

Models



SELECT * FROM SONG;

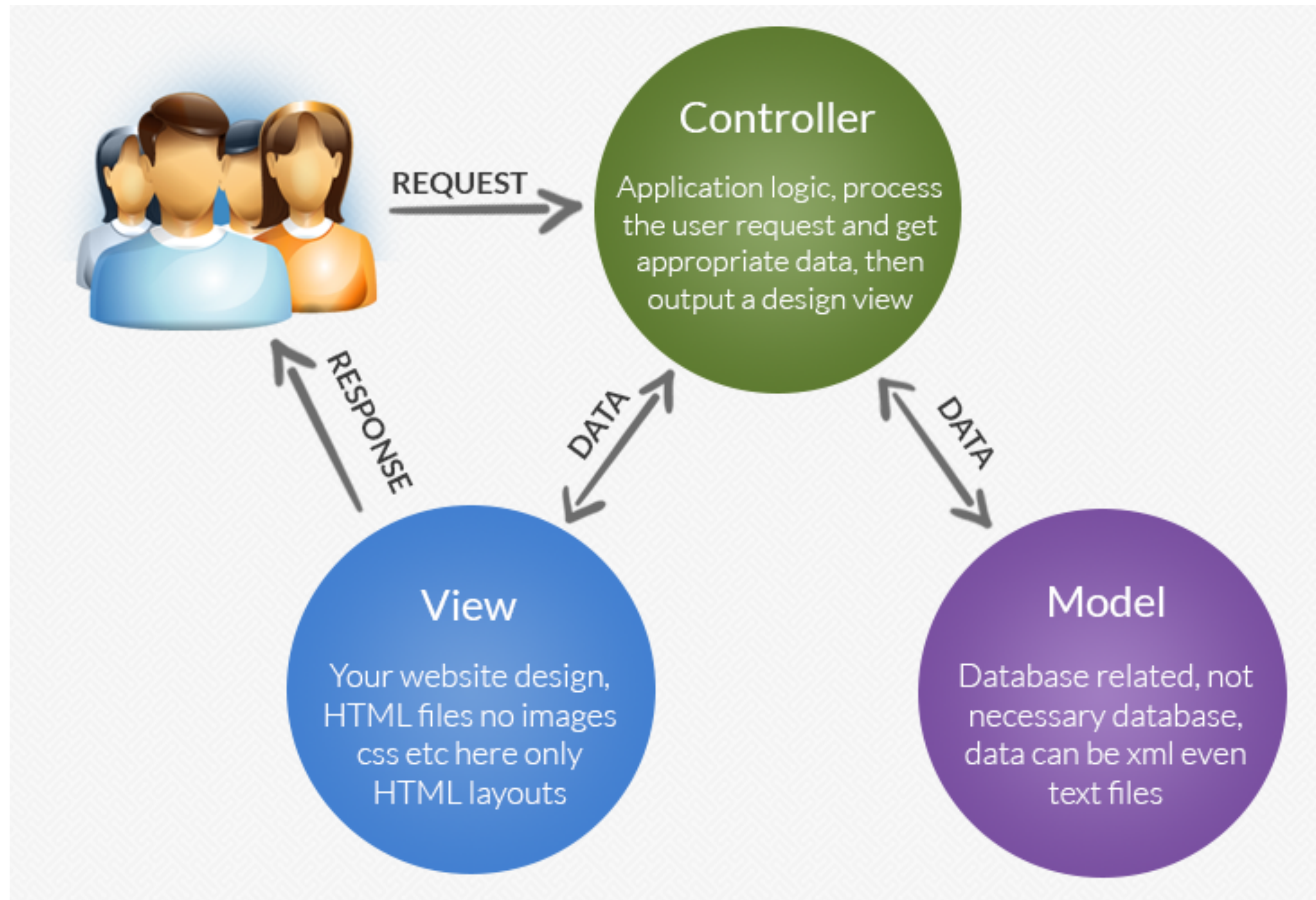
ID	ARTIST	DURATION	TITLE
1	Beethoven	0	Piano Sonata No. 3
2	Beethoven	0	Piano Sonata No. 7
3	Beethoven	0	Piano Sonata No. 10
4	Beethoven	0	Piano Concerto No. 27
5	Beethoven	0	Piano Concertos No. 17
6	Beethoven	0	Piano Concerto No. 10

(6 rows, 6 ms)

Edit

table · row · mem · yaml ·
Song · Playlist · ArrayList ·
OneToMany · ...

