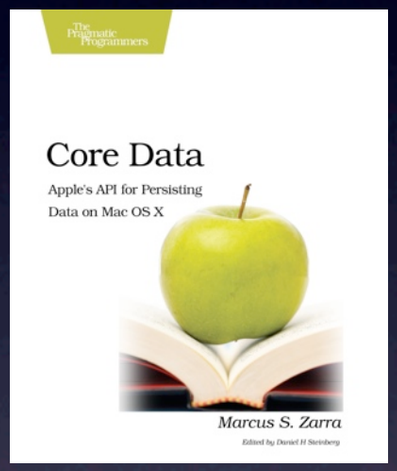
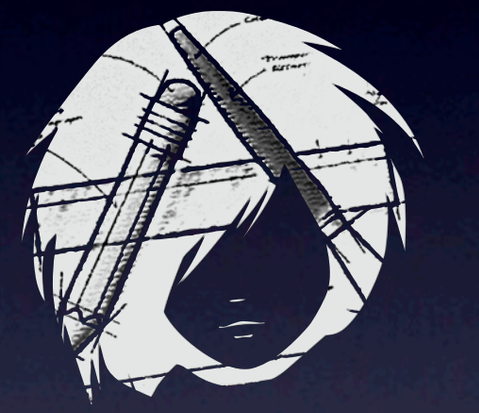
Flexible JSON Importing

Marcus S. Zarra

Who Am I?





COCOA IS MY GIRLFRIEND
CIMGF.COM

stackoverflow.com

News Apps

News Apps

Tiny Devices

Flexible Importing

Transport Format

Transport Format

Binary PList

Transport Format

~~Binary~~ PList

Transport Format

~~Binary~~ / PList

XML

Transport Format

~~Binary~~ PList

~~XML~~

Transport Format

~~Binary~~ / ~~PList~~

~~XML~~

JSON

Transport Protocol

Transport Protocol

HTTP

Transport Protocol

HTTP

Transport Protocol

~~HTTP~~

Compressed HTTP

JSON Parser

JSON Parser

Yet Another JSON Library (YAJL)

http://lloyd.github.com/yajl/

ZDSStreamParser

ZDSStreamParser

- Stream JSON Parser

ZDSStreamParser

- Stream JSON Parser
- Inserts Data Into Core Data

ZDSStreamParser

- Stream JSON Parser
- Inserts Data Into Core Data
- Dynamic Entity Resolution

Data Flow

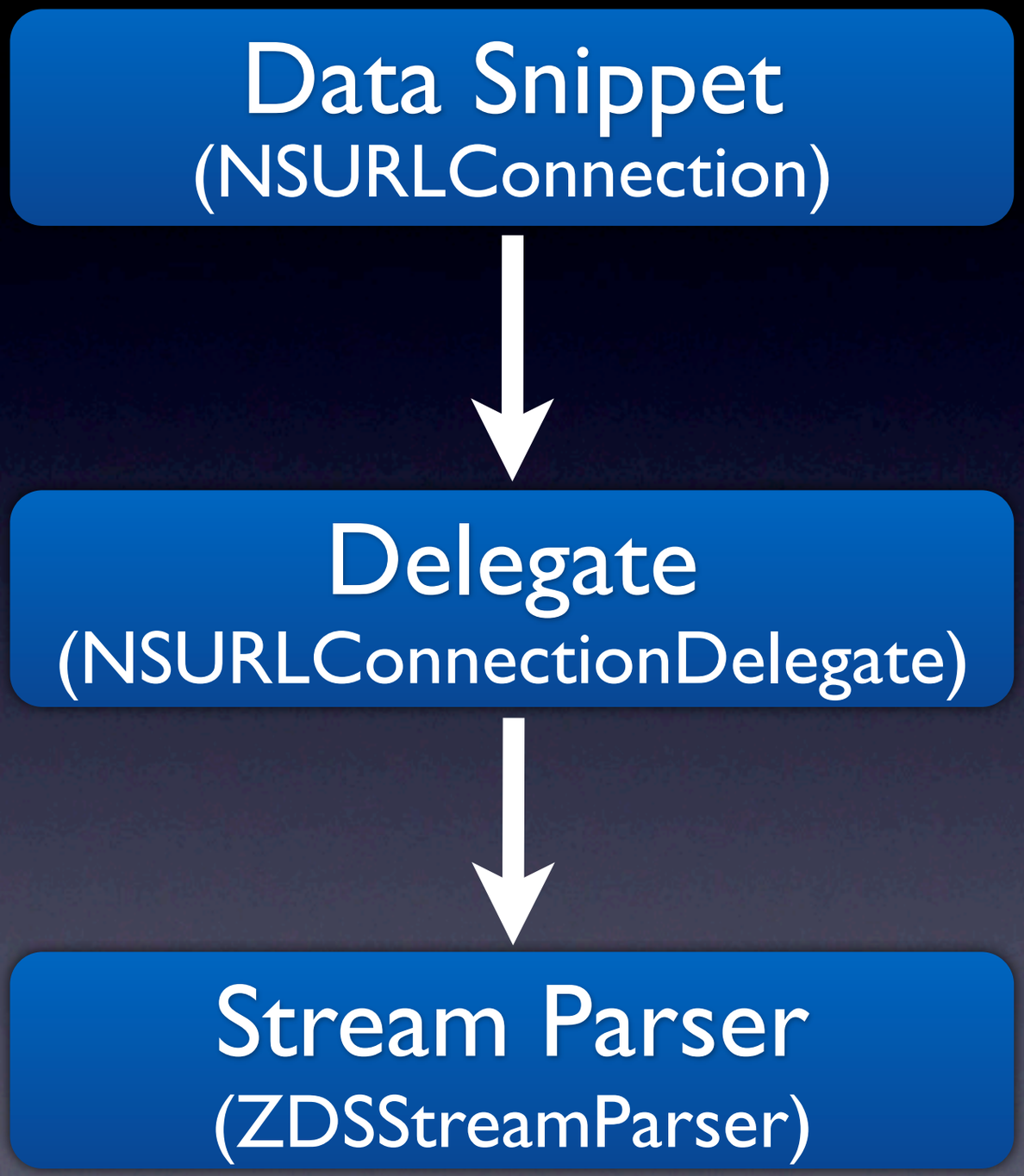
Data Snippet
(NSURLConnection)

