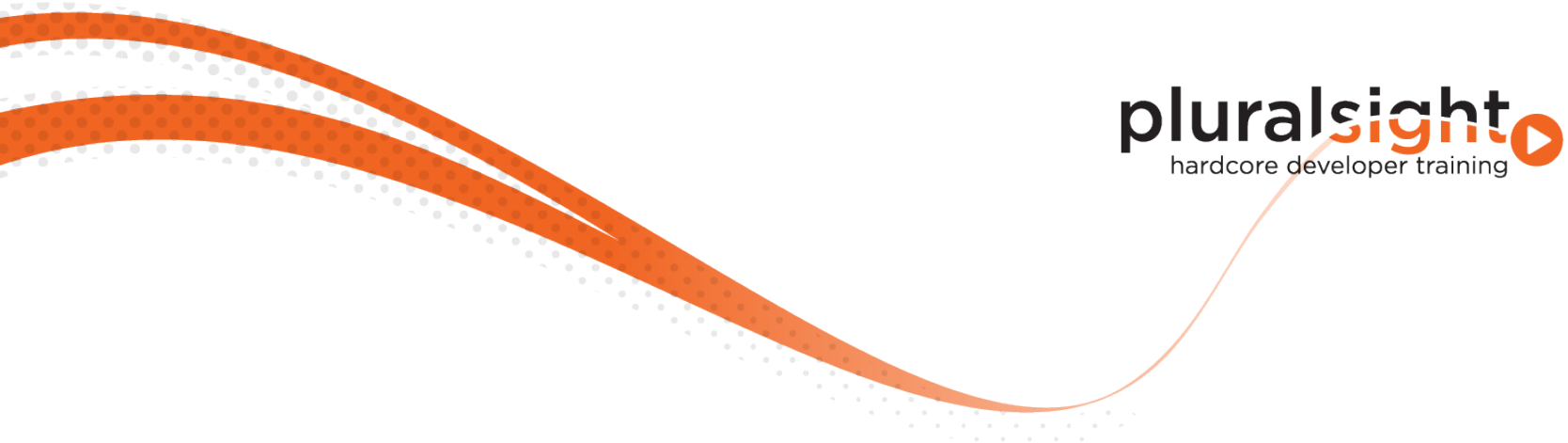
ActiveRecord



Models

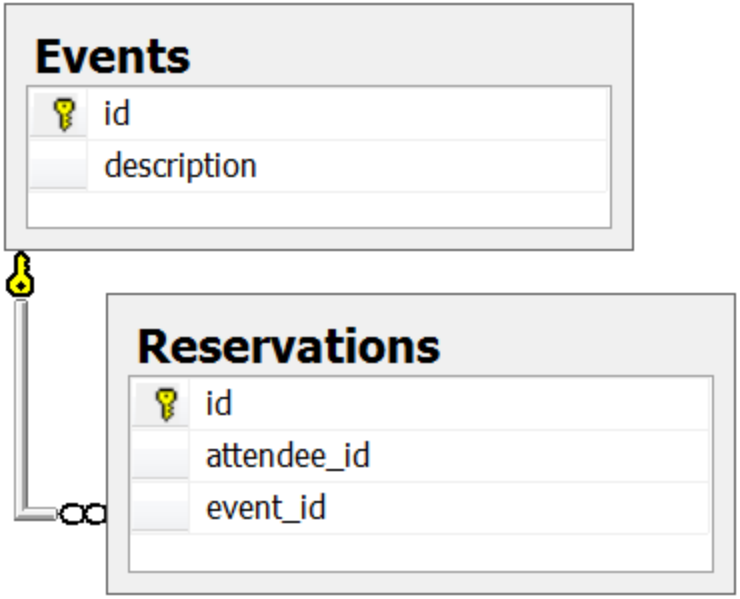
- Represent data
- Control business rules for manipulation
- In Rails, they are used to manage interaction with tables
- One model for each table

ActiveRecord

- Default ORM in Rails
- Connects data models to database tables
- Perform CRUD and manage model relationships for us
- ActiveRecord is an implementation of the ActiveRecord pattern
- Models carry both data and behavior
- Model attributes are inferred from the table definition

Relationships

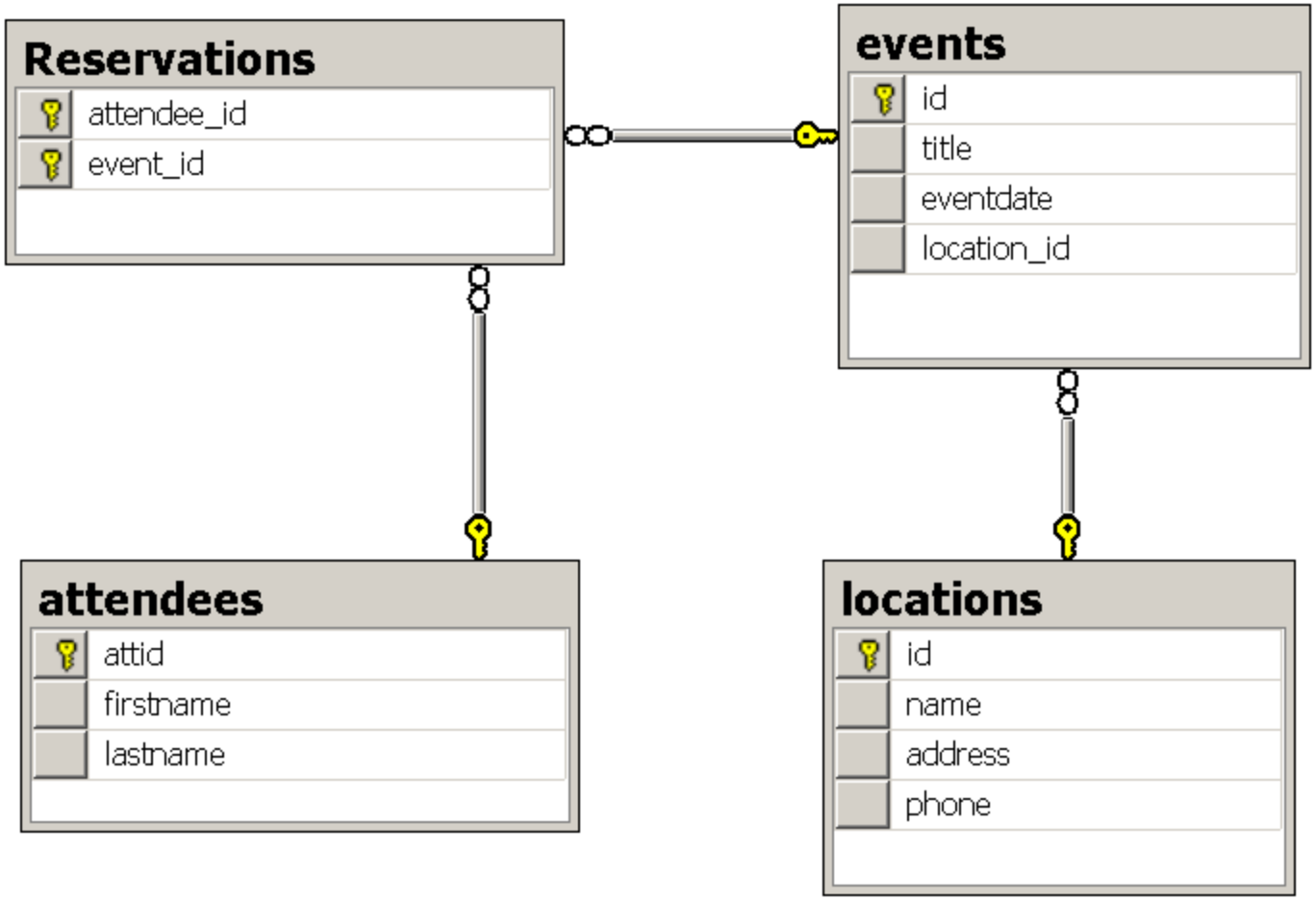
- **ActiveRecord favors convention over configuration**
- **Naming conventions are used to map models to tables**
  - Book -> Books
  - Models should use CamelCase (e.g. BookClub)
  - Tables should use Snake_Case (e.g. Book_Clubs)
- **Primary keys**
  - Named “ID”
  - Integer
  - Auto incrementing
- **Forign Keys**
  - Format: SingularTableName_PrimaryKey



Relationships

- **Has many through**
  - One to many association
  - Done through a 3rd model
- **"Join" table**
  - Has two forign keys
  - Provides loose coupling

Relationships

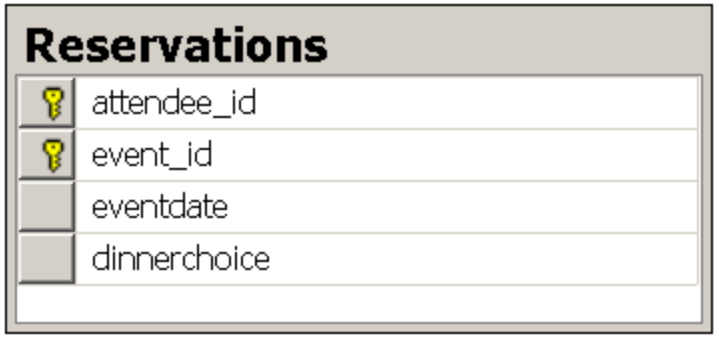


Relationships

- **Has and Belongs to Many**
  - Many to many association
  - HABTM
  - Less setup

Relationships

- **Use has many through**
  - When you need to work with the intermediate model
- **Use Has and Belongs to Many**
  - When you only n



Validation helpers

- **Uniqueness**
  - Ensures unique values
  - Performs actual database query

```
class Account

  validates :phone, :uniqueness => true

end
```

Validation helpers

- **Acceptance**
  - Validates the state of a check box

```
class Account

  validates :agrees_to_terms, :acceptance => true

end
```

Validation helpers

- **Confirmation**
  - Confirms that two values match
  - Will validate against two attributes using convention
  - email Vs. email_confirmation

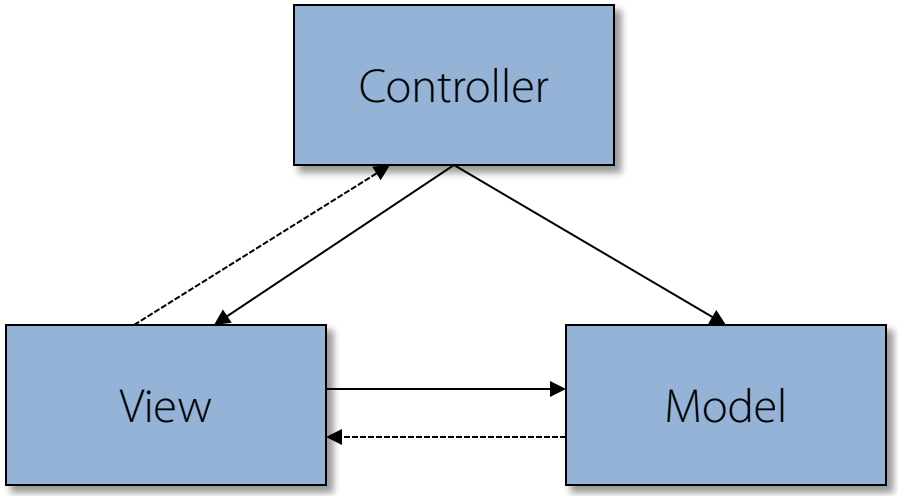
```
class Account

  validates :email, :confirmation => true

end
```

Validation helpers

- **Allow Nil, Allow Blank**
  - Prevent nil or empty values
  - Can be used along side other validations to allow nil or empty values

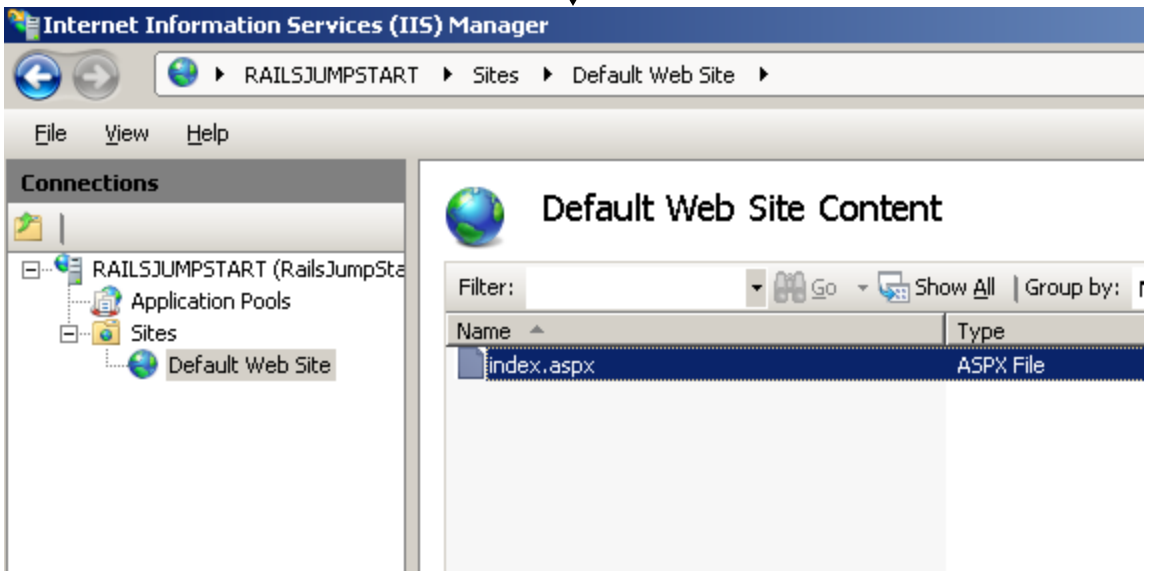
```
class Account

  validates :address, :format => { :with => /\A[0-9]+\z/ }, :allow_blank => true
  validates :street, :length =>  { :minimum => 4 }, :allow_nil => true
end
```

Controllers



Controllers

http://www.myblog.com



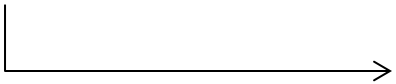
Controllers

http://www.myblog.com

```
class BlogPostsController < ApplicationController
  # GET /blog_posts
  # GET /blog_posts.json
  def index
    @blog_posts = BlogPost.all

    if user_signed_in?
      render
    else
      render 'home/index'
    end

  end

  #POST /comment
  def comment
  @comment = Comment.new(params[:comment])

    if !@comment.save
      flash[:comment_errors] = @comment.errors.full_messages
    end

    redirect_to blog_post_path(@comment.blog_post_id)
  end
```

Routing

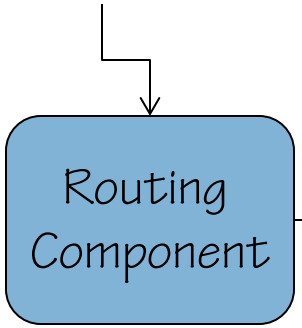
http://www.myblog.com



```ruby
class BlogPostsController < ApplicationController
  # GET /blog_posts
  # GET /blog_posts.json
  def index
    @blog_posts = BlogPost.all

    if user_signed_in?
      render
    else
      render 'home/index'
    end

  end

  #POST /comment
  def comment
  @comment = Comment.new(params[:comment])

    if !@comment.save
      flash[:comment_errors] = @comment.errors.full_messages
    end

    redirect_to blog_post_path(@comment.blog_post_id)
  end
```

Routing

http://www.myblog.com

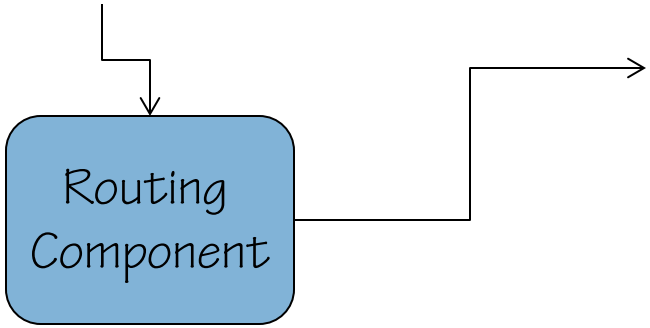



```ruby
class BlogPostsController < ApplicationController
  # GET /blog_posts
  # GET /blog_posts.json
  def index
    @blog_posts = BlogPost.all

    if user_signed_in?
      render
    else
      render 'home/index'
    end

  end

  #POST /comment
  def comment
  @comment = Comment.new(params[:comment])

    if !@comment.save
      flash[:comment_errors] = @comment.errors.full_messages
    end

    redirect_to blog_post_path(@comment.blog_post_id)
  end
```

```ruby
get "home/index"
get "home/newaction" => "blog_posts#index"

get ":title/p/:id" => "blog_posts#show", :id => /[0-9]+/

post "blogposts/comment" => "blog_posts#comment"
```

Routing

http://www.myblog.com




```ruby
class BlogPostsController < ApplicationController
  # GET /blog_posts
  # GET /blog_posts.json
  def index
    @blog_posts = BlogPost.all

    if user_signed_in?
      render
    else
      render 'home/index'
    end

  end

  #POST /comment
  def comment
  @comment = Comment.new(params[:comment])

    if !@comment.save
      flash[:comment_errors] = @comment.errors.full_messages
    end

    redirect_to blog_post_path(@comment.blog_post_id)
  end
```

```ruby
get "home/index"
get "home/newaction" => "blog_posts#index"

get ":title/p/:id" => "blog_posts#show", :id => /[0-9]+/

post "blogposts/comment" => "blog_posts#comment"
```

Summary

- **ActiveRecord is the default ORM in Rails**
- **ActiveRecord connects our models to database tables**
- **ActiveRecord Conventions**
  - Table names should be plural
  - Model names should be singular
  - For compound name, use snake case on tables and camel case on models
  - Primary key should be an auto incrementing integer column named ‘id’
  - Foreign keys should contain the table name and primary key column name
- **We ca override convention**
  - self.primary_key
  - self.table_name

Summary

- **One-to-one**
  - belongs_to
- **One-to-many**
  - has_many
- **Many-to-many**
  - Has_many :through
  - Has_and_belongs_to_many

Summary

- **Validates**
  - Format
  - Length
  - Uniqueness
  - Acceptance
  - Confirmation
  - Allow_nil
  - Allow_blank

- **http://guides.rubyonrails.org**