
import db.api;

struct Record {
    int id;
    string name;
}

Record processRecord(Row r) {
    Record result;
    import std.traits;
    static foreach(i, m; FieldNameTuple!Record) {
        __traits(getMember, result, m) =
            r.getWithTypemap!(typeof(__traits(getMember, result, m)))(m);
    }
}
```

The whole function
A More complicated case

```python
import db.api;

enum RecordType
{
    person,
    location
}

enum Permission
{
    admin = 0x01,
    restricted = 0x02,
    guest = 0x04,
}

struct Point
{
    int x;
}
A More complicated case

```
import db.api;

enum RecordType
{
    person,
    location
}

enum Permission
{
    admin = 0x01,
    restricted = 0x02,
    guest = 0x04,
}

struct Point
{
    int x,
    int y;
}
```
A More complicated case

```cpp
enum Permission {
    admin = 0x01,
    restricted = 0x02,
    guest = 0x04,
};

struct Point {
    int y;
    int x;
}

struct Record {
    int id;
    string name;
    RecordType type; // stored as integer
};

 Enums and Types
A More complicated case

```c
struct Record
{
    int id;
    string name;
    RecordType type; // stored as integer
    Point location; // location_x, location_y, only present if type is location
    string comment; // optional
    Permission permissions; // stored as comma-separated string list
    int _class; // really called "class" in the DB
}
```

More complex, more convenient
A More complicated case

```java
String name,
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location
string comment; // optional
Permission permissions; // stored as comma-separated string list
int _class; // really called "class" in the DB
}

Record processRecord(Row r)
{
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("name"); // typo corrected!
    // Record type is stored as an integer, so use a direct cast
    result.type = cast(RecordType)r.getAsInteger("type");
    // location is stored as 2 fields, location_x, and location_y
    if(type == RecordType.location)
    
        location = Point(r.getAsInteger("location_x"),
```

Declare the result
A More complicated case

```java
struct Record {
    int id;
    string name;
    RecordType type; // stored as integer
    Point location; // location_x, location_y, only present if type is location
    string comment; // optional
    Permission permissions; // stored as comma-separated string list
    int _class; // really called "class" in the DB
}

Record processRecord(Row r) {
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("name"); // typo corrected!
    // Record type is stored as an integer, so use a direct cast
    result.type = cast(RecordType)r.getAsInteger("type");
}
```

Same as before
A More complicated case

```java
int id;
string name;
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location
string comment; // optional
Permission permissions; // stored as comma-separated string list
int _class; // really called "class" in the DB

Record processRecord(Row r)
{
    Record result;
    result.id = r.getAsInteger("id");
    result.name = r.getAsString("name"); // typo corrected!
    // Record type is stored as an integer, so use a direct cast
    result.type = cast(RecordType)r.getAsInteger("type");
    // location is stored as 2 fields, location_x, and location_y
    // (type = RecordType.location)
}
```

Need to parse this as an integer
A More complicated case

```java
int id;
string name;
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location ...

Record processRecord(Row r)
{
...

// Record type is stored as an integer, so use a direct cast
result.type = cast(RecordType)r.getInteger("type");
// location is stored as 2 fields, location_x, and location_y
if(type == RecordType.location)
    location = Point(r.getInteger("location_x"),
                     r.getInteger("location_y"));
// comment is optional, avoid exception for NULL storage
```
A More complicated case

```java
string name;
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location
string comment; // optional
...
```

Record processRecord(Row r)
{
    ...
    if(type == RecordType.location)
    {
        location = Point(r.getAsInteger("location_x"),
                        r.getAsInteger("location_y"));
        // comment is optional, avoid exception for NULL storage
        if(!r.fieldIsNull("comment"))
            result.comment = r.getAsString("comment");
```

Only store comment if present
A More complicated case

```java
string name;
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location
string comment; // optional
Permission permissions; // stored as comma-separated string list
int _class; // really called "class" in the DB
}

Record processRecord(Row r)
{
...  
// permissions is stored as a comma-separated string list.
result.permissions = r.getAsString("permissions")
    .splitter(\',\')
    .map(s => s.to!Permission)
    .reduce!((a, b) => cast(Permission)(a|b));
result._class = r.getAsInteger("class");
```