Data Flow

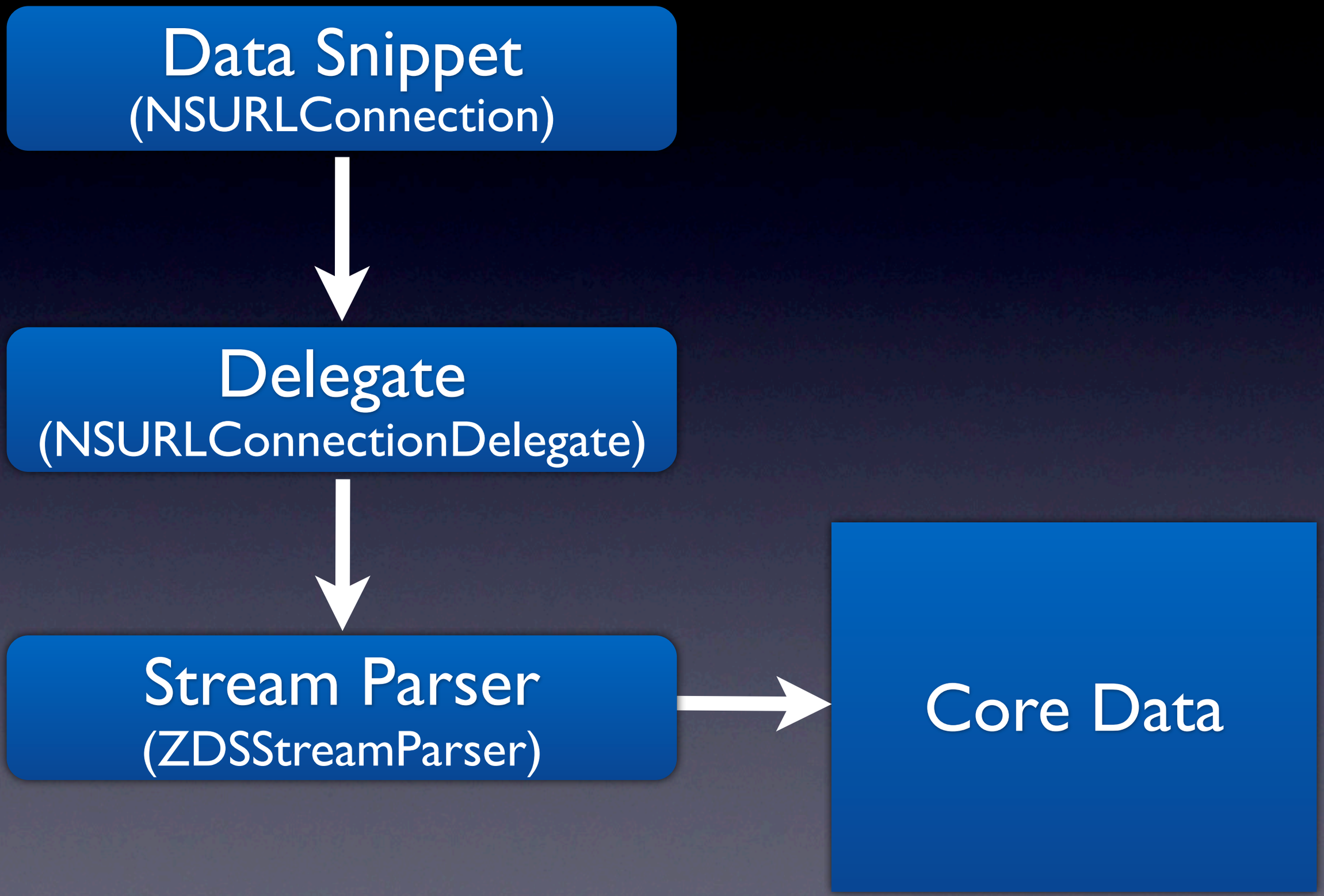
Data Snippet
(NSURLConnection)

