# InfoSymbolServer

- Home
- Components
  - InfoSymbolServer.Domain
  - InfoSymbolServer.Infrastructure
    - Data Access
    - Background Jobs
    - Notifications
  - InfoSymbolServer.Application
  - InfoSymbolServer.Presentation
  - **InfoSymbolServer** *(current)*
- Versioning
- Configuration
- Deployment

## Overview

The InfoSymbolServer project is the entry point and startup project for the entire application. It serves as the composition root that wires together all the different layers of the application:

- Domain
- Infrastructure
- Application
- Presentation

This project doesn't contain much business logic on its own but is responsible for bootstrapping the application, configuring services, and hosting the web application.

## Key Components

### Program.cs

The `Program.cs` file is the main entry point for the application:

```
using InfoSymbolServer.Application.Extensions;
using InfoSymbolServer.Infrastructure.Extensions;
using InfoSymbolServer.Presentation.Extensions;
using Serilog;

var builder = WebApplication.CreateBuilder(args);
```

```
builder.Host.UseSerilog((context, config) =>
    config.ReadFrom.Configuration(context.Configuration));

builder.Services.AddInfrastructure(builder.Configuration,
builder.Environment);
builder.Services.AddApplication();
builder.Services.AddPresentation();

builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

app.ApplyMigrations();

// Adds all three supported exchanges to database.
await app.SeedExchangesAsync();

// Validate notification configurations
app.ValidateNotificationConfigurations();

app.AddPresentation(app.Environment);

app.UseSerilogRequestLogging();

app.Run();

public partial class Program { }
```

The startup process follows these steps:

1. **Serilog Configuration**: Sets up structured logging with Serilog, reading settings from the configuration
2. **Infrastructure Setup**: Registers infrastructure services including database access, repositories, and background jobs
3. **Application Setup**: Registers application services, AutoMapper, and validation
4. **Presentation Setup**: Registers controllers, API endpoints, and middleware
5. **Database Migrations**: Applies any pending database migrations
6. **Application Pipeline**: Configures the HTTP request pipeline with middleware

## Extension Methods

The startup project uses extension methods from each layer to configure the application:

### Infrastructure Extensions

```
builder.Services.AddInfrastructure(builder.Configuration,
builder.Environment);

await app.SeedExchangesAsync();

app.ApplyMigrations();
app.ValidateNotificationConfigurations();
```

These methods:

- Configure the database context
- Register repositories
- Set up Quartz.NET for background jobs
- Apply any pending database migrations
- Adds supported exchanges
- Validates notification services configuration

### Application Extensions

```
builder.Services.AddApplication();
```

This method:

- Registers application services
- Sets up AutoMapper for object mapping
- Configures validation
- Registers Problem Details for standardized error responses

### Presentation Extensions

```
builder.Services.AddPresentation();
app.AddPresentation(app.Environment);
```

These methods:

- Register controllers
- Configure Swagger documentation
- Set up exception handling middleware
- Configure routing and endpoints

## Dependencies

The startup project references all other projects in the solution:

- **InfoSymbolServer.Domain**: Core entities and business logic
- **InfoSymbolServer.Infrastructure**: Database access, external integrations
- **InfoSymbolServer.Application**: Application services and DTOs
- **InfoSymbolServer.Presentation**: API controllers and middleware