Comma-seperated map parsing
A More complicated case

```java
string name;
RecordType type; // stored as integer
Point location; // location_x, location_y, only present if type is location
string comment; // optional
Permission permissions; // stored as comma-separated string list
int _class; // really called "class" in the DB
}

Record processRecord(Row r)
{
...
    // permissions is stored as a comma-separated string list.
    result.permissions = r.getAsString("permissions")
        .splitter(',')
        .map(s => s.to!Permission)
        .reduce((a, b) => cast(Permission)(a|b));
    result._class = r.getAsInteger("class");
}

avoid keyword collision
result.name = r.getAsString("name"); // typo corrected!

// Record type is stored as an integer, so use a direct cast
result.type = cast(RecordType)r.getAsInteger("type");

// location is stored as 2 fields, location_x, and location_y
if(type == RecordType.Location)
    location = Point(r.getAsInteger("location_x"),
                    r.getAsInteger("location_y"));

// comment is optional, avoid exception for NULL storage
if(r.hasField("comment") && !r.fieldIsNull("comment"))
    result.comment = r.getAsString("comment");

// permissions is stored as a comma-separated string list.
result.permissions = r.getAsString("permissions")
    .splitter(',')
    .map(s => s.to!Permission)
    .reduce((a, b) => cast(Permission)(a|b));
result._class = r.getAsInteger("class");

Can you spot the bug?
struct Point
{
    int y;
    int x;
}

Record processRecord(Row r)
{
    ...
    // location is stored as 2 fields, location_x, and location_y
    if(type == RecordType.location)
        location = Point(r.getAsInteger("location_x"),
                         r.getAsInteger("location_y"));
    // comment is optional: avoid exception for NULL storage

How about now?
MORE GENERATIVE APPROACH

- User Data Attributes (UDAs) instead of comments
- Compiler can react to them, just like we react to the comments.
Replace comments with UDAs

```c
struct Record {
    int id;
    string name;
    RecordType type; // stored as integer
    Point location; // stored as location_x, location_y, only
    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
}
```
Replace comments with UDAs

```c
struct Record {
    int id;
    string name;
    RecordType type; // stored as integer
    Point location; // stored as location_x, location_y, only
    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
}
```
Replace comments with UDAs

```c
struct Record
{
    int id;
    string name;

    @dbType!int
    RecordType type;

    Point location; // stored as location_x, location_y, only
    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
}
```

You can pass types to UDAs
Replace comments with UDAs

```c
struct Record {
    int id;
    string name;

    @dbType!int
    RecordType type;

    Point location; // stored as location_x, location_y, only
    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
};
```
Replace comments with UDAs

```c
struct Record {
    int id;
    string name;

    @dbType!int
    RecordType type;

    @dbOnlyIf!(R => r.type == RecordType.location)
    Point location;

    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
}
```

And lambdas
Replace comments with UDAs

```csharp
struct Record
{
    int id;
    string name;

    @dbType!int
    RecordType type;

    @dbOnlyIf!(R => r.type == RecordType.location)
    Point location;

    string comment; // optional
    Permission permissions; // stored as comma-separated string
    int _class; // really called "class" in the DB
}
```
Replace comments with UDAs

```csharp
int id;
string name;

@dbType!int
RecordType type;

@dbOnlyIf!(R => r.type == RecordType.location)
Point location;

@dbOptional
string comment;

Permission permissions; // stored as comma-separated string
int _class; // really called "class" in the DB
```
Replace comments with UDAs

```java
int id;
string name;

@dbType!int
RecordType type;

@dbOnlyIf!(R => r.type == RecordType.location)
Point location;

@dbOptional
string comment;

Permission permissions; // stored as comma-separated string
int _class; // really called "class" in the DB
```
Replace comments with UDAs

@dbType!int
RecordType type;

@dbOnlyIf!(R => r.type == RecordType.location)
Point location;

@dbOptional
string comment;

@dbProcessWith!(parseCommaList)
Permission permissions;

int _class; // really called "class" in the DB

Pass a normal function
Replace comments with UDAs