Delegate
(NSURLConnectionDelegate)

Data Flow


Data Snippet
(NSURLConnection)
Delegate
(NSURLConnectionDelegate)
Stream Parser
(ZDSStreamParser)

Data Flow


Data Snippet
(NSURLConnection)
Delegate
(NSURLConnectionDelegate)
Stream Parser
(ZDSStreamParser)
Core Data

NSURLConnectionDelegate

```
- (void)connection:(NSURLConnection*)connection
    didReceiveData:(NSData*)newData
{
  YAJLParserStatus status = [[self parser] parse:newData];
  if (status == YAJLParserStatusInsufficientData) return;
  if (status == YAJLParserStatusOK) return;

  NSError *error = [[self parser] parserError];
  ALog(@"Data parsing has failed: %@", error);
}
```

NSURLConnectionDelegate

```
- (void)connection:(NSURLConnection*)connection
    didReceiveData:(NSData*)newData
{
    YAJLParserStatus status = [[self parser] parse:newData];
    if (status == YAJLParserStatusInsufficientData) return;
    if (status == YAJLParserStatusOK) return;

    NSError *error = [[self parser] parserError];
    ALog(@"Data parsing has failed: %@", error);
}
```

NSURLConnectionDelegate

```
- (void)connection:(NSURLConnection*)connection
    didReceiveData:(NSData*)newData
{
  YAJLParserStatus status = [[self parser] parse:newData];
  if (status == YAJLParserStatusInsufficientData) return;
  if (status == YAJLParserStatusOK) return;

  NSError *error = [[self parser] parserError];
  ALog(@"Data parsing has failed: %@", error);
}
```

NSURLConnectionDelegate

```
- (void)connection:(NSURLConnection*)connection
    didReceiveData:(NSData*)newData
{
    YAJLParserStatus status = [[self parser] parse:newData];
    if (status == YAJLParserStatusInsufficientData) return;
    if (status == YAJLParserStatusOK) return;

    NSError *error = [[self parser] parserError];
    ALog(@"Data parsing has failed: %@", error);
}
```

YAJL Parser Delegate

- -parserDidStartDictionary:
- -parserDidEndDictionary:
- -parserDidStartArray:
- -parserDidEndArray:
- -parser: didMapKey:
- -parser: didAdd:

YAJL Parser Delegate

- -parserDidStartDictionary:
- -parserDidEndDictionary:
- -parserDidStartArray:
- -parserDidEndArray:
- -parser: didMapKey:
- -parser: didAdd:

YAJL Parser Delegate

- -parserDidStartDictionary:
- -parserDidEndDictionary:
- ~~-parserDidStartArray:~~
- ~~-parserDidEndArray:~~
- -parser: didMapKey:
- -parser: didAdd:

