In [12]:
```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap1\sourcecode")
# 导入基础计算库
import numpy as np
# 导入绘图库
import matplotlib.pyplot as plt
# 导入线性回归模型
from sklearn.linear_model import Ridge, LassoCV
# 导入管道处理工具
from sklearn.pipeline import make_pipeline
# 导入数值计算库
import scipy as sp
# 导入数据分析库
import pandas as pd
# 导入模型评估的工具
from sklearn.metrics import mean_squared_error, r2_score
# 导入数据集获取工具
from sklearn.datasets import fetch_openml
# 导入元回归估计器
from sklearn.compose import TransformedTargetRegressor
# 导入数据集划分工具
from sklearn.model_selection import train_test_split
# 导入模型评估工具
from sklearn.metrics import median_absolute_error, PredictionErrorDisplay
# 导入列转换工具
from sklearn.compose import make_column_transformer
# 导入one-hot编码工具
from sklearn.preprocessing import OneHotEncoder
# 导入统计绘图库
import seaborn as sns
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用seaborn风格绘图
plt.style.use("seaborn-v0_8")
```

In [13]:
```
# 导入数据集
survey = fetch_openml(data_id=534, as_frame=True, parser="pandas")
# 协变量X
X = survey.data[survey.feature_names]
print("解释变量X的描述性统计表如下：", X.describe(include="all"), sep="\n")
# 响应变量y
y = survey.target.values.ravel()
print("y的前五行为：", survey.target.head(), sep="\n")
```

解释变量X的描述性统计表如下：

	EDUCATION	SOUTH	SEX	EXPERIENCE	UNION	AGE	RACE
count	534.000000	534	534	534.000000	534	534.000000	534
unique	NaN	2	2	NaN	2	NaN	3
top	NaN	no	male	NaN	not_member	NaN	White
freq	NaN	378	289	NaN	438	NaN	440
mean	13.018727	NaN	NaN	17.822097	NaN	36.833333	NaN
std	2.615373	NaN	NaN	12.379710	NaN	11.726573	NaN
min	2.000000	NaN	NaN	0.000000	NaN	18.000000	NaN
25%	12.000000	NaN	NaN	8.000000	NaN	28.000000	NaN
50%	12.000000	NaN	NaN	15.000000	NaN	35.000000	NaN
75%	15.000000	NaN	NaN	26.000000	NaN	44.000000	NaN
max	18.000000	NaN	NaN	55.000000	NaN	64.000000	NaN