```csharp
@dbType!int
RecordType type;

@dbOnlyIf!(R => r.type == RecordType.location)
Point location;

@dbOptional
string comment;

@dbProcessWith!(parseCommaList)
Permission permissions;

int _class; // really called "class" in the DB
```
Replace comments with UDAs

@dbType!int
RecordType type;

@dbOnlyIf!(R => r.type == RecordType.location)
Point location;

@dbOptional
string comment;

@dbProcessWith!(parseCommaList)
Permission permissions;

@dbName("class")
int _class;

and so on...
Replace comments with UDAs

```csharp
struct Record
{
    int id;
    string name;

    @dbType:int
    RecordType type;

    @dbOnlyIf!(R => r.type == RecordType.location)
    Point location;

    @dbOptional
    string comment;

    @dbProcessWith!(parseCommaList)
    Permission permissions;

    @dbName("class")
    int _class;
}
```

The result
Generative serialization

Record processRecord(Row r)
{
    import std.traits;

    Record result;
    foreach(m; __traits(allMembers, Record))
    {
        static if(hasUDA!(__traits(getMember, Record, m), DbOptional))
        {
            if(!r.hasField(m) || r.fieldIsNull(m))
                continue;
        }
        static if(hasUDA!(__traits(getMember, Record, m), DbOnlyIf))
        {
            if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
                continue;
        }
        static if(hasUDA!(__traits(getMember, Record, m), DbPresenceOnly))
        {
            // Additional code for presence only
        }
    }
}
Generative serialization

```plaintext
Record processRecord(Row r)
{
    import std.traits;

    Record result;
    foreach(m; __traits(allMembers, Record))
    {
        static if(hasUDA!(__traits(getMember, Record, m), DbOptional))
        {
            if(!r.hasField(m) || r.fieldIsNull(m))
                continue;
        }
        static if(hasUDA!(__traits(getMember, Record, m), DbOnlyIf))
        {
            if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
                continue;
        }
    }
}
```

Begins the same as our other functions
Generative serialization

Record processRecord(Row r)
{
    import std.traits;

    Record result;
    foreach(m; __traits(allMembers, Record))
    {
        static if(hasUDA!(__traits(getMember, Record, m), DbOptional))
        {
            if(!r.hasField(m) || r.fieldIsNull(m))
                continue;
        }
        static if(hasUDA!(__traits(getMember, Record, m), DbOnlyIf))
        {
            if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
                continue;
        }
    }
}

use NORMAL foreach?
Generative serialization

```rust
import std.traits;

Record result;
foreach(m; __traits(allMembers, Record))
{
    static if(hasUDA!(__traits(getMember, Record, m), DbOptional))
    {
        if(!r.hasField(m) || r.fieldIsNull(m))
            continue;
    }
    static if(hasUDA!(__traits(getMember, Record, m), DbOnlyIf))
    {
        if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
            continue;
    }
    static if(hasUDA!(__traits(getMember, Record, m), DbProcessWith))
}
```

skip members if they are optional and not present
Generative serialization

```c

{ static if(hasUDA!(__traits(getMember, Record, m), DbOptional))
  {
    if(!r.hasField(m) || r.fieldIsNull(m))
      continue;
  }
  static if(hasUDA!(__traits(getMember, Record, m), DbOnlyIf))
  {
    if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
      continue;
  }
  static if(hasUDA!(__traits(getMember, Record, m), DbProcessWith))
  {
    __traits(getMember, Record, m) =
    getUDA!(__traits(getMember, Record, m), DbProcessWith).processor(r)
  }
  else

skip members if some predicate says we should

```
Generative serialization

```c
static if(hasUDA(!__traits(getMember, Record, m), DbType))
{
    alias TypeToRead =
        getUDA!(__traits(getMember, Record, m), DbType).Type;
}
```

custom function to process the data from the row
Generative serialization