Fracking Server Guys

id

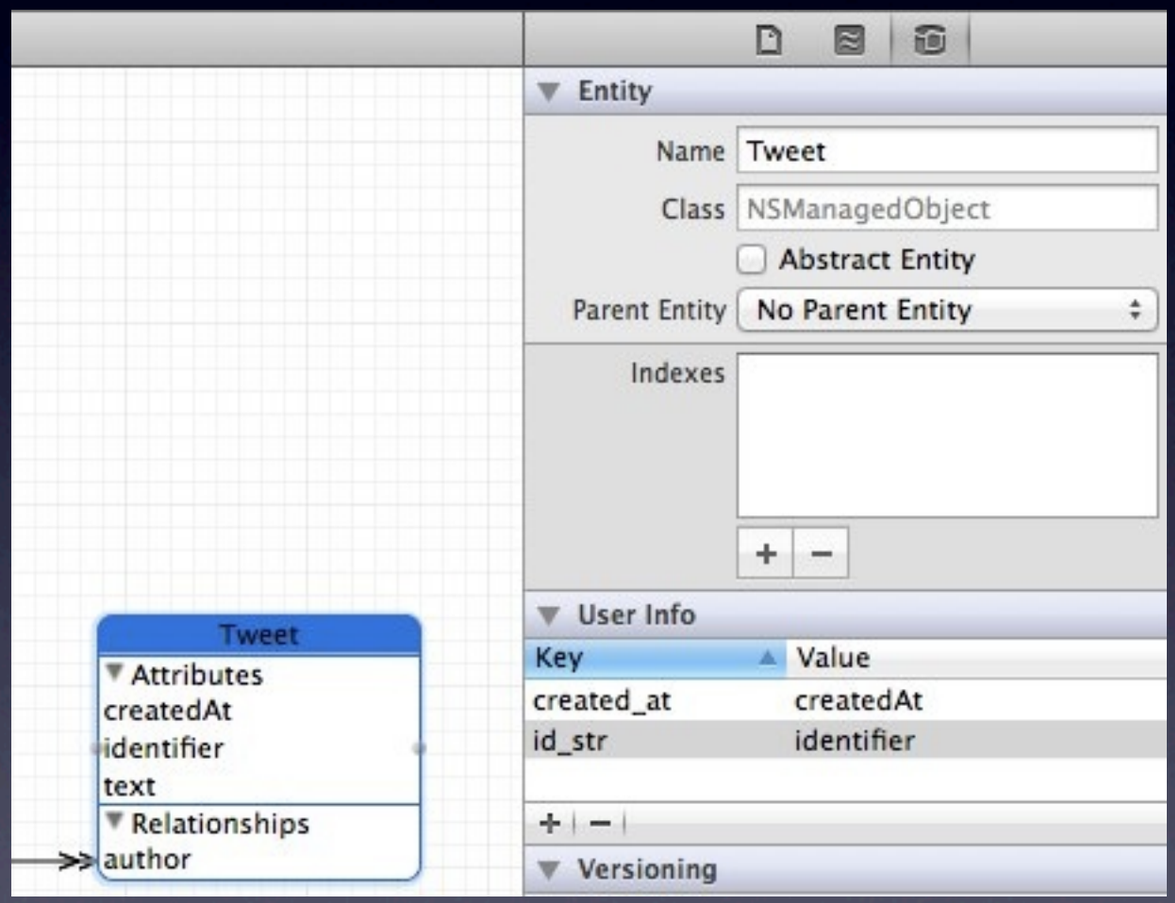
description

date_added

id_str

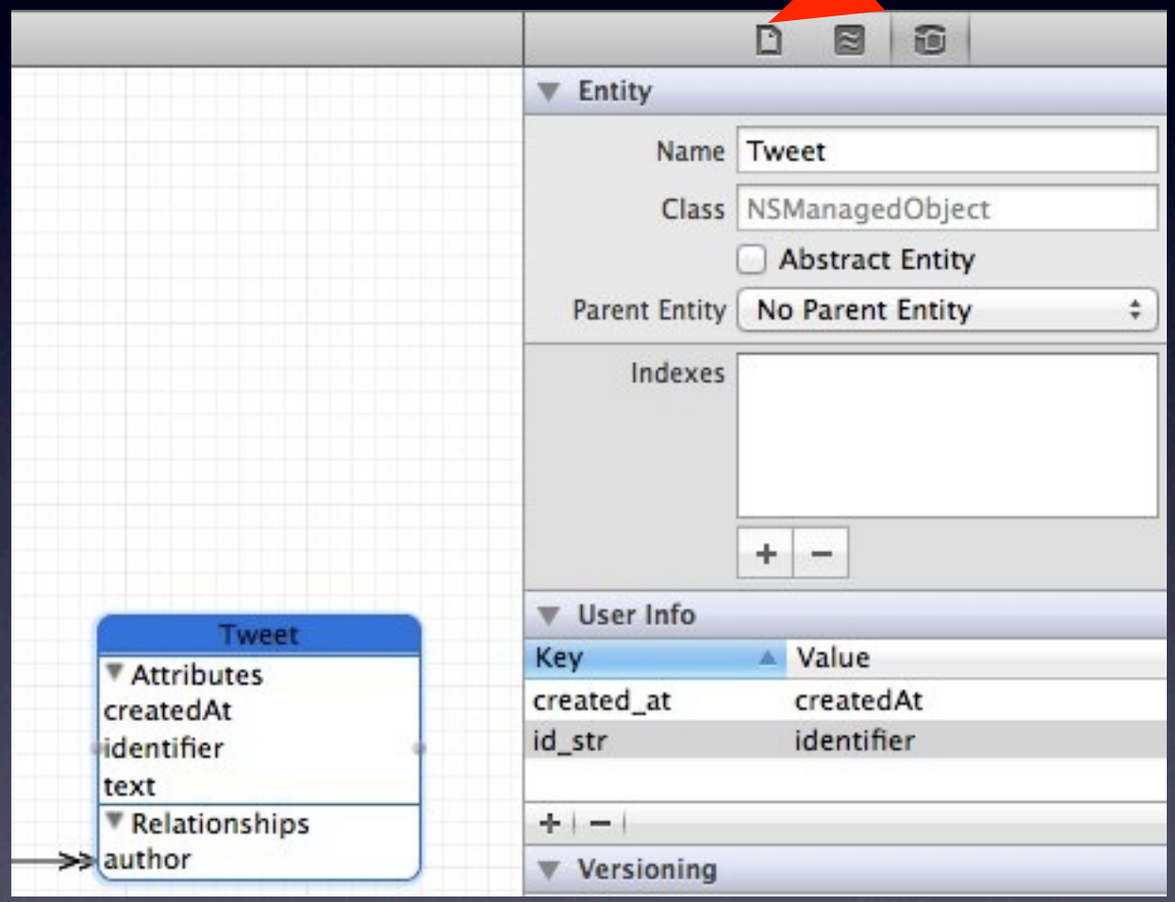
in_reply_to_user_id_str

Core Data Modeler



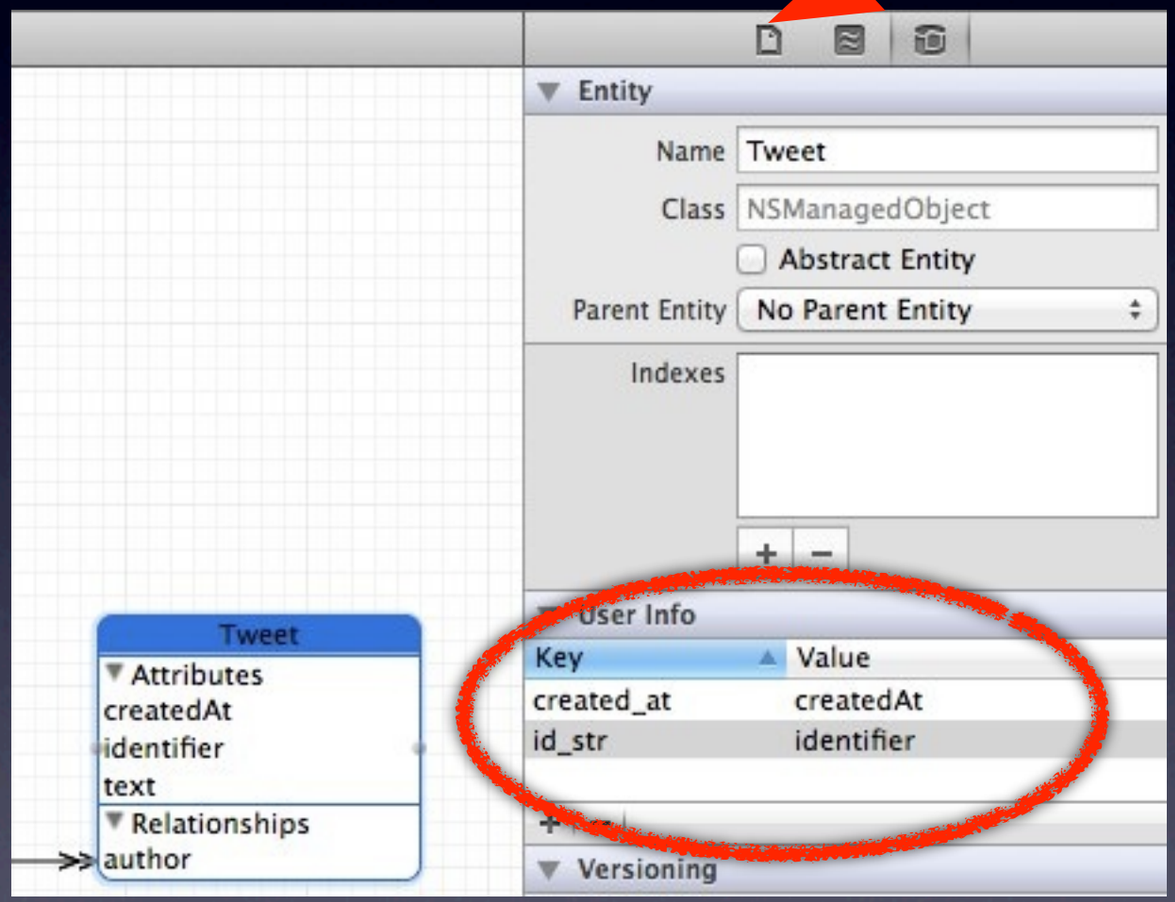
Core Data Modeler

Third Assistant



Core Data Modeler

Third Assistant



-parser: didMapKey:

```
- (void)parser:(YAJLParser*)parser didMapKey:(NSString*)key
{

  if (![self currentObject]) {
    [self setCurrentKey:key];
    return;
  }

  NSDictionary *userInfo = [[[self currentObject] entity] userInfo];
  if (!userInfo) {
    [self setCurrentKey:key];
    return;
  }

  NSString *resolvedKey = [userInfo valueForKey:[self currentKey]];
  if (!resolvedKey) {
    [self setCurrentKey:key];
    return;
  }

  [self setCurrentKey:key];
}
```

-parser: didMapKey:

```
- (void)parser:(YAJLParser*)parser didMapKey:(NSString*)key
{
  if (![self currentObject]) {
    [self setCurrentKey:key];
    return;
  }
  NSDictionary *userInfo = [[[self currentObject] entity] userInfo];
  if (!userInfo) {
    [self setCurrentKey:key];
    return;
  }

  NSString *resolvedKey = [userInfo valueForKey:[self currentKey]];
  if (!resolvedKey) {
    [self setCurrentKey:key];
    return;
  }

  [self setCurrentKey:key];
}
```

-parser: didMapKey:

```