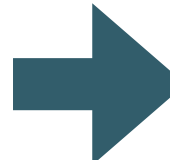
Model View Controller



Database in Play

Configuration file
specifies a
database that will
be integrated into
the application

conf/application.conf

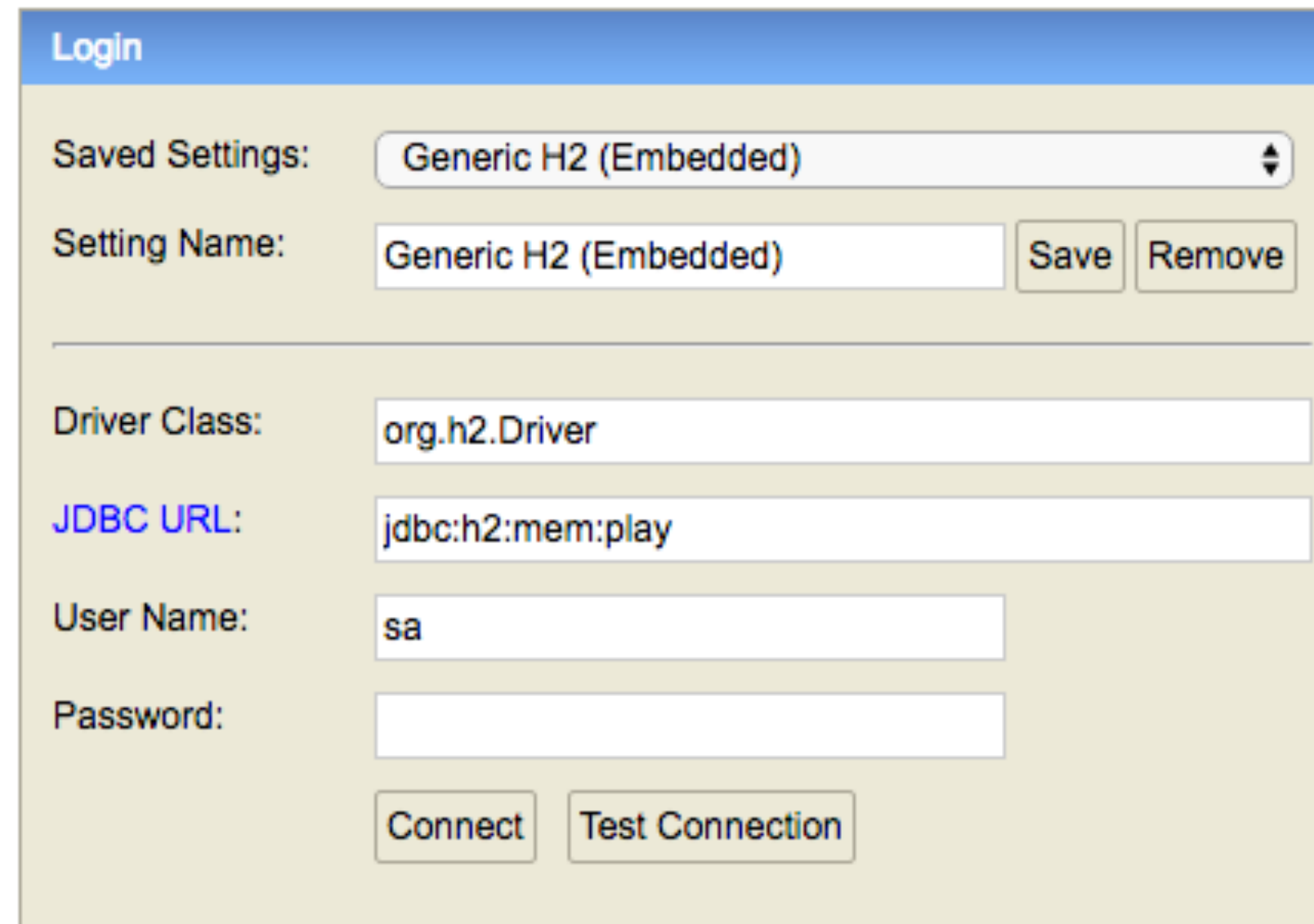


```
...  
# Database configuration  
# ~~~~~  
# Enable a database engine if needed.  
#  
# To quickly set up a development database, use either:  
#   - mem : for a transient in memory database (H2 in memory)  
#   - fs  : for a simple file written database (H2 file stored)  
db.default=mem  
...
```

In Memory test database
Full SQL support
Replaced with 'production'
database at a later stage

Inspecting the Database in Play

When app is running, browse to ➡ <http://localhost:9000/@db>



The screenshot shows a web interface titled "Login" with a blue header. Below the header, there are several input fields and buttons. The "Saved Settings" section shows a dropdown menu with "Generic H2 (Embedded)" selected. Below this, the "Setting Name" field also contains "Generic H2 (Embedded)", with "Save" and "Remove" buttons to its right. A horizontal line separates this from the configuration section. The "Driver Class" field contains "org.h2.Driver". The "JDBC URL" field contains "jdbc:h2:mem:play". The "User Name" field contains "sa". The "Password" field is empty. At the bottom, there are "Connect" and "Test Connection" buttons.

Login	
Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded) [Save] [Remove]
<hr/>	
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:play
User Name:	sa
Password:	
[Connect] [Test Connection]	

log in to database

Database console

Database
Tables



The screenshot shows the Database console interface. On the left, a tree view displays the database schema for 'jdbc:h2:mem:play'. It includes tables 'playlist' and 'song', each with columns 'id', 'duration', and 'title'. There are also indexes for 'playlist_song' and 'song'. The bottom of the tree shows 'information_schema', 'Sequences', and 'Users'. The version 'H2 1.4.193 (2016-10-31)' is displayed at the bottom left.