```cpp
if(!getUDA!(__traits(getMember, Record, m), DbOnlyIf).pred(result))
    continue;
}
static if(hasUDA!(__traits(getMember, Record, m), DbProcessWith))
{
    __traits(getMember, Record, m) =
    getUDA!(__traits(getMember, Record, m), DbProcessWith).processor(r))
} else
{
    alias mType = typeof(__traits(getMember, Record, m));
    static if(hasUDA!(__traits(getMember, Record, m), DbType))
        alias TypeToRead =
        getUDA!(__traits(getMember, Record, m), DbType).Type;
    else
        alias TypeToRead = mType;
    static if(hasUDA!(__traits(getMember, Record, m), DbName))
}

Otherwise, we are going to read it normally, but...
```
Generative serialization

```cpp
{  
  _traits(getMember, Record, m) =  
    getUDA!(__traits(getMember, Record, m), DbProcessWith).processor(r))
}
else  
{
  alias mType = typeof(__traits(getMember, Record, m));
  static if(hasUDA!(__traits(getMember, Record, m), DbType))  
    alias TypeToRead =  
      getUDA!(__traits(getMember, Record, m), DbType).Type;  
  else  
    alias TypeToRead = mType;
  static if(hasUDA!(__traits(getMember, Record, m), DbName))  
    enum fieldName =  
      getUDA!(__traits(getMember, Record, m), DbName).name;  
  else  
    enum fieldName = m;
}
```

Use the correct type for reading
Generative serialization

```cpp
alias mType = typeof(__traits(getMember, Record, m));
static if(hasUDA!(__traits(getMember, Record, m), DbType))
    alias TypeToRead =
        getUDA!(__traits(getMember, Record, m), DbType).Type;
else
    alias TypeToRead = mType;
static if(hasUDA!(__traits(getMember, Record, m), DbName))
    enum fieldName =
        getUDA!(__traits(getMember, Record, m), DbName).name;
else
    enum fieldName = m;
__traits(getMember, Record, m) = cast(mType)
    r.getWithTypeNameTypeToRead(fieldName);
}
}
return result;
```

Check for an override of the name in the DB
Generative serialization

```c++
static if(hasUDA!(__traits(getMember, Record, m), DbType)) {
    alias TypeToRead =
        getUDA!(__traits(getMember, Record, m), DbType).Type;
} else {
    alias TypeToRead = mType;
}
static if(hasUDA!(__traits(getMember, Record, m), DbName)) {
    enum fieldName =
        getUDA!(__traits(getMember, Record, m), DbName).name;
} else {
    enum fieldName = m;
}
__traits(getMember, Record, m) = cast(mType)
    r.getType!TypeToRead(fieldName);
return result;
}
```

Do the actual read
Generative serialization

```cpp
static if(hasUDA!(__traits(getMember, Record, m), DbType))
    alias TypeToRead =
        getUDA!(__traits(getMember, Record, m), DbType).Type;
else
    alias TypeToRead = mType;
static if(hasUDA!(__traits(getMember, Record, m), DbName))
    enum fieldName =
        getUDA!(__traits(getMember, Record, m), DbName).name;
else
    enum fieldName = m;
__traits(getMember, Record, m) = cast(mType)
    r.getType!TypeToRead(fieldName);
}

return result;
```
GENERATIVE VS. MANUAL

- 41 lines (Generative) vs. 21 (Manual)
- Generative version DRYer than Manual
- Generative version does not need updating when struct updates
- Generative tends to compile if correct, manual bugs are more subtle.
MIXINS AND CTFE

- String manipulation in D is strong, and mostly available for CTFE
- You truly can generate code!
- More difficult to follow
- More difficult to debug
- Allows for modular usage
OTHER LANGUAGES
JAVA

- Statically compiled
- Supports generics, but not templates
- Strong Runtime Type Info
public interface Serializable {}

- Java uses reflection for introspection
- Serializable interface is just a tag
- There are some hooks, but not statically defined.
- Optimization must rely on JIT
C/C++

• Generative template capabilities
• But not as useful introspection capabilities!
• No good way to loop through members
class Archiveable {
    int x;
    template<class Archive>
    void save(Archive & ar, const unsigned int version) const {
        ar & x;
    }
    template<class Archive>
    void load(Archive & ar, const unsigned int version) {
        ar & x;
    }
};

TRADITIONAL SERIALIZATION
TRADITIONAL SERIALIZATION