- (void)parser:(YAJLParser*)parser didMapKey:(NSString*)key
{

  if (![self currentObject]) {
    [self setCurrentKey:key];
    return;
  }

  NSDictionary *userInfo = [[[self currentObject] entity] userInfo];
  if (!userInfo) {
    [self setCurrentKey:key];
    return;
  }

  NSString *resolvedKey = [userInfo valueForKey:[self currentKey]];
  if (!resolvedKey) {
    [self setCurrentKey:key];
    return;
  }

  [self setCurrentKey:key];
}
```

-parser: didMapKey:

```
- (void)parser:(YAJLParser*)parser didMapKey:(NSString*)key
{

  if (![self currentObject]) {
    [self setCurrentKey:key];
    return;
  }

  NSDictionary *userInfo = [[[self currentObject] entity] userInfo];
  if (!userInfo) {
    [self setCurrentKey:key];
    return;
  }

  NSString *resolvedKey = [userInfo valueForKey:[self currentKey]];
  if (!resolvedKey) {
    [self setCurrentKey:key];
    return;
  }

  [self setCurrentKey:key];
}
```

-parser: didMapKey:

```
- (void)parser:(YAJLParser*)parser didMapKey:(NSString*)key
{

  if (![self currentObject]) {
    [self setCurrentKey:key];
    return;
  }

  NSDictionary *userInfo = [[[self currentObject] entity] userInfo];
  if (!userInfo) {
    [self setCurrentKey:key];
    return;
  }

  NSString *resolvedKey = [userInfo valueForKey:[self currentKey]];