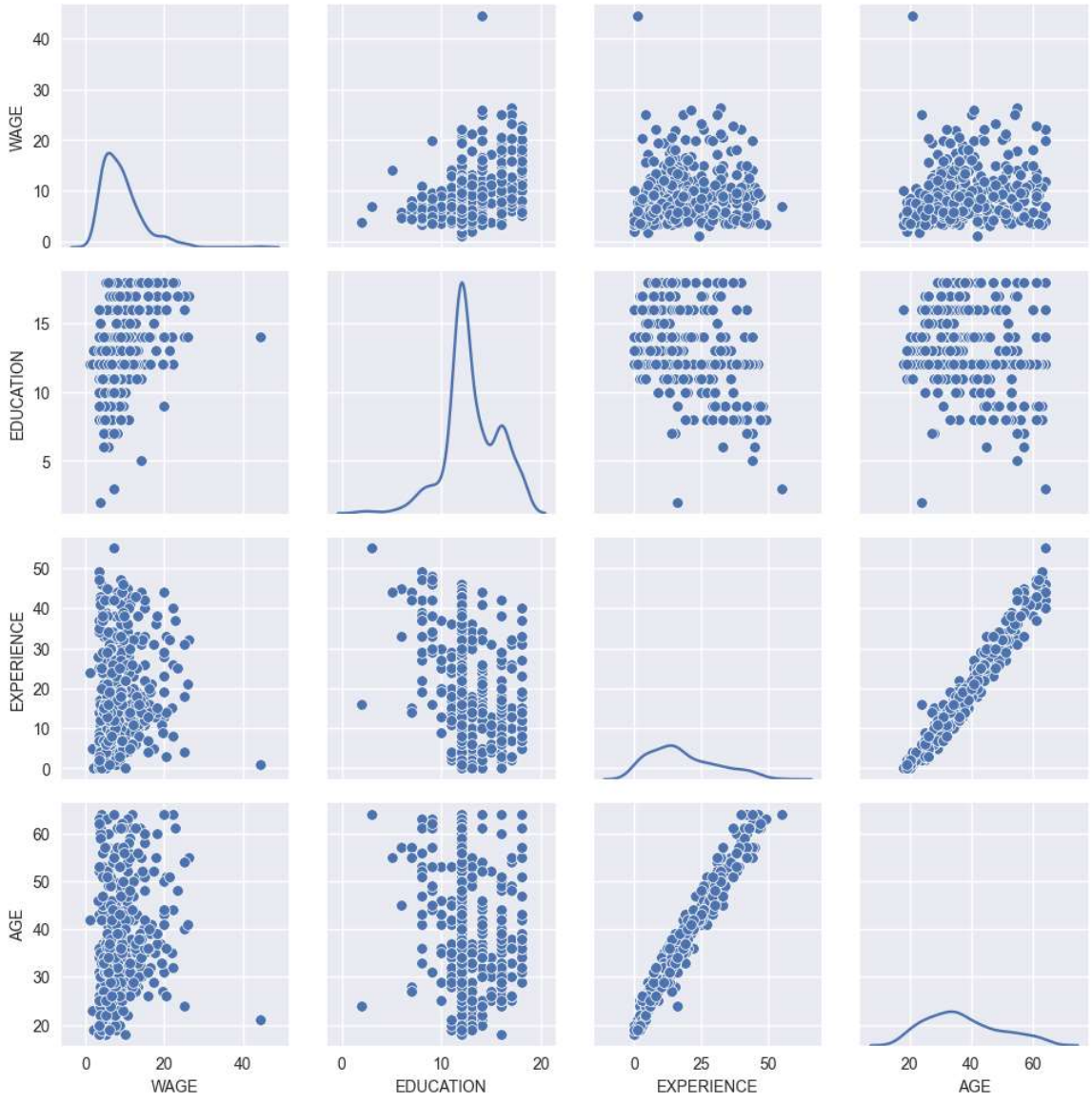
	OCCUPATION	SECTOR	MARR
count	534	534	534
unique	6	3	2
top	Other	Other	Married
freq	156	411	350
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

y的前五行为：

```
0    5.10
1    4.95
2    6.67
3    4.00
4    7.50
Name: WAGE, dtype: float64
```

In [14]:

```python
# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=42, test_size=0.25
)
# 复制一份训练集
train_dataset = X_train.copy()
# 插入一列数据，作为第一列
train_dataset.insert(0, "WAGE", y_train)
# 绘制矩阵散点图
fig = sns.PairGrid(train_dataset)
# 对角线上的图形
fig.map_diag(sns.kdeplot)
# 非对角线上的图形
fig.map_offdiag(sns.scatterplot)
fig.savefig("../codeimage/code6.pdf")
```

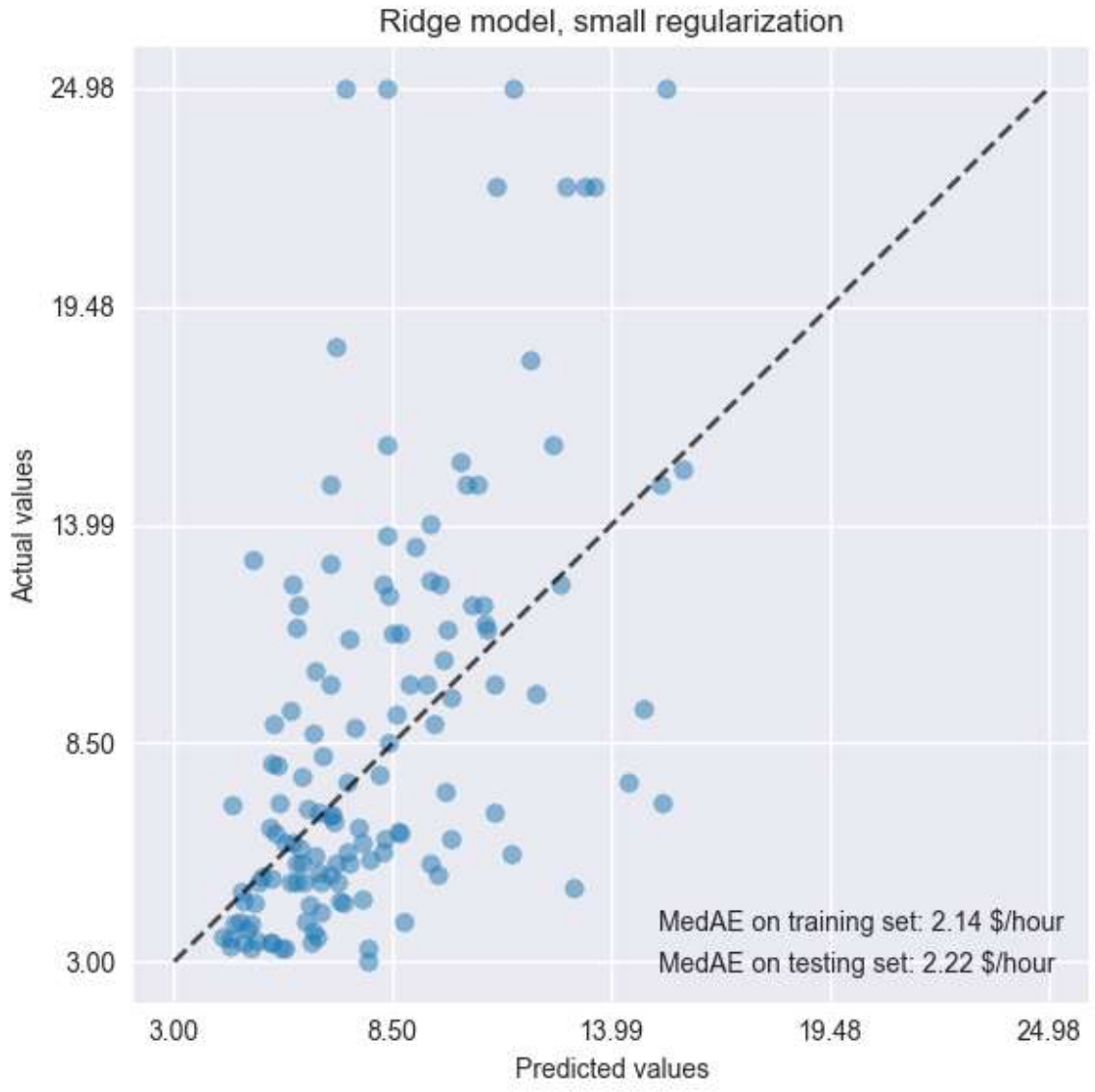


In [15]:
```
# 查看下数据变量的变量情况
print("数据集变量的情况：")
survey.data.info()
```

```
数据集变量的情况：
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 534 entries, 0 to 533
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   EDUCATION   534 non-null    int64   
 1   SOUTH       534 non-null    category
 2   SEX         534 non-null    category
 3   EXPERIENCE  534 non-null    int64   
 4   UNION       534 non-null    category
 5   AGE         534 non-null    int64   
 6   RACE        534 non-null    category
 7   OCCUPATION  534 non-null    category
 8   SECTOR      534 non-null    category
 9   MARR        534 non-null    category
dtypes: category(7), int64(3)
memory usage: 17.2 KB
```

In [16]:
```
# 对分类变量进行one-hot编码
# 分类变量的列名
```

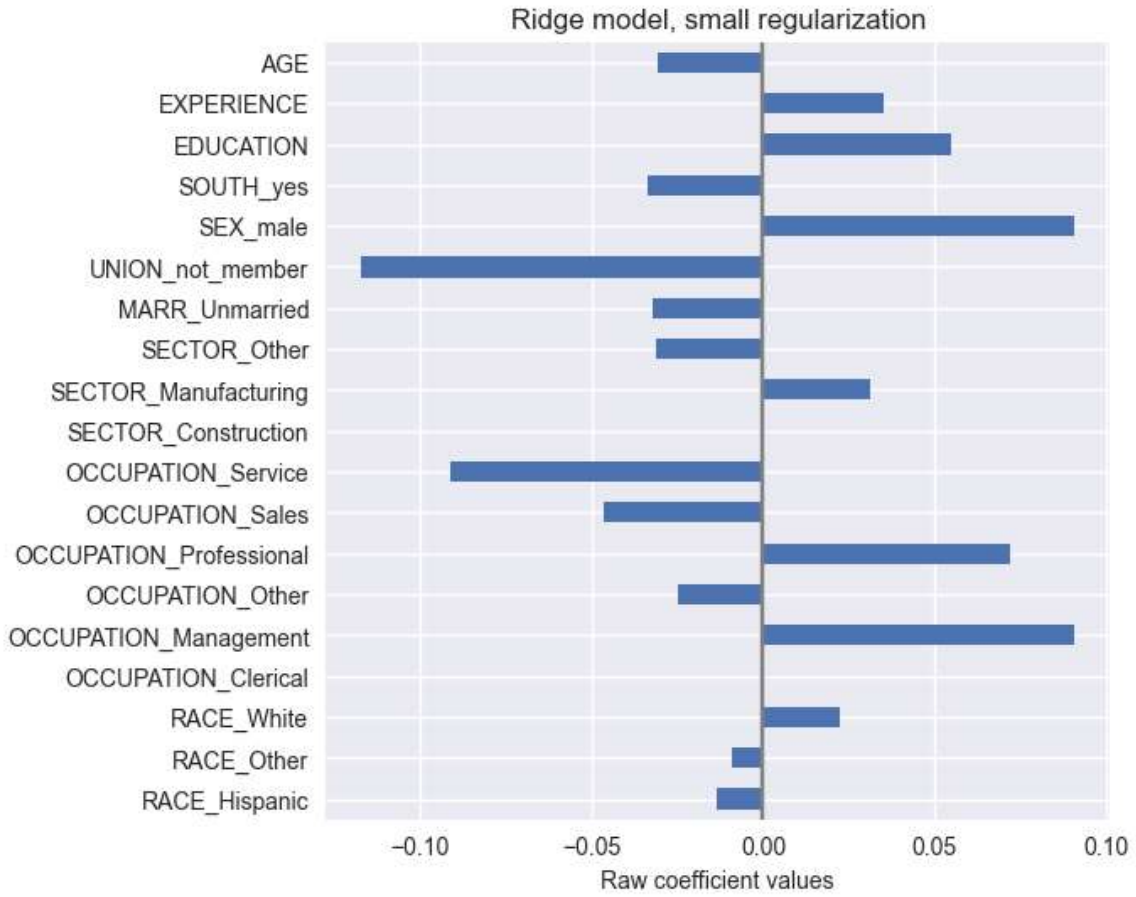
```
categorical_columns = [
    "RACE", "OCCUPATION", "SECTOR",
    "MARR", "UNION", "SEX", "SOUTH"
]
# 数值变量的列名
numerical_columns = ["EDUCATION", "EXPERIENCE", "AGE"]
# 进行分类变量列之间的one-hot编码
preprocessor = make_column_transformer(
    (
        OneHotEncoder(drop="if_binary"), # one-hot编码
        categorical_columns # 对这些分类变量
    ),
    remainder="passthrough", # 保留非分类变量
    verbose_feature_names_out=False
)
# 构造岭回归模型，惩罚系数非常小，接近于OLS
model = make_pipeline(
    preprocessor, # preprocess对象
    TransformedTargetRegressor(
        regressor=Ridge(alpha=1e-10), # 模型
        func=np.log10, # 它将作用于目标变量wage上
        inverse_func=sp.special.exp10
    ),
)
# 模型拟合
model.fit(X_train, y_train)
# 预测
y_train_fit = model.predict(X_train)
# 训练集上的绝对误差的中位数
mae_train = median_absolute_error(y_train, y_train_fit)
# 预测
y_pred = model.predict(X_test)
# 测试集上的绝对误差中位数
mae_test = median_absolute_error(y_test, y_pred)
scores = {
    "MedAE on training set": "{:.2f} $/hour".format(mae_train),
    "MedAE on testing set": "{:.2f} $/hour".format(mae_test)
}
# 开始绘图
fig2, ax = plt.subplots(figsize=(6, 6))
display = PredictionErrorDisplay.from_predictions(
    y_test, y_pred,
    kind="actual_vs_predicted",
    ax=ax,
    scatter_kwargs={"alpha": 0.5}
)
ax.set_title("Ridge model, small regularization")
# 添加图例
for name, score in scores.items():
    ax.plot([], [], " ", label=f"{name}: {score}")
ax.legend(loc="lower right")
plt.tight_layout()
plt.show()
fig2.savefig("../codeimage/code7.pdf")
```



In [17]:
```
# 查看下岭回归的系数估计值
# 系数对应的变量名
feature_names = model[:-1].get_feature_names_out()
# 构造dataframe
coefs = pd.DataFrame(
    model[-1].regressor_.coef_,
    columns=["Coefficients"],
    index=feature_names,
)
print("系数估计值为：", coefs, sep="\n")
# 图形展示系数估计值
fig3, ax = plt.subplots(figsize=(6,6))
# 水平柱状图
coefs.plot(kind="barh", ax=ax)
# 设置标题
ax.set_title("Ridge model, small regularization")
# 绘制一条竖直线
ax.axvline(x=0, color=".5")
# 不显示图例，默认显示
ax.legend([])
# 设置横纵标签
ax.set_xlabel("Raw coefficient values")
plt.show()
fig3.savefig("../codeimage/code8.pdf")
```

系数估计值为：

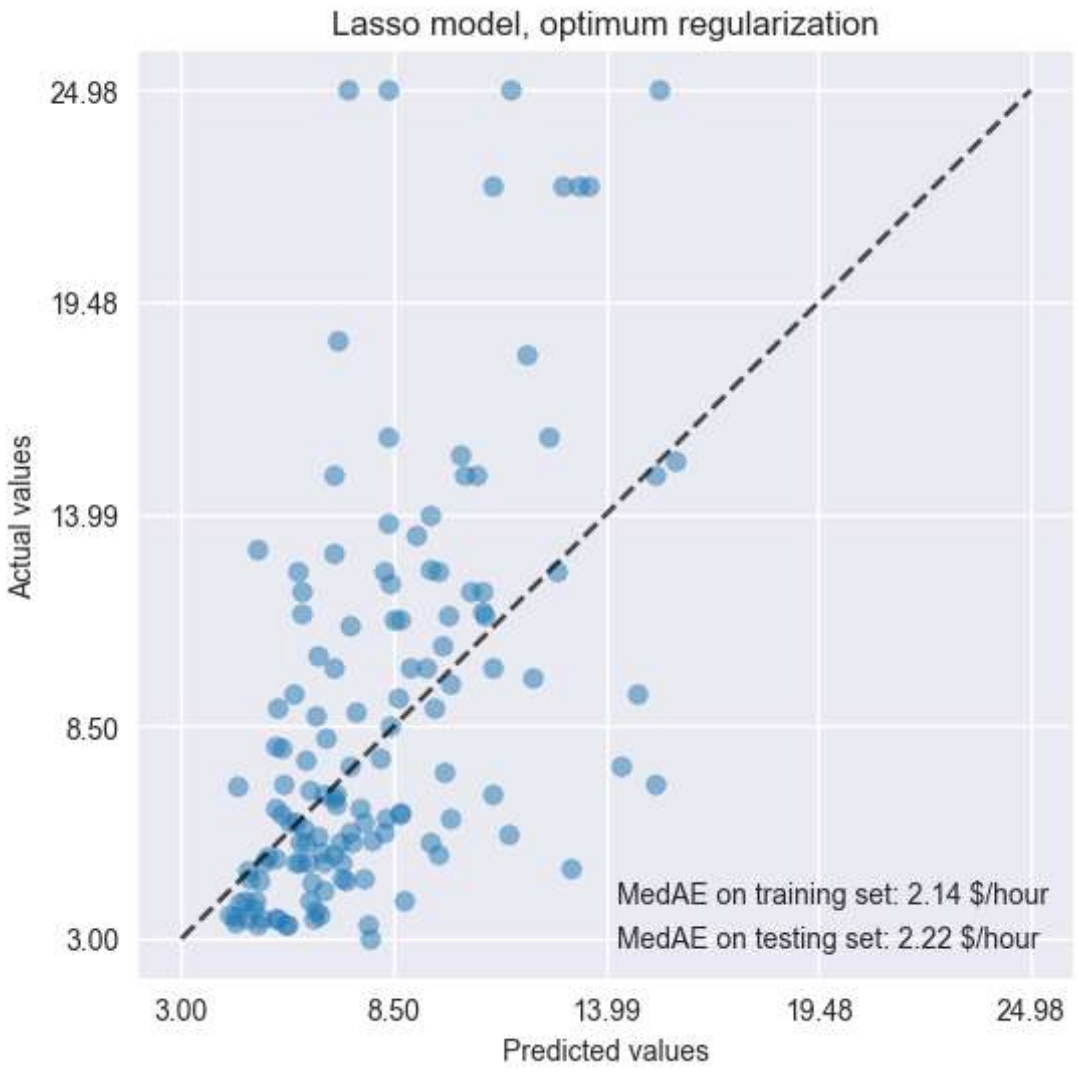
	Coefficients
RACE_Hispanic	-0.013558
RACE_Other	-0.009114
RACE_White	0.022555
OCCUPATION_Clerical	0.000062
OCCUPATION_Management	0.090545
OCCUPATION_Other	-0.025084
OCCUPATION_Professional	0.071981
OCCUPATION_Sales	-0.046619
OCCUPATION_Service	-0.091036
SECTOR_Construction	-0.000188
SECTOR_Manufacturing	0.031265
SECTOR_Other	-0.031015
MARR_Unmarried	-0.032405
UNION_not_member	-0.117154
SEX_male	0.090808
SOUTH_yes	-0.033823
EDUCATION	0.054699
EXPERIENCE	0.035005
AGE	-0.030867



In [18]:
```
# 使用Lasso模型来拟合
# Lasso惩罚系数
alphas = np.logspace(-10, 10, 21)
# 构建LassoCV模型
model = make_pipeline(
    preprocessor,
    TransformedTargetRegressor(
        regressor=LassoCV(alphas=alphas, max_iter=100000),
        func=np.log10,
        inverse_func=sp.special.exp10,
    ),
```

```
)
# 模型拟合
model.fit(X_train, y_train)
print("所选的lasso模型对应的系数为：", model[-1].regressor_.alpha_, sep="\n")
# 模型预测训练集
y_pred_lasso_train = model.predict(X_train)
mae_train = median_absolute_error(y_train, y_pred_lasso_train)
# 模型预测测试集
y_pred_lasso_test = model.predict(X_test)
mae_test = median_absolute_error(y_test, y_pred_lasso_test)
# 开始绘图
fig4, ax = plt.subplots(figsize=(6, 6))
display = PredictionErrorDisplay.from_predictions(
    y_test, y_pred,
    kind="actual_vs_predicted",
    ax=ax,
    scatter_kwargs={"alpha": 0.5}
)
ax.set_title("Lasso model, optimum regularization")
# 设置图例
for name, score in scores.items():
    ax.plot([], [], " ", label=f"{name}: {score}")
ax.legend(loc="lower right")
plt.show()
fig4.savefig("../codeimage/code9.pdf")
```

所选的lasso模型对应的系数为：
0.001



In [19]:
```
# 绘制系数估计的条行图
fig5, ax = plt.subplots(figsize=(6,6))
coefs = pd.DataFrame(
    model[-1].regressor_.coef_,
    columns=["Coefficients importance"],
    index=feature_names,
)
coefs.plot(kind="barh", ax=ax)
# 不显示图例，默认显示
ax.legend([])
ax.set_title("Lasso model, optimum regularization, normalized variables")
ax.axvline(x=0, color=".5")
plt.show()
fig5.savefig("../codeimage/code10.pdf")
```