```cpp
class Archiveable {
    int x;
    template<class Archive>
    void save(Archive & ar, const unsigned int version) const
    { ar & x; }
    template<class Archive>
    void load(Archive & ar, const unsigned int version) {
        ar & x;
    }
};
```

- Abstracts what the archiver does
- Focused more on supporting multiple formats than eliminating boilerplate
- You still define how to serialize all the members
WHAT ABOUT C?

- No introspection capabilities, but has preprocessor!
- Most projects focused on serialization format, not ease of use
IOPIPE AND JSON SERIALIZATION
Usage for tokenizer

```python
import iopipe.json.parser;

void main()
{
    auto message = `"id" : 1, "name": "Steve"`;  
    auto parser = message.jsonTokenizer!(false);
    auto jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.ObjectStart);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.String);
    assert(jsonItem.data(parser.chain) == "id");
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Colon);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Number);
}`
Usage for tokenizer

```cpp
import io.pipe.json.parser;

void main()
{
    auto message = `"{"id": 1, "name": "Steve"}`;
    auto parser = message.jsonTokenizer!(false);
    auto jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.ObjectStart);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.String);
    jsonItem = parser.next;
    assert(jsonItem.data(parser.chain) == "id");
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Colon);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Number);
}
```

Declare a message (input io.pipe)
Usage for tokenizer

```python
import iopipe.json.parser;

void main()
{
    auto message = `"id" : 1, "name": "Steve"`;
    auto parser = message.jsonTokenizer!(false);
    auto jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.ObjectStart);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.String);
    assert(jsonItem.data(parser.chain) == "id");
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Colon);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Number);
}
```

Create a tokenizer for that iopipe
import iopipe.json.parser;

void main()
{
  auto message = `"id" : 1, "name": "Steve"`;
  auto parser = message.jsonTokenizer!(false);
  auto jsonItem = parser.next;
  assert(jsonItem.token == JSONToken.ObjectStart);
  jsonItem = parser.next;
  assert(jsonItem.token == JSONToken.String);
  assert(jsonItem.data(parser.chain) == "id");
  jsonItem = parser.next;
  assert(jsonItem.token == JSONToken.Colon);
  jsonItem = parser.next;
  assert(jsonItem.token == JSONToken.Number);
}

The object starts!
void main()
{
    auto message = `"id" : 1, "name" : "Steve"`;
    auto parser = message.jsonTokenizer!(false);
    auto jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.ObjectStart);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.String);
    assert(jsonItem.data(parser.chain) == "id");
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Colon);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Number);
    assert(jsonItem.data(parser.chain) == "1");
}

Usage for tokenizer

First member and colon
Usage for tokenizer

```cpp
void main()
{
    auto message = `{"id" : 1, "name": "Steve"}`;
    auto parser = message.jsonTokenizer!(false);
    auto jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.ObjectStart);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.String);
    jsonItem = parser.next;
    assert(jsonItem.token == JSONToken.Colon);
// and so on...
}
```

The first member's value
Usage for tokenizer

```cpp
auto message = `{"id" : 1, "name": "Steve"}`;
auto parser = message.jsonTokenizer!(false);
auto jsonItem = parser.next;
assert(jsonItem.token == JSONToken.ObjectStart);
jsonItem = parser.next;
assert(jsonItem.token == JSONToken.String);
assert(jsonItem.data(parser.chain) == "id");
jsonItem = parser.next;
assert(jsonItem.token == JSONToken.Colon);
jsonItem = parser.next;
assert(jsonItem.token == JSONToken.Number);
assert(jsonItem.data(parser.chain) == "1");
// and so on...
```
DESERIALIZATION

- Use template overloading to select how to deserialize
- **Primitive types:** easy, just use to
- **Arrays:** loop and recurse
- **Objects:** loop and recurse, using D introspection techniques!
Some UDA definitions for specialized behavior
Handle "Roll-your-own"
if (Т == struct) && __traits(hasMember, T, "fromJSON")
{
    item.fromJSON(tokenizer, relPol);
}

private void
deserializeImpl(T, JT)(ref JT tokenizer, ref T item, ReleasePolicy relPol)
if (is(T == struct) && __traits(hasMember, T, "fromJSON")
{
    // check to see if any member is defined as the representation
    import std.traits;
    alias representers = getSymbolsByUDA!(T, SerializeAs());
    static if(representers.length > 0) {
        static assert(representers.length == 1, "Only one field can be used to represent an object.