  if (!resolvedKey) {
    [self setCurrentKey:key];
    return;
  }

  [self setCurrentKey:key];
}
```

-parserDidStartDictionary:

```
entity = [[self currentObject] entity];
relationships = [entity relationshipsByName];
relationship = [relationships objectForKey:[self currentKey]];

if (!relationship) {
  DLog(@"Unknown relationship in the stream: %@; skipping", [self currentKey]);
  [self setSkipDictionaryCount:([self skipDictionaryCount] + 1)];
  return;
}

destinationObjectName = [[relationship destinationEntity] name];
id destinationObject = [NSEntityDescription insertNewObjectForEntityForName:destinationObjectName
                                                      inManagedObjectContext:[self moc]];

if ([relationship isToMany]) {
  NSMutableSet *children = [[self currentObject] mutableSetValueForKey:[self currentKey]];
  [children addObject:destinationObject];
} else {
  [[self currentObject] setValue:destinationObject forKey:[self currentKey]];
}


ZDSStreamJSONParser *aChildParser = nil;
aChildParser = [[ZDSStreamJSONParser alloc] initWithManagedObjectContext:[self moc]];
[aChildParser setParent:self];
[aChildParser setCurrentObject:destinationObject];
[parser setDelegate:aChildParser];
[self setChildParser:aChildParser];
```