On the right, the SQL statement panel is visible. It has a toolbar with buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. Below the toolbar is a large text area for the SQL statement. The panel also contains sections for 'Important Commands', 'Sample SQL Script', and 'Adding Database Drivers'.

Important Commands

Icon	Command	Description
?		Displays this Help Page
📜		Shows the Command History
▶	Ctrl+Enter	Executes the current SQL statement
▶	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
🔌		Disconnects from the database

Sample SQL Script

Action	SQL Statement
Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

Adding Database Drivers


Additional database drivers can be registered by adding the Jar file location of the driver to the the env CLASSPATH. Example (Windows): to add the database driver library C:/Programs/hsqldb/lib/hsqldb.jar, H2DRIVERS to C:/Programs/hsqldb/lib/hsqldb.jar.

SQL
Panel



Preloading the Database - YAML

https://en.wikipedia.org/wiki/YAML



WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)
[Donate to Wikipedia](#)
[Wikipedia store](#)

Interaction
[Help](#)
[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact page](#)

Tools
[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Page information](#)
[Wikidata item](#)
[Cite this page](#)

Article

Talk

Read

Edit

View history

Search Wikipedia

YAML

From Wikipedia, the free encyclopedia

YAML (/ˈjæməl/, rhymes with *camel*) is a [human-readable data serialization language](#). It is commonly used for [configuration files](#), but could be used in many applications where data is being stored (e.g. debugging output) or transmitted (e.g. document headers). YAML targets many of the same communications applications as [XML](#), but has taken a more minimal approach which intentionally breaks compatibility with [SGML](#).^[1] YAML 1.2 is a [superset](#) of [JSON](#), another minimalist data serialization format where braces and brackets are used instead of indentation.^[2]

Custom data types are allowed, but YAML natively encodes [scalars](#) (such as [strings](#), [integers](#), and [floats](#)), [lists](#), and [associative arrays](#) (also known as hashes or dictionaries). These data types are based on the [Perl](#) programming language, though all commonly-used high-level programming languages share very similar concepts. YAML supports both [Python](#)-style indentation to indicate nesting, and a more compact format that uses [for lists and { for hashes.^[1] The colon-centered syntax used to express [key-value pairs](#) is inspired by [electronic mail](#) headers as defined in [RFC 0822](#), and the [document separator](#) "--" is borrowed from [MIME](#) ([RFC 2045](#)). [Escape sequences](#) are reused from [C](#), and whitespace wrapping for multi-line strings is inspired from [HTML](#). Lists and hashes can contain nested lists and hashes, forming a [tree structure](#); arbitrary [graphs](#) can be represented using YAML aliases (similar to XML in [SOAP](#)).^[1] YAML is intended to be read and written in streams, a feature inspired by [SAX](#).^[1]

Support for reading and writing YAML is available for several programming languages.^[3] Some source code editors such as [Emacs](#)^[4] and various [integrated development environments](#)^{[5][6][7]} have features that make editing YAML easier, such as folding up nested structures or automatically highlighting syntax errors.

YAML

Filename extension	.yaml , .yml
Internet media type	<i>not registered</i>
Initial release	11 May 2001; 15 years ago
Latest release	1.2 (Third Edition) (1 October 2009; 7 years ago)
Type of format	Data interchange
Open format?	Yes
Website	yaml.org

YAML is a widely used notion for representing structured information

YAML Example

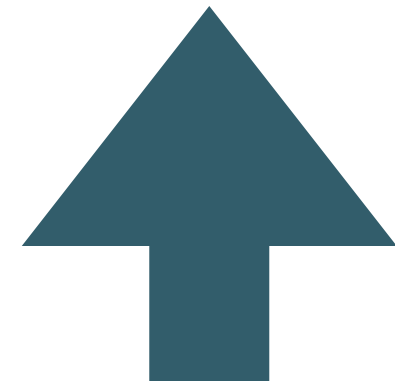
An invoice expressed via
YAML. Structure is shown
through indentation (one or
more spaces). Sequence
items are denoted by a
dash, and key value pairs
within a map are separated
by a colon.

```
invoice: 34843
date   : 2001-01-23
bill-to: &id001
  given  : Chris
  family : Dumars
  address:
    lines: |
      458 Walkman Dr.
      Suite #292
    city   : Royal Oak
    state  : MI
    postal : 48046
ship-to: *id001
product:
  - sku      : BL394D
    quantity : 4
    description : Basketball
    price     : 450.00
  - sku      : BL4438H
    quantity : 1
    description : Super Hoop
    price     : 2392.00
tax   : 251.42
total: 4443.52
comments: >
  Late afternoon is best.
  Backup contact is Nancy
  Billsmer @ 338-4338.
```



```
Song s1 = new Song("Piano Sonata No. 3", "Beethoven");
Song s2 = new Song("Piano Sonata No. 7", "Beethoven");
Song s3 = new Song("Piano Sonata No. 10", "Beethoven");
Playlist p1 = new Playlist("Beethoven Sonatas");
p1.songs.add (s1);
p1.songs.add (s2);
p1.songs.add (s3);
```

java



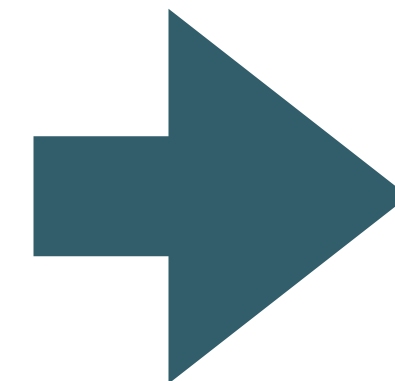
Embedded in a compiled
program.

When running, objects
occupy appropriate in
memory data structures.

Just a File format.

Used to represent structured
information in a flat file.

Must be processed by various
tools in order to be useful.



```
Song(s1):
  title: Piano Sonata No. 3
  artist: Beethoven
  duration: 5
Song(s2):
  title: Piano Sonata No. 7
  artist: Beethoven
  duration: 6
Song(s3):
  title: Piano Sonata No. 10
  artist: Beethoven
  duration: 8

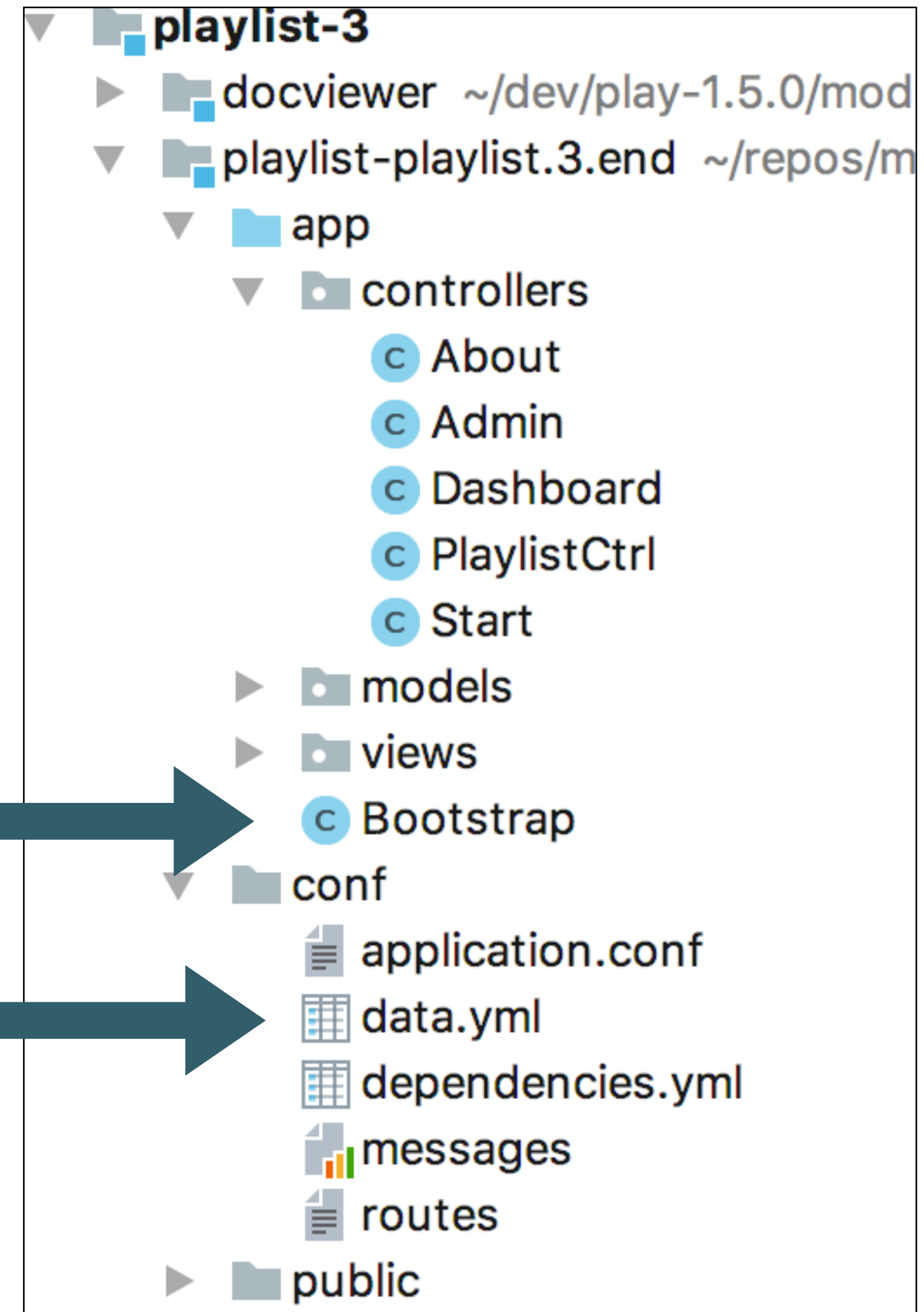
Playlist(p1):
  title: Bethoven Sonatas
  duration: 19
  songs:
    - s1
    - s2
    - s3
```

yaml

yaml in Play

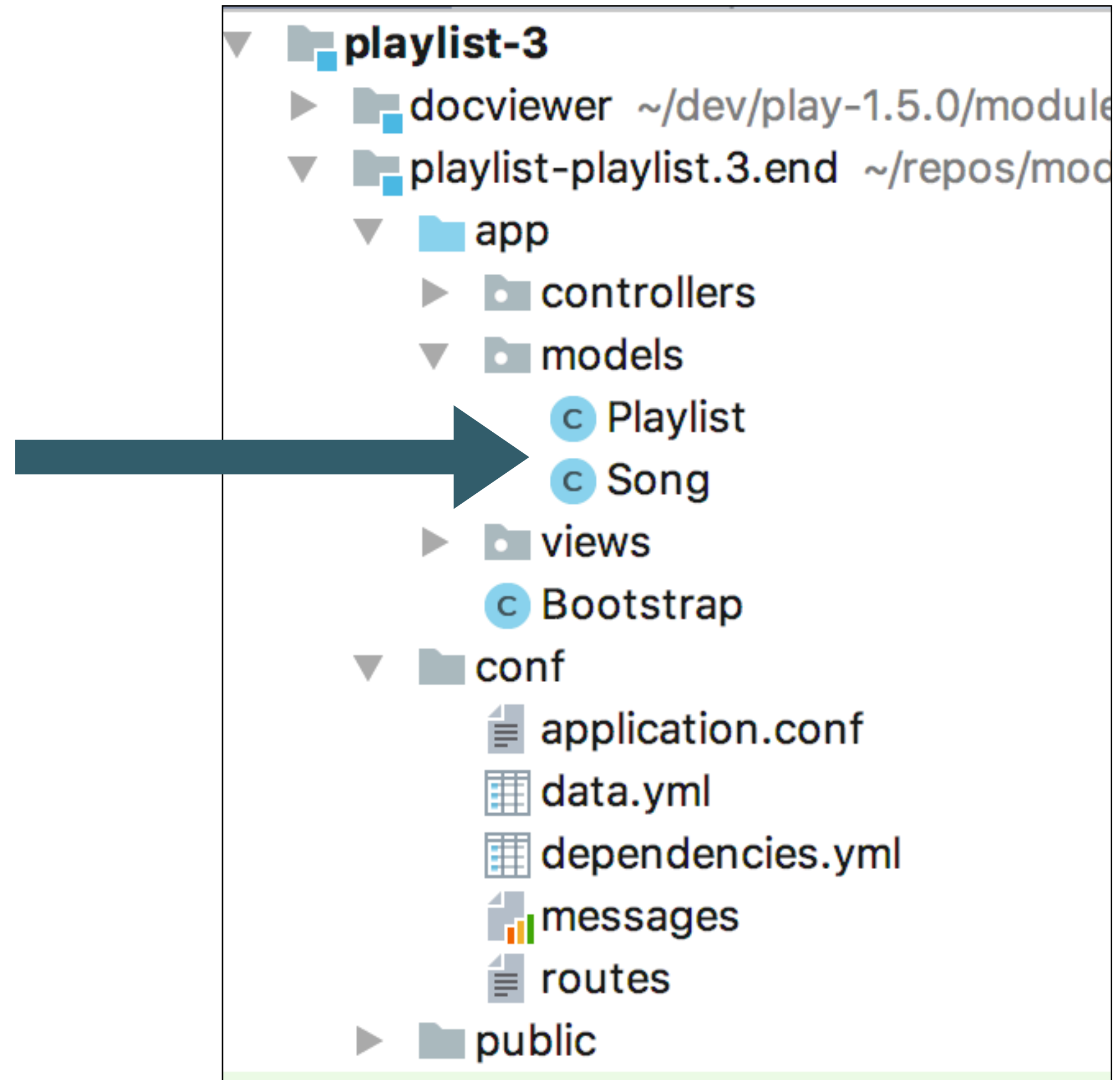
Bootstrap class
contains instruction
to load a model from
yaml file

data.yml contains the
model representation



yaml in Play

Model data will be
loaded into model
objects



Revised Model Class: Song

Plain Old Java Object (POJO)

```
package models;

public class Song
{
    public String title;
    public String artist;

    public Song(String title, String artist)
    {
        this.title = title;
        this.artist = artist;
    }
}
```

Entity Model Object

```
package models;

import javax.persistence.Entity;
import play.db.jpa.Model;

@Entity
public class Song extends Model
{
    public String title;
    public String artist;
    public int duration;

    public Song(String title, String artist, int duration)
    {
        this.title = title;
        this.artist = artist;
        this.duration = duration;
    }
}
```

“extends” from Model
class (inheritance).
Marked as “@Entity”
(Annotation).