Function to handle standard structs
Check for structs that are parsed into one member
Otherwise, get the list of members we will serialize
Filter out ignored members
Assume optionals are visited already
static if(hasUDA!(__traits(getMember, T, m), Optional()))
    visited[idx] = true;

auto jsonItem = tokenizer.next;
jsonExpect(jsonItem, JSONToken.ObjectStart, "Parsing " ~ T.stringof);

// look at each member name, then parse the given values
jsonItem = tokenizer.next();
static if(members.length == 0)
{
    // no members, expect an object end
    jsonExpect(jsonItem, JSONToken.ObjectEnd, "Expecting end of memberless
}
else
{
    while(jsonItem.token != JSONToken.ObjectEnd)
    {

Special case no members
Standard parsing fare
jsonExpect(jsonItem, JSONToken.Colon, "Expecting colon when parsing " ~ T.stringo

OBJ_MEMBER_SWITCH:
    switch(name)
    {
        static foreach(i, m; members)
        {
            case m:
                tokenizer.deserializeImpl(__traits(getMember, item, m),
                relPol);
                visited[i] = true;
                break OBJ_MEMBER_SWITCH;
        }

        default:
            import std.format : format;
            throw new Exception(format("No member named '%s' in type '%s'

        }
    }

if(relPol == ReleasePolicy.afterMembers)

The generative part
Verify all non optional members were found
EXPOSING A D API ON THE WEB

- Use all available information
- Map names intuitively
- Where custom behavior is needed, use UDAs
Encapsulate the API in a class

```javascript
import vibe.vibe;
import vibe.web.auth;

@path("/wh")
class WebHandler
{
    string[string] items;

    void getIndex(HTTPServerResponse res)
    {
        auto items = this.items;
        res.render!("index.dt", items);
    }

    void postItem(HTTPServerRequest req, string name, string value,
```
Encapsulate the API in a class

```java
import vibe.vibe;
import vibe.web.auth;

@path("/wh")
class WebHandler
{
    string[string] items;

    void getIndex(HTTPServerResponse res)
    {
        auto items = this.items;
        res.render!("index.dt", items);
    }

    void postItem(HTTPServerRequest req, string name, string value,
```
Encapsulate the API in a class

```cpp
import vibe.vibe;
import vibe.web.auth;

@path("/wh")
class WebHandler
{
    string[string] items;

    void getIndex(HTTPServerResponse res)
    {
        auto items = this.items;
        res.render!("index.dt", items);
    }

    void postItem(HTTPServerRequest req, string name, string value,
```
Encapsulate the API in a class

```csharp
@path("/wh")
class WebHandler
{
    string[string] items;

    void getIndex(HTTPServerResponse res)
    {
        auto items = this.items;
        res.render!("index.dt", items);
    }

    void postItem(HTTPServerRequest req, string name, string value,
    HTTPServerResponse res)
    {
        items[name] = value;
    }
}
```

Simple get route
Encapsulate the API in a class

```cpp
auto items = this.items;
res.render!("index.dt", items);

void postItem(HTTPServerRequest req, string name, string value, HTTPServerResponse res)
{
    items[name] = value;
    res.redirect("/wh/index");
}

@path("item/:name")
void getItem(string _name, HTTPServerResponse res)
{

```
Encapsulate the API in a class

```plaintext
items[name] = value;
res.redirect("/wh/index");

@path("item/:name")
void getItem(string _name, HTTPServerResponse res)
{
    if (auto v = _name in items)
        res.writeBody("<html><body>Item named " ~ _name ~ " is "
                     ~ v ~ "<br><a href='/wh/index'>Home</a></body></html>",
                      "text/html");
    else
        res.redirect("/wh/index");
}
```