-parserDidStartDictionary:

```
entity = [[self currentObject] entity];
relationships = [entity relationshipsByName];
relationship = [relationships objectForKey:[self currentKey]];

if (!relationship) {
  DLog(@"Unknown relationship in the stream: %@; skipping", [self currentKey]);
  [self setSkipDictionaryCount:([self skipDictionaryCount] + 1)];
  return;
}


destinationObjectName = [[relationship destinationEntity] name];
id destinationObject = [NSEntityDescription insertNewObjectForEntityForName:destinationObjectName
                                                      inManagedObjectContext:[self moc]];

if ([relationship isToMany]) {
  NSMutableSet *children = [[self currentObject] mutableSetValueForKey:[self currentKey]];
  [children addObject:destinationObject];
} else {
  [[self currentObject] setValue:destinationObject forKey:[self currentKey]];
}


ZDSStreamJSONParser *aChildParser = nil;
aChildParser = [[ZDSStreamJSONParser alloc] initWithManagedObjectContext:[self moc]];
[aChildParser setParent:self];
[aChildParser setCurrentObject:destinationObject];
[parser setDelegate:aChildParser];
[self setChildParser:aChildParser];
```

-parserDidStartDictionary:

```
entity = [[self currentObject] entity];
relationships = [entity relationshipsByName];
relationship = [relationships objectForKey:[self currentKey]];

if (!relationship) {
  DLog(@"Unknown relationship in the stream: %@; skipping", [self currentKey]);
  [self setSkipDictionaryCount:([self skipDictionaryCount] + 1)];
  return;
}

destinationObjectName = [[relationship destinationEntity] name];
id destinationObject = [NSEntityDescription insertNewObjectForEntityForName:destinationObjectName
                                                      inManagedObjectContext:[self moc]];

if ([relationship isToMany]) {
  NSMutableSet *children = [[self currentObject] mutableSetValueForKey:[self currentKey]];
  [children addObject:destinationObject];
} else {
  [[self currentObject] setValue:destinationObject forKey:[self currentKey]];
}

ZDSStreamJSONParser *aChildParser = nil;
aChildParser = [[ZDSStreamJSONParser alloc] initWithManagedObjectContext:[self moc]];
[aChildParser setParent:self];
[aChildParser setCurrentObject:destinationObject];
[parser setDelegate:aChildParser];
[self setChildParser:aChildParser];
```

-parserDidEndDictionary:

```
- (void)parserDidEndDictionary:(YAJLParser*)parser
{
    if ([self skipDictionaryCount] > 0) {
        [self setSkipDictionaryCount:([self skipDictionaryCount] - 1)];
        return;
    }
    [parser setDelegate:[self parent]];
}
```

-parser: didAdd:

```
- (void)parser:(YAJLParser*)parser didAdd:(id)value;
{
  ZAssert([self currentObject], @"Add value without object: %@\n%@",
    [self currentKey], value);

  NSEntityDescription *entity = [[self currentObject] entity];
  NSDictionary *properties = [entity propertiesByName];
  id property = [properties valueForKey:[self currentKey]];

  if (!property) { // Fall back to KVC
    SEL selector = NSSelectorFromString([self currentKey]);
    if (![[self currentObject] respondsToSelector:selector]) {
      return;
    }

    [[self currentObject] setValue:value
                            forKey:[self currentKey]];
    return;
  }
```

-parser: didAdd:

```
switch ([property attributeType]) {
  case NSStringAttributeType:
    if ([value isKindOfClass:[NSString class]]) {
      [[self currentObject] setValue:value forKey:[self currentKey]];
      return;
    } else if ([value isKindOfClass:[NSNumber class]]) {
      [[self currentObject] setValue:[value stringValue] forKey:[self currentKey]];
      return;
  } else if ([value isKindOfClass:[NSNull class]]) {
    [[self currentObject] setValue:nil forKey:[self currentKey]];
    return;
  }
  ALog(@"unparsable data class %@ to string against class %@",
       [value class], [[[self currentObject] entity] name]);
  return;
```

-parser: didAdd:

```
case NSDateAttributeType:
  ZAssert([value isKindOfClass:[NSString class]],
          @"unparsable data class %@ to number against class %@",
          [value class], [[[self currentObject] entity] name]);
  if ([value length] == 0) return;
  [[self currentObject] setValue:[dateFormatter dateFromString:value]
                          forKey:[self currentKey]];
  return;
```

-parser: didAdd:

```
case NSInteger16AttributeType:
case NSInteger32AttributeType:
case NSInteger64AttributeType:
case NSDoubleAttributeType:
case NSFloatAttributeType:
case NSBooleanAttributeType:
  if ([value isKindOfClass:[NSNumber class]]) {
    [[self currentObject] setValue:value forKey:[self currentKey]];
    return;
  } else if ([value isKindOfClass:[NSString class]]) {

  } else {
    ALog(@"unparsable data class %@ to number against class %@",
         [value class], [[[self currentObject] entity] name]);
    return;
  }
```

Current Limitations

- Top of the tree is manual
- Update vs. Insert is manual
- Incremental saves not implemented

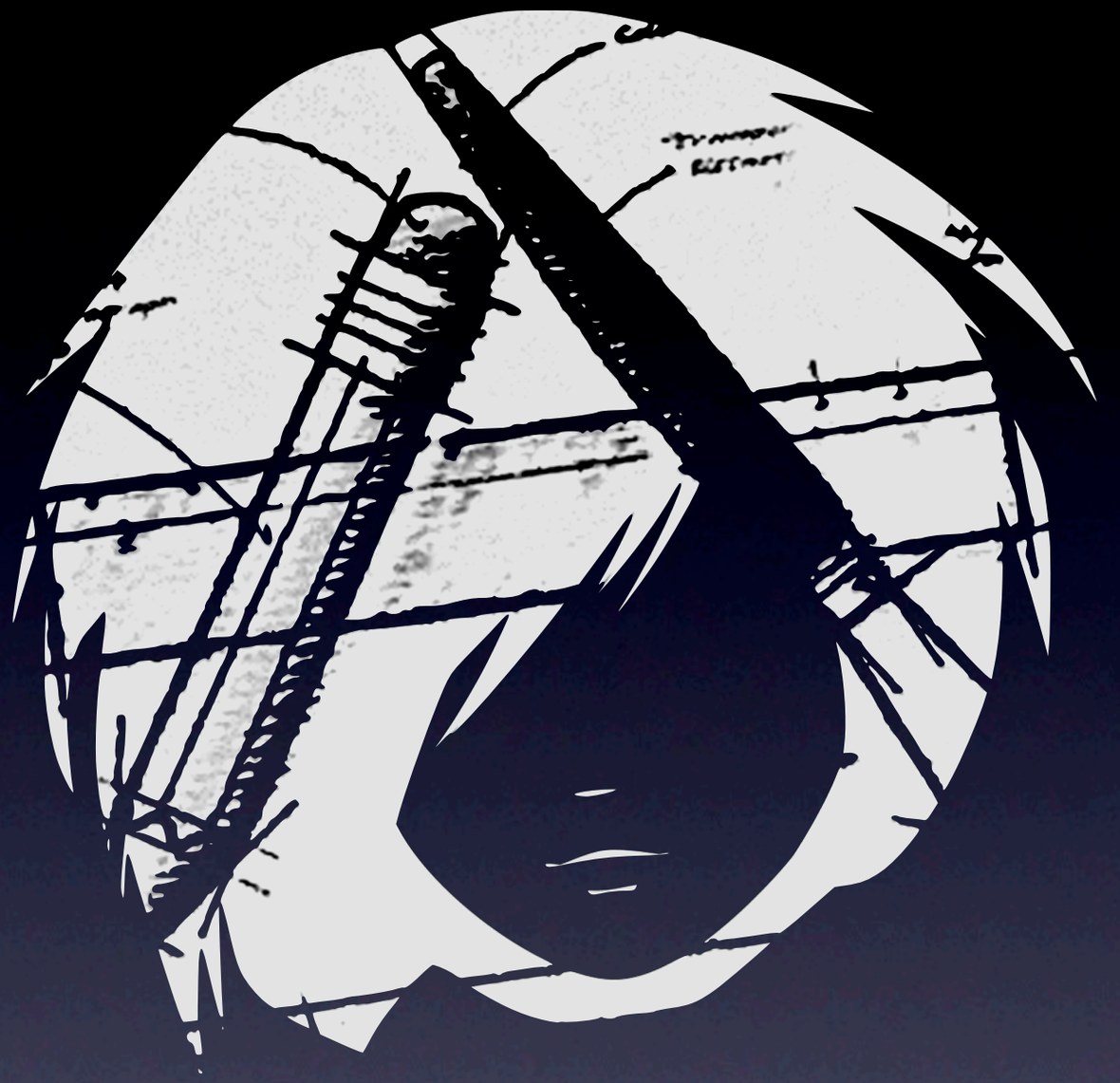
Source Code

github.com/organizations/ZarraStudios

ZDS_Shared

cimgf.com

@mzarra



Marcus S. Zarra

marcus@cimgf.com