Revised Model Class: Playlist

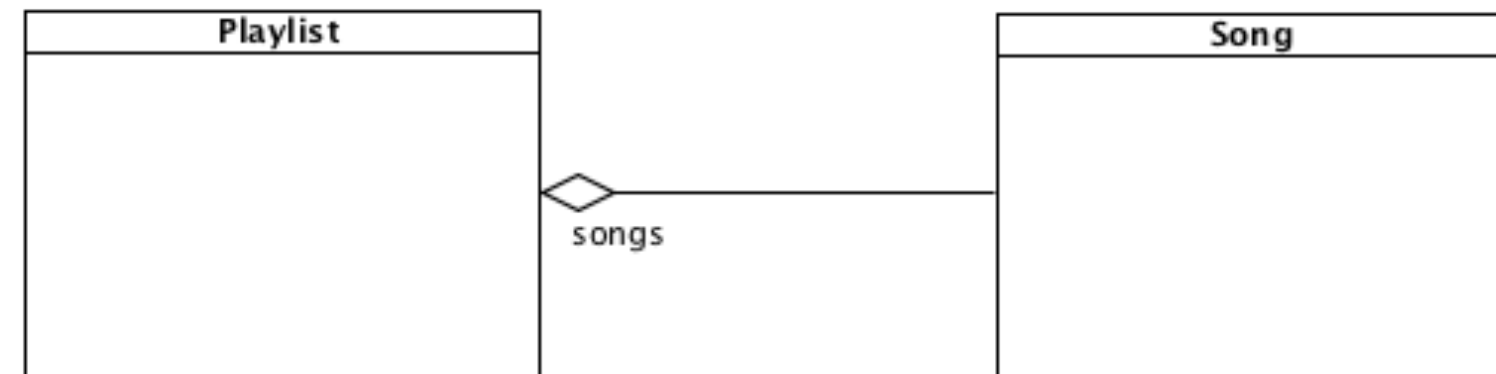
Plain Old Java Object (POJO)

```
package models;

import java.util.ArrayList;
import java.util.List;

public class Playlist
{
    public String title;
    public List<Song> songs = new ArrayList<Song>();

    public Playlist(String title)
    {
        this.title = title;
    }
}
```



“extends” from Model
class (inheritance).
Marked as “@Entity”
(Annotation).

Entity Model Object

```
package models;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.OneToMany;

import play.db.jpa.Model;

@Entity
public class Playlist extends Model
{
    public String title;

    @OneToMany(cascade = CascadeType.ALL)
    public List<Song> songs = new ArrayList<Song>();

    public Playlist(String title, int duration)
    {
        this.title = title;
        this.duration = duration;
    }
}
```

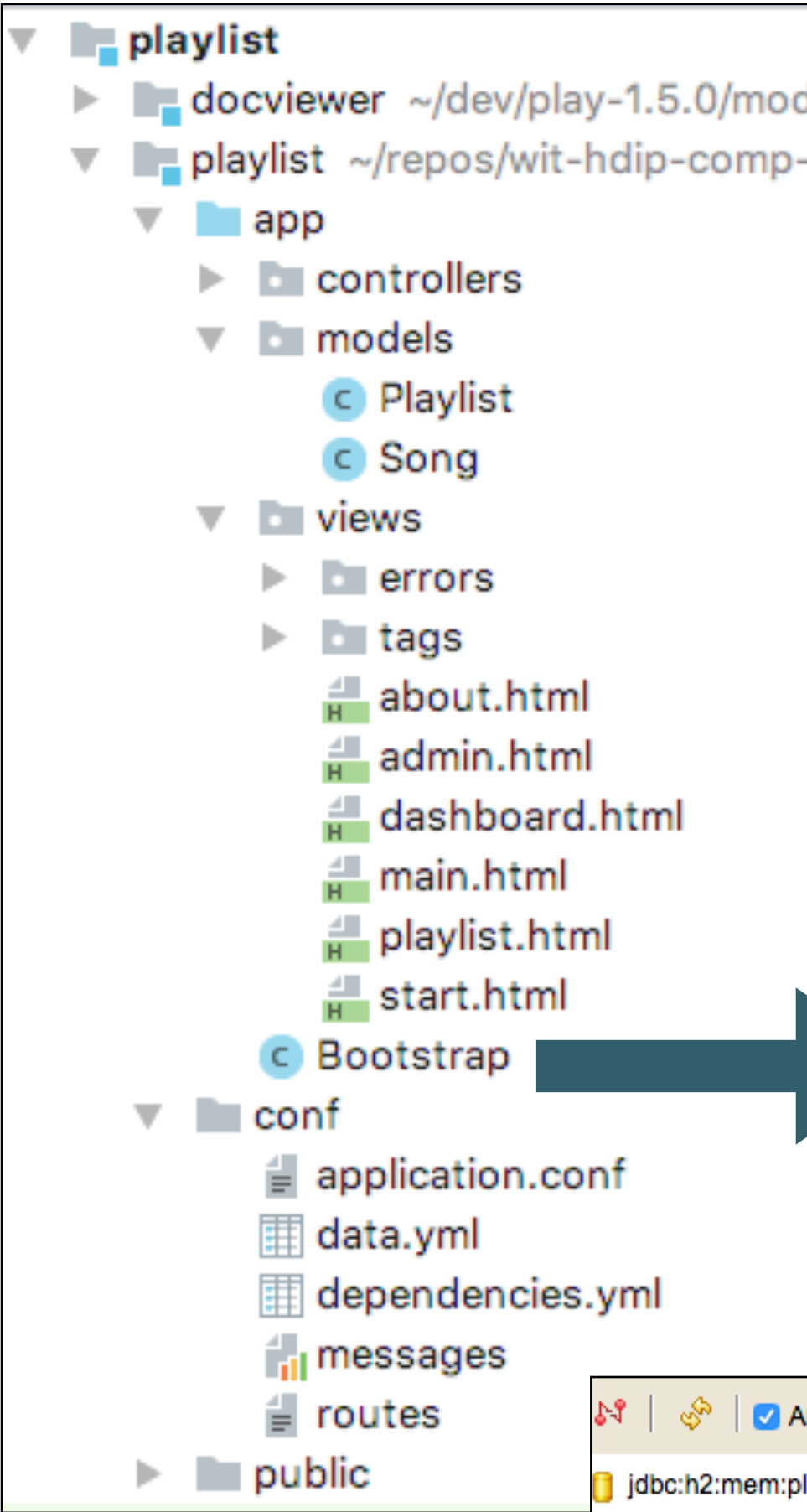
“@OneToMany” (Annotation)
describes Playlist->Song
relationship for database

Song(s1):
title: Piano Sonata No. 3
artist: Beethoven
duration: 5

Song(s2):
title: Piano Sonata No. 7
artist: Beethoven
duration: 6

Song(s3):
title: Piano Sonata No. 10
artist: Beethoven
duration: 8

Playlist(p1):
title: Bethoven Sonatas
duration: 19
songs:
- s1
- s2
- s3



```
import java.util.List;

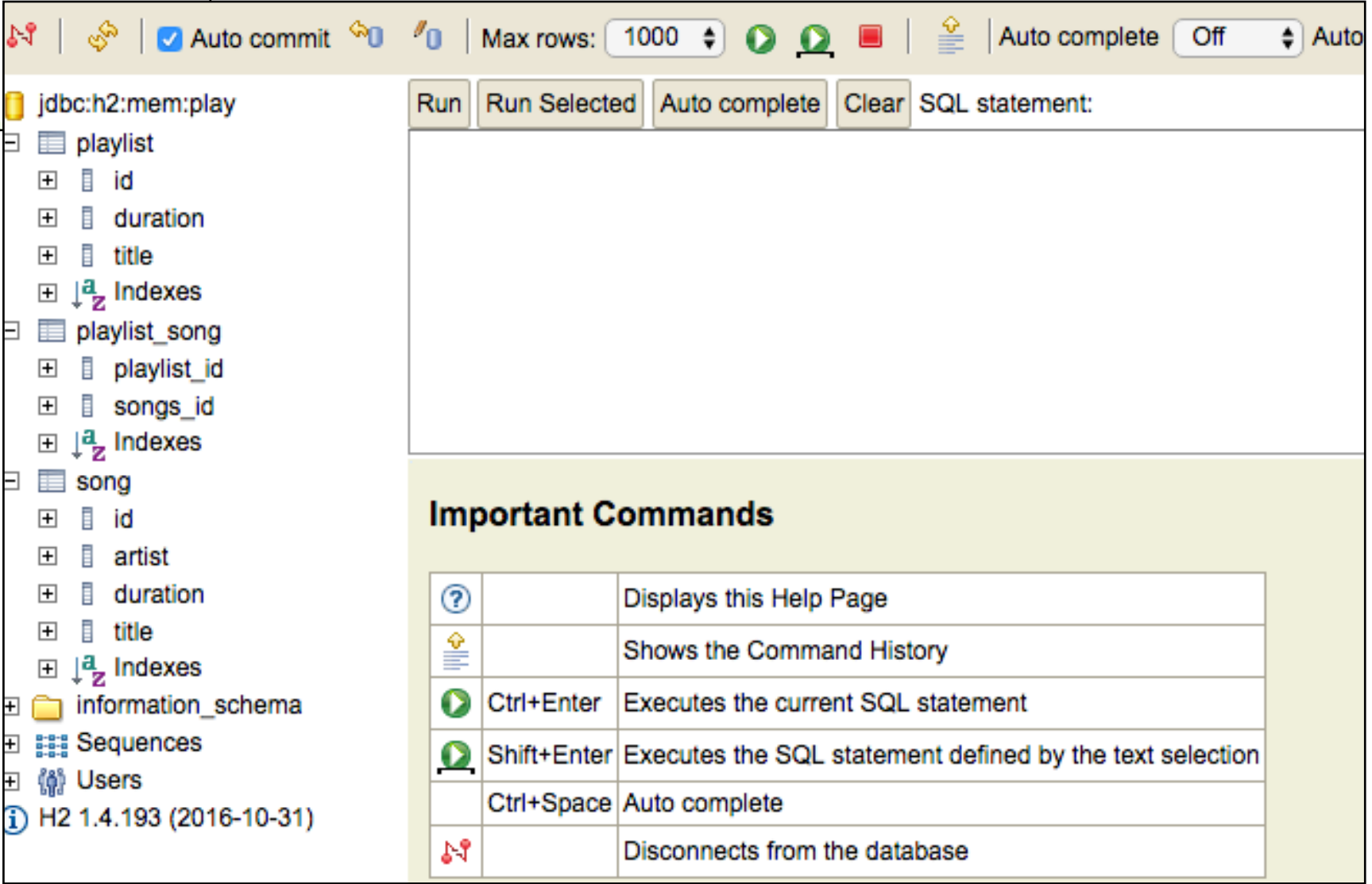
import play.*;
import play.jobs.*;
import play.test.*;

import models.*;

@OnApplicationStart
public class Bootstrap extends Job
{
    public void doJob()
    {
        Fixtures.loadModels("data.yml");
    }
}
```

localhost:9000/@db

When play app starts -
Bootstrap.doJob() called



Bootstrap class

```
iMac:playlist-2 edeleastar$ play run
~
~ _ _ _ | | _ _ _ _ _ | |
~ | ' _ \ | / _ ' | | | |
~ | _ _ / | \ _ _ _ \ _ _ ( _ )
~ | _ | _ _ _ _ _ | _ _ /
~
~ play! 1.5.0, https://www.playframework.com
~
~ Ctrl+C to stop
~
~ using java version "1.8.0_162"
Listening for transport dt_socket at address: 8000
17:53:49,806 INFO ~ Starting /Users/edeleastar/dev/playlist-2
17:53:49,906 WARN ~ You're running Play! in DEV mode
17:53:50,007 INFO ~ Listening for HTTP on port 9000 (Waiting a first request to start) ...
~ Server is up and running
17:53:59,610 INFO ~ HikariPool-1 - Starting...
17:53:59,648 INFO ~ HikariPool-1 - Start completed.
17:53:59,662 INFO ~ Connected to jdbc:h2:mem:play for default
17:54:00,703 INFO ~ Application 'playlist-2' is now started !
17:54:01,593 INFO ~ Rendering Start
```

```
import java.util.List;

import play.*;
import play.jobs.*;
import play.test.*;

import models.*;

@OnApplicationStart
public class Bootstrap extends Job
{
    public void doJob()
    {
        Fixtures.loadModels("data.yml");
    }
}
```

Application Lifecycle -> Run Bootstrap doJob
once, when application launched
This pre-loads the database with test objects

Inspecting the Playlist Table

The screenshot shows the H2 Database GUI interface. On the left, a tree view displays the database schema for 'jdbc:h2:mem:play'. The 'playlist' table is expanded, showing columns: id, duration, title, and Indexes. Below it, the 'playlist_song' table is also expanded, showing columns: playlist_id, songs_id, and Indexes. The 'song' table is also visible with columns: id, artist, duration, title, and Indexes. At the bottom of the tree, there are 'information_schema', 'Sequences', and 'Users' folders, and a version string 'H2 1.4.193 (2016-10-31)'.

The main area of the GUI shows the SQL statement 'SELECT * FROM PLAYLIST' entered in the 'SQL statement:' field. Below the statement, the results of the query are displayed in a table with 2 rows and 3 columns: ID, DURATION, and TITLE. The first row contains '1', '0', and 'Bethoven Sonatas'. The second row contains '2', '0', and 'Bethoven Concertos'. Below the table, it says '(2 rows, 3 ms)'. There is an 'Edit' button below the table.

Max rows: 1000 | Auto complete: Off | Auto select: On

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM PLAYLIST
```

ID	DURATION	TITLE
1	0	Bethoven Sonatas
2	0	Bethoven Concertos

(2 rows, 3 ms)

Edit

localhost:9000/@db

Inspecting the Songs Table

The screenshot shows the H2 database console interface. On the left is a tree view of the database schema. The main area on the right shows the SQL statement 'SELECT * FROM PLAYLIST_SONG;' and its results. The results are displayed in a table with 6 rows and 2 columns: PLAYLIST_ID and SONGS_ID. Below the table, it indicates '(6 rows, 4 ms)' and there is an 'Edit' button.

jdbc:h2:mem:play

- playlist
 - id
 - duration
 - title
 - Indexes
- playlist_song
 - playlist_id
 - songs_id
 - Indexes
- song
 - id
 - artist
 - duration
 - title
 - Indexes
- information_schema
- Sequences
- Users

H2 1.4.193 (2016-10-31)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM PLAYLIST_SONG;

PLAYLIST_ID	SONGS_ID
1	1
1	2
1	3
2	4
2	5
2	6

(6 rows, 4 ms)

Edit

localhost:9000/@db

Models



SELECT * FROM SONG;

ID	ARTIST	DURATION	TITLE
1	Beethoven	0	Piano Sonata No. 3
2	Beethoven	0	Piano Sonata No. 7
3	Beethoven	0	Piano Sonata No. 10
4	Beethoven	0	Piano Concerto No. 27
5	Beethoven	0	Piano Concertos No. 17
6	Beethoven	0	Piano Concerto No. 10

(6 rows, 6 ms)

Edit

table · row · mem · yaml ·
Song · Playlist · ArrayList ·
OneToMany · ...