Retrieve an item
Add authentication

```c
import vibe.vibe;
import vibe.web.auth;

struct AuthInfo
{
    string username;

    @safe:
    bool isEditor() {
        return username == "root";
    }

    bool isViewer(string _name) {
        return _name.length > 0 && _name[0] == 'a';
    }
}
```
Add authentication

```c++
import vibe.web.auth;

struct AuthInfo
{
    string username;
@safef
    bool isEditor()
    {
        return username == "root";
    }

    bool isViewer(string _name)
    {
        return _name.length > 0 && _name[0] == 'a';
    }
};
```

The authorization state
Add authentication

```cpp
bool isViewer(string _name) {
    return _name.length > 0 && _name[0] == 'a';
}

@Path("/wh2")
@requiresAuth
class WebHandler2 {

    string[string] items;

    @noAuth
    void getIndex(HTTPServerResponse res) {
```

Now, our class requires authorization
Add authentication

```javascript
@safe @noRoute AuthInfo
    authenticate(scope HTTPServerRequest req,
                  scope HTTPServerResponse res)
{
    // terrible authentication mechanism
    AuthInfo result;
    result.username = req.form.get("username", "anonymous");
    return result;
}
```

And the class defines how to get authentication
Add authentication

```cpp
@path("/wh2")
@requiresAuth
class WebHandler2

{
    string[string] items;

    @noAuth
    void getIndex(HTTPServerResponse res)
    {
        auto items = this.items;
        res.render!("index.dt", items);
    }

demo(Role.editor)
```

Listing the items doesn't require any auth
Add authentication

```java
@noAuth
void getIndex(HTTPServerResponse res)
{
    auto items = this.items;
    res.render!("index.dt", items);
}

@auth(Role.editor)
void postItem(HTTPServerRequest req, string name, string value,
               HTTPServerResponse res)
{
    items[name] = value;
    res.redirect("/wh2/index");
}
```

But setting an item does
Add authentication

```cpp
void postItem(HTTPSRequest req, string name, string value,
              HTTPSResponse res)
{
    items[name] = value;
    res.redirect("/wh2/index");
}

@auth(Role.viewer)
@path("item/:name")
void getItem(string _name, HTTPSResponse res)
{
    if(auto v = _name in items)
        res.writeBody("<html><body>Item named " ~ _name ~ " is "
                      ~ +v ~ "<br><a href='/wh2/index'>Home</a></body></html> ",
                      res.contentType("text/html");
}
```

```cpp
_name is passed to authinfo.isViewer
```
DEMO OF VIBE.D
FIXING THE DRAWBACKS
COMPILE TIME IS IMPORTANT

• template and generative programming are compile-time killers.
• vibe.d default installation takes 4.5 seconds to compile on my 2014 Mac.
• CTFE and templates are suspects in compile time bloaters.
AVOID THE BOILERPLATE

- Any time you can avoid boilerplate, you should.
- Have the compiler write your boilerplate, it's better at it.
- Generative programming is the crown jewel of D!
YOU HAVE A D COMPILER, USE IT!
D POWERING DEVICES
WHAT IF D COULD RUN ON DRONES?

- Dconf 2014 - Michael V. Franklin "Tiny Ubiquitous Machines Powered by D"
- Used spec sheet text directly to create register maps
- Well, almost directly
USE TEMPLATES TO GENERATE PERFECT BOILER PLATE REGISTER MAPS

library: https://github.com/JinShil/memory_mapped_io
USE TEMPLATES TO GENERATE PERFECT BOILER PLATE REGISTER MAPS

• Enforces all reads and writes are done with as few instructions as possible, and with the correct access type.
• Generated access was 6.5x faster than the chip supplier's C library call
• Once you map the spec to code, ALL the boilerplate is solved for every register
• 1700 page spec
CHIP SPEC CAN BE USED AS CODE?

• Imagine if you can simply import the spec, and process it.
• The spec becomes the code, transpiled by the compiler at compile time to the most efficient register mapping possible
• No more discrepancies between code libraries